Fault-Tolerant Private Information Retrieval via Threshold Distributed Point Functions

Dazeng Yuan , Xiheng Liu , Bin Liu *

Posted Date: 12 May 2026

doi: 10.20944/preprints202605.0700.v1

Keywords: private information retrieval; function secret sharing; distributed point function; secure multi-party computation

 doi:10.20944/preprints202605.0700.v1

Article

Fault-Tolerant Private Information Retrieval via Threshold Distributed Point Functions

Dazeng Yuan [1], Xiheng Liu [2] and Bin Liu [2,*]

[1] Dazhou Vocational College of Chinese Medicine, Dazhou 635000, China
[2] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610032, China
* Correspondence: liubin10@foxmail.com

Abstract

Multi-server private information retrieval (PIR) based on function secret sharing (FSS) has emerged as a prominent paradigm for achieving sublinear communication. However, standard FSS constructions strictly require full server participation, making them highly vulnerable to single-node fail-stop faults. Existing fault-tolerant schemes mitigate this but inevitably inflate the downlink response overhead to scale with the database size N (e.g., $O(\sqrt{N})$). To overcome this limitation, we propose a (t, p)-fault-tolerant PIR (FT-PIR) protocol grounded in a newly designed generalized (t, p)-fault-tolerant distributed point function (FT-DPF). By introducing a hierarchical recursive patching mechanism, our scheme transforms rigid all-party evaluations into flexible t-out-of-p reconstructions. This architecture completely decouples the response communication from N and ensures efficient client-side reconstruction via lightweight XOR aggregations, fundamentally bypassing heavy algebraic interpolations. Formal analysis proves that our strictly stateless protocol guarantees $(t-1)$-computational privacy under the semi-honest model. Asymptotic evaluations demonstrate that the proposed FT-PIR achieves an optimal downlink complexity bounded to $O(poly(t, p) \cdot \log p)$, significantly outperforming existing robust baselines for large-scale datasets.

Keywords: private information retrieval; function secret sharing; distributed point function; secure multi-party computation

1. Introduction

Private information retrieval (PIR) is a fundamental cryptographic primitive that enables a client to retrieve specific records from a public database containing N items without revealing the query index [1]. As a cornerstone for privacy-preserving data access, PIR has seen widespread adoption in private contact discovery [2], private web search [3], anonymity networks [4], and encrypted search systems [5]. Historically, PIR research has bifurcated into two primary architectures based on the underlying trust model: single-server and multi-server PIR.

Single-server PIR operates without non-collusion assumptions and is predominantly driven by two technical paradigms: homomorphic encryption (HE) and hint-based preprocessing. However, both paradigms encounter inherent mechanistic bottlenecks. In HE-based constructions, schemes employing packing techniques to optimize bandwidth [6] suffer from prohibitive homomorphic computational overheads. Even with homomorphic Thorp shuffles to minimize circuit depth [7], the high constant-factor costs of HE evaluations remain a significant barrier. To accelerate online performance, recent schemes introduced preprocessing mechanisms [8] to achieve sublinear query times, albeit by imposing substantial state storage overhead on the client. Alternatively, a parallel route leverages hint-based preprocessing [9–12]. While these schemes utilize lightweight symmetric primitives to yield attractive time-space trade-offs, they mandatorily require exhaustive offline database scans. Furthermore, their hint mechanisms face periodic and heavy reconstruction burdens due to limited query budgets, restricting their practical viability in high-frequency scenarios.

 doi:10.20944/preprints202605.0700.v1

To circumvent the performance bottlenecks of single-server models, multi-server PIR achieves highly efficient retrieval by distributing trust across p non-colluding servers. Under this architecture, information-theoretic (IT) constructions based on algebraic coding have been extensively studied [13]. These schemes utilize polynomials and erasure codes to establish a communication efficiency of $\mathcal{O}(N^{1/3})$ [14] and optimal robustness bounds [15], ultimately achieving sub-polynomial communication via matching vector codes [16]. Despite these theoretical milestones, IT-PIR typically involves complex finite-field operations that concretely diminish their practical bandwidth advantages. Although recent advancements like multi-server preprocessing [17] attempt to amortize online computation, they inevitably force the client to maintain persistent states, failing to simultaneously balance sub-polynomial communication, stateless operation, and efficient fault tolerance.

Parallel to IT-PIR, the function secret sharing (FSS) paradigm has emerged as a highly promising trajectory for constructing lightweight multi-server PIR [18][19]. Specifically, its core instantiation for point queries—the distributed point function (DPF)—enables clients to compress query key overhead to $\mathcal{O}(\log N)$. However, scaling standard DPFs from a two-server to a p-server environment ($p \geq 3$) initially introduced severe scalability issues, with request bandwidth exhibiting exponential growth ($\mathcal{O}(2^p\sqrt{N})$) relative to the number of servers [18]. While recent combinatorial design optimizations have successfully compressed this multi-party key size to $\mathcal{O}(p^3\sqrt{N})$ [20], a far more critical vulnerability persists: standard FSS-based PIR protocols are exceptionally fragile as they strictly require all p servers to remain online during the evaluation and reconstruction phases. Recent state-of-the-art works have attempted to enhance system robustness by introducing fault-tolerant DPF mechanisms [21]; however, such improvements implicitly inflate the response overhead to $\mathcal{O}(\sqrt{N})$.

To overcome this fundamental limitation, we propose a novel (t, p)-fault-tolerant PIR scheme (FT-PIR) based on a newly designed fault-tolerant distributed point function (FT-DPF). Our scheme employs a recursive patching mechanism to extend a base two-server DPF into a (t, p)-threshold setting. This design enables the client to correctly reconstruct the target data using responses from any t surviving servers, achieving a robust yet lightweight retrieval process as comprehensively compared with state-of-the-art PIR protocols in Table 1.

Table 1. Comparison with prior works.

Schemes	Fault-Tolerate	Offline Communication	Request Communication	Response Communication	Extra Space
[10]	×	$\mathcal{O}(N)$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{N})$
[11]	×	$\mathcal{O}(N)$	$\mathcal{O}(N^{1/4})$	$\mathcal{O}(N^{1/4})$	$\mathcal{O}(\sqrt{N})$
[18]	×	0	$\mathcal{O}(2^p \cdot \sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
[21]	✓	0	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$
Ours	✓	0	$\mathcal{O}(\mathsf{poly}(t,p) \cdot \log p \cdot \sqrt{N})$	$\mathcal{O}(\mathsf{poly}(t,p) \cdot \log p)$	$\mathcal{O}(1)$

Here, N denotes the number of items in database, p denotes the number of servers, t denotes the threshold. The extra space denotes the client's extra storage.

1.1. Our Contributions

The main contributions of this paper are summarized as follows:

- **A Novel (t, p)-Fault-Tolerant DPF Scheme:** We formally define and construct a generalized (t, p)-fault-tolerant distributed point function (FT-DPF) through a hierarchical, bottom-up design. Specifically, we first optimize the key length of the state-of-the-art multi-party DPF [20]. Building upon this optimized primitive, we construct a highly efficient base $(2, p)$ DPF scheme. Finally, we introduce a recursive patching mechanism to seamlessly extend this base construction into a generalized (t, p)-threshold setting. This allows any t surviving servers to correctly reconstruct the evaluation, effortlessly tolerating up to $p - t$ fail-stop drop-outs.
  Database-Size-Independent Downlink FT-PIR Protocol: Built upon our generalized FT-DPF, we propose a strictly stateless, fault-tolerant PIR protocol. Unlike Shamir-based schemes burdened by an $\mathcal{O}(\sqrt{N})$ response overhead, our architecture completely decouples downlink communication

 doi:10.20944/preprints202605.0700.v1

from N, bounding the complexity to $\mathcal{O}(poly(t,p)\cdot \log p)$. Given that practical non-colluding deployments treat p as a small constant relative to massive datasets ($N \gg p \geq t$), this design achieves optimal scalability for massive datasets. Furthermore, the client-side reconstruction relies purely on straightforward XOR aggregations, avoiding the heavy algebraic interpolations typical of traditional robust schemes.

- **Formal Security and Efficiency Analysis:** We provide a formal security analysis and prove that the proposed protocol achieves $(t-1)$-computational privacy under the semi-honest model. Detailed asymptotic evaluations show that our scheme completely decouples the downlink communication cost from the database size compared to the best-known robust PIR scheme [21], and achieves strictly sublinear query efficiency.

1.2. Organization

The remainder of this paper is organized as follows. Section 2 introduces the necessary cryptographic preliminaries. Section 3 formalizes the system model and definitions. Building upon this foundation, Section 4 details the core technical contribution: the hierarchical construction of the generalized threshold DPF. Section 5 then deploys this primitive to construct the fault-tolerant PIR protocol. Finally, Section 6 reviews the related literature, and Section 7 concludes the paper.

2. Preliminaries

2.1. Distributed Point Function

A distributed point function (DPF) [18] [19] is a cryptographic primitive that enables a client to secretly share a point function across multiple servers. Let $(\mathbb{G},+)$ be an Abelian group. A point function $f_{\alpha,\beta}:\{0,1\}^n \to \mathbb{G}$ is defined such that $f_{\alpha,\beta}(x)=\beta$ if $x=\alpha$, and 0 otherwise, where $\alpha \in \{0,1\}^n$ is the target input and $\beta \in \mathbb{G}\setminus\{0\}$ is the non-zero output value. In a p-party DPF, the client splits $f_{\alpha,\beta}$ into p distinct keys $k_1,\ldots,k_p$. This splitting guarantees that any strict subset of the keys reveals no information about the secret parameters (α,β), yet the aggregated evaluations of all keys at a given point x correctly reconstruct $f_{\alpha,\beta}(x)$. Conceptually, these keys act as secret shares of a vector of length $N=2^n$, where all elements are zero except for the index corresponding to α. Formally, a DPF scheme is defined as follows:

Definition 1 (Distributed Point Function [18]). *A p-party DPF scheme over the domain $\{0,1\}^n$ and an Abelian group $(\mathbb{G},+)$ comprises two probabilistic polynomial-time (PPT) algorithms $(\mathsf{Gen},\mathsf{Eval})$ satisfying the following properties:*

- ***Syntax:***
  - $(k_1,\ldots,k_p)\leftarrow \mathsf{Gen}(1^\lambda,\alpha,\beta)$*: Takes as input a security parameter λ, a target index $\alpha\in\{0,1\}^n$, and a payload $\beta\in\mathbb{G}$. It outputs a p-tuple of keys $(k_1,\ldots,k_p)$.*
  - $y_i\leftarrow \mathsf{Eval}(i,k_i,x)$*: Takes as input a party index $i\in[p]$, the corresponding key k_i, and an evaluation point $x\in\{0,1\}^n$. It outputs a group element $y_i\in\mathbb{G}$.*
- ***Correctness:*** *For any $\alpha\in\{0,1\}^n$, $\beta\in\mathbb{G}$, and any $x\in\{0,1\}^n$, if $(k_1,\ldots,k_p)\leftarrow\mathsf{Gen}(1^\lambda,\alpha,\beta)$, then it holds with probability 1 that $\sum_{i=1}^{p}\mathsf{Eval}(i,k_i,x)=f_{\alpha,\beta}(x)$.*
- ***Privacy:*** *For any strict subset of indices $\mathcal{I}\subset[p]$ (i.e., $|\mathcal{I}|<p$), the joint distribution of the keys $\{k_i\}_{i\in\mathcal{I}}$ generated by $\mathsf{Gen}(1^\lambda,\alpha,\beta)$ is computationally indistinguishable from a distribution of keys simulated without the knowledge of (α,β).*

2.2. (t,p)-Threshold Replicated Secret Sharing

As the fundamental cryptographic primitive underlying our proposed fault-tolerant PIR protocol, we employ (t,p)-threshold replicated secret sharing (RSS) [22]. Unlike standard full-participation secret sharing schemes, a (t,p)-threshold RSS provides flexible resilience in distributed environments: it enables a secret to be distributed across p servers such that any subset of t surviving servers can correctly reconstruct it, while any unauthorized coalition of up to $t-1$ servers obtains zero information

 doi:10.20944/preprints202605.0700.v1

regarding the secret. Adapted from the conceptual framework of [22], the formal syntax and security properties of this primitive are defined as follows.

Definition 2 ((t, p)-Threshold Replicated Secret Sharing [22]). *Let $\mathbb{F}$ be a finite field. A (t, p)-threshold replicated secret sharing (RSS) scheme for a secret $s \in \mathbb{F}$ consists of a pair of polynomial-time algorithms $(\mathsf{Share}_{\mathsf{tofp}}, \mathsf{Rec}_{\mathsf{tofp}})$ defined as follows:*

- $(sh_1, \ldots, sh_p) \leftarrow \mathsf{Share}_{\mathsf{tofp}}(s, t, p)$*: A randomized distribution algorithm executed by the dealer. It takes as input a secret s, a reconstruction threshold t, and the total number of servers p. It outputs p share sets, privately allocating sh_i to server P_i.*
- $s \leftarrow \mathsf{Rec}_{\mathsf{tofp}}(\mathcal{P}, \{sh_i\}_{i\in\mathcal{P}})$*: A deterministic reconstruction algorithm. It takes as input a valid surviving server subset $\mathcal{P} \subseteq \{P_1, \ldots, P_p\}$ with cardinality $|\mathcal{P}| \geq t$, along with their corresponding shares. It outputs the reconstructed secret s.*

The scheme must satisfy two fundamental properties:

1. ***Correctness:*** *For any valid secret $s \in \mathbb{F}$ and any server subset $\mathcal{P}$ of size $|\mathcal{P}| \geq t$, the reconstruction always succeeds: $\Pr[\mathsf{Rec}_{\mathsf{tofp}}(\mathcal{P}, \{sh_i\}_{i\in\mathcal{P}}) = s] = 1$.*
2. ***Privacy:*** *For any unqualified subset $\mathcal{P}^*$ of size $|\mathcal{P}^*| < t$, the joint distribution of their shares $\{sh_i\}_{i\in\mathcal{P}^*}$ is strictly independent of the secret s, yielding zero information leakage.*

2.3. Randomized Special Combinatorial Design

To circumvent the exponential share size lower bound inherent in deterministic constructions [18], our protocol leverages a randomized special combinatorial design. Following [20], we recall its formal structural and security properties as follows.

Definition 3 (Special Combinatorial Design [20]). *Let $\lambda, p \in \mathbb{N}$, and l, r be polynomial functions of λ and p. A special combinatorial design consists of two bipartite graphs G_0 and G_1, both containing p left vertices (representing servers) and right vertices (representing pseudorandom seeds). The design must satisfy the following three properties:*

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

As illustrated in Figure 1, a concrete instantiation of this design is constructed as follows: the baseline graph G_0 is generated by independently sampling two random left vertices (with replacement) for each of its l right vertices. To construct the entire graph G_1, the baseline G_0 is augmented with r additional right vertices, each connected to exactly one randomly sampled left vertex. As demonstrated in [20], this construction efficiently satisfies all properties in Definition 3 with overwhelming probability when the parameters l and r are sufficiently large.

 doi:10.20944/preprints202605.0700.v1

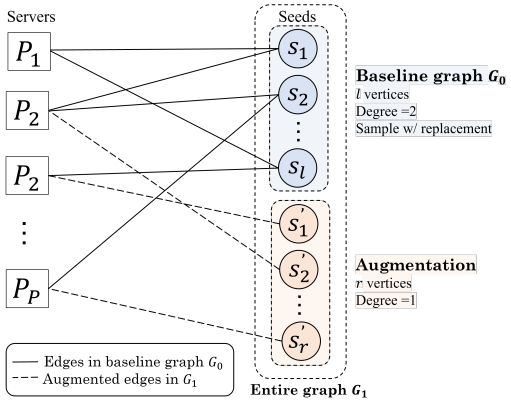

Figure 1. The randomized special combinatorial design. The baseline graph G_0 (solid edges) is utilized for sharing 0. The entire graph G_1, which encompasses both the baseline edges and the augmented edges (dashed lines), is utilized for sharing the target evaluation. We emphasize that the two random neighbors in G_0 are sampled *with replacement*.

3. Problem Formulation and Definitions

In this section, the system model and the threat model are presented. Moreover, the design goals and formal definitions of the proposed fault-tolerant DPF and PIR protocols are established.

3.1. System Model

We consider a standard multi-server PIR setting comprising a single stateless client $\mathcal{C}$ and a set of p distributed servers, denoted as $\mathcal{P} = \{P_1, P_2, \ldots, P_p\}$. As shown in Figure 2, the servers hold identical copies of a public database DB consisting of N elements, i.e., $\mathsf{DB} = (x_1, x_2, \ldots, x_N)$, where each element $x_i \in \mathbb{G}$.

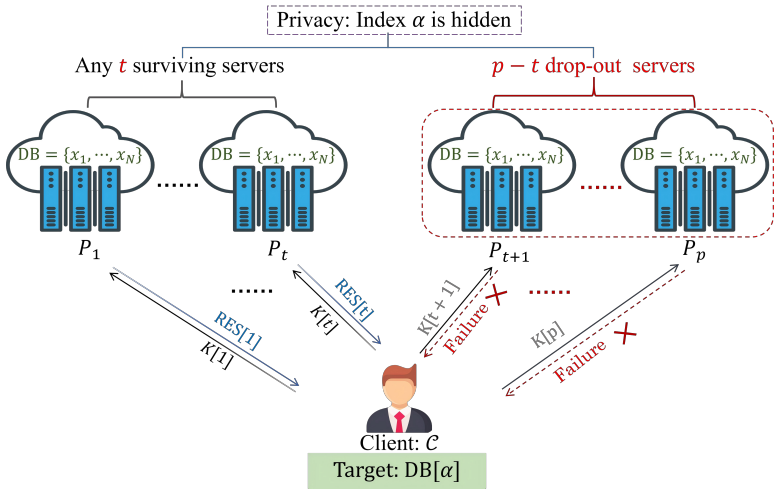

Figure 2. System Model.

When the client $\mathcal{C}$ wishes to retrieve the α-th element $\mathsf{DB}[\alpha]$ without revealing the index α to the servers, the system execution proceeds through three primary phases:

1. **Query Generation (**Query**):** The client $\mathcal{C}$ executes the qeury algorithm of the (t, p)-threshold DPF to create p keys $K[1], \ldots, K[p]$. Each $K[i]$ is transmitted to the corresponding server P_i over a secure channel.
2. **Evaluation (**Eval**):** Upon receiving the query key, each available server P_i linearly scans the database DB and locally evaluates the DPF. It computes a partial response y_i and sends it back to the client.
3. **Crash-Resilient Reconstruction (**Rec**):** Due to potential network fluctuations or node failures, $\mathcal{C}$ may only receive a subset of responses. As long as the client collects answers from any t surviving

 doi:10.20944/preprints202605.0700.v1

servers (where $t \leq p$), it applies the threshold reconstruction algorithm to recover the exact value of $\mathsf{DB}[\alpha] = x_\alpha$.

3.2. Threat Model

Our system considers a distributed architecture distributing trust across p servers. Following standard multi-party computation paradigms, we define a hybrid adversarial model $\mathcal{A}$ bounded by a threshold parameter t ($2 \leq t \leq p$), capturing both confidentiality threats and availability disruptions:

Definition 4 (Semi-Honest Adversary with Fail-Stop Faults)**.** *The capabilities of the computationally bounded adversary $\mathcal{A}$ are strictly constrained by the following two fault types:*

- ***Semi-Honest Collusion (Eavesdropping faults):*** *$\mathcal{A}$ may statically corrupt a subset of up to $t-1$ servers. These corrupted servers honestly follow the protocol specifications but attempt to infer sensitive information (e.g., the client's query index α) from their joint view, which includes their internal states, received secret shares, and transcribed messages.*
- ***Unresponsiveness (Halting faults):*** *$\mathcal{A}$ may cause up to $p-t$ servers to abruptly crash, experience severe network latency, or become entirely unresponsive at any point during the protocol execution. A halted server simply stops participating and sends no further responses to the client.*

Under this rigorous threat model, our proposed FT-PIR protocol guarantees absolute query privacy against any coalition of up to $t-1$ curious servers, while ensuring strict system availability and correctness as long as any t servers remain online to fulfill the client's request.

3.3. Design Goals

Given the aforementioned architecture and threat model, our proposed FT-PIR scheme aims to achieve the following design goals:

- **Fault Tolerance (Crash-Resilient Correctness):** Even if up to $p-t$ servers simultaneously drop out or fail to respond, the client $\mathcal{C}$ must still be able to output the correct element $\mathsf{DB}[\alpha]$ with probability 1, relying solely on the responses from the remaining t available servers.
- **Query Privacy:** The target index α must remain computationally completely hidden from any coalition of up to $t-1$ colluding servers.
- **Communication Efficiency:** The protocol ensures a strictly sublinear uplink query size and completely decouples the downlink response overhead from the database capacity N. This N-independent response size eliminates the heavy bandwidth bottlenecks of traditional robust PIR, making the scheme highly practical for massive datasets.
- **Stateless Client:** The client should require only $\mathcal{O}(1)$ local storage, maintaining no persistent state, cryptographic keys, or database metadata across consecutive queries.

3.4. Definitions of Fault-Tolerant DPF

To achieve robust resilience against server halting faults in Private Information Retrieval, we formalize the extension of the standard two-party DPF [18] [19] to a generalized threshold setting. Unlike traditional DPFs that strictly require the participation of all p servers to reconstruct the evaluation, a (t,p)-fault-tolerant distributed point function (FT-DPF) leverages a structurally redundant access policy. It guarantees correct and deterministic reconstruction from any subset of t surviving servers, seamlessly tolerating up to $p-t$ fail-stop faults. We formalize this primitive as follows:

Definition 5 (Generalized (t,p)-Fault-Tolerant DPF)**.** *A (t,p)-FT-DPF scheme specialized for the Boolean point function $f_{\alpha,1}: \{0,1\}^n \to \{0,1\}$ over the Boolean domain comprises four probabilistic polynomial-time (PPT) algorithms $\mathsf{DPF}_{tofp} = (\mathsf{Init}_{tofp}, \mathsf{Share}_{tofp}, \mathsf{Eval}_{tofp}, \mathsf{Rec}_{tofp})$ satisfying the following properties:*

- ***Syntax:***

 doi:10.20944/preprints202605.0700.v1

- $\mathbf{S} \leftarrow \mathsf{Init}_{tofp}(t, p, 1^\lambda, f_{\alpha,1})$: *A randomized setup algorithm executed by the client. It takes as input the threshold configuration (t, p), a security parameter λ, and the target Boolean point function $f_{\alpha,1}$. It outputs the foundational base key state $\mathbf{S}$.*
- $(\mathbf{K}_1, \ldots, \mathbf{K}_p) \leftarrow \mathsf{Share}_{tofp}(t, p, \mathbf{S}, \mathcal{P})$: *A recursive distribution algorithm. Taking the base state $\mathbf{S}$ and a target distribution set $\mathcal{P}$ (where initially $\mathcal{P} = [p]$), it outputs a p-tuple of customized evaluation key sets $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$, privately allocating $\mathbf{K}_i$ to server P_i.*
- $\mathbf{Y}_i \leftarrow \mathsf{Eval}_{tofp}(i, \mathbf{K}_i, x)$: *A deterministic local evaluation algorithm executed by server P_i. Given its index i, the local key set $\mathbf{K}_i$, and an evaluation point $x \in \{0,1\}^n$, it outputs a partial Boolean evaluation vector $\mathbf{Y}_i$.*
- $y \leftarrow \mathsf{Rec}_{tofp}(t, \mathcal{P}_{rec}, \{\mathbf{Y}_i\}_{i \in \mathcal{P}_{rec}})$: *A deterministic generalized reconstruction algorithm. It takes as input a subset of surviving server indices $\mathcal{P}_{rec} \subseteq [p]$ of size t, and their corresponding evaluation vectors. It executes a generalized shadow backtracking logic to output the reconstructed Boolean value $y \in \{0,1\}$.*

- ***Correctness:*** *For any query target $\alpha \in \{0,1\}^n$, any evaluation point $x \in \{0,1\}^n$, and any valid subset of surviving servers $\mathcal{P}_{rec} \subseteq [p]$ with $|\mathcal{P}_{rec}| = t$, if $\mathbf{S} \leftarrow \mathsf{Init}_{tofp}(t, p, 1^\lambda, f_{\alpha,1})$ and $(\mathbf{K}_1, \ldots, \mathbf{K}_p) \leftarrow \mathsf{Share}_{tofp}(t, p, \mathbf{S}, [p])$, then it strictly holds that:*

$$\Pr\Big[\mathsf{Rec}_{tofp}\big(t, \mathcal{P}_{rec}, \{\mathsf{Eval}_{tofp}(i, \mathbf{K}_i, x)\}_{i \in \mathcal{P}_{rec}}\big) = f_{\alpha,1}(x)\Big] = 1 \tag{1}$$

- ***Privacy:*** *The scheme guarantees $(t-1)$-computational privacy. For any unauthorized subset of colluding server indices $\mathcal{I} \subset [p]$ with $|\mathcal{I}| < t$, there exists a PPT simulator Sim such that the joint view of their key sets $\{\mathbf{K}_i\}_{i \in \mathcal{I}}$ generated by Share_{tofp} is computationally indistinguishable from a null distribution synthesized by $\mathsf{Sim}(1^\lambda, \mathcal{I})$ without knowledge of the target index α.*

3.5. Definitions of Fault-Tolerant PIR

Building upon the generalized FT-DPF, we construct a multi-server PIR protocol that enables a client to privately retrieve a target record from a public database DB replicated across a cluster of p servers. The architecture guarantees both strict query confidentiality against semi-honest coalitions and high retrieval availability, seamlessly tolerating up to $p - t$ server halting faults during the online evaluation phase.

Definition 6 (Generalized (t, p)-Fault-Tolerant PIR)**.** *A (t, p)-FT-PIR scheme operating over a database $\mathsf{DB} = (\mathsf{DB}[1], \ldots, \mathsf{DB}[N])$ comprises three probabilistic polynomial-time (PPT) algorithms $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Eval}, \mathsf{Rec})$ satisfying the following properties:*

- ***Syntax:***
  - $(\mathbf{K}_1, \ldots, \mathbf{K}_p) \leftarrow \mathsf{PIR.Query}(t, p, 1^\lambda, N, \alpha)$: *A randomized algorithm executed by the client. It takes as input the threshold configuration (t, p), a security parameter λ, the database capacity N, and the target retrieval index $\alpha \in [N]$. It generates and outputs a p-tuple of customized query key sets $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$.*
  - $\mathbf{RES}_j \leftarrow \mathsf{PIR.Eval}(j, \mathbf{K}_j, \mathsf{DB})$: *A deterministic algorithm executed locally by server P_j. It takes as input the server's index j, its designated query key set $\mathbf{K}_j$, and the entire database replica DB. It computes the bitwise XOR inner product across the database and outputs an aggregated response vector $\mathbf{RES}_j$.*
  - $y \leftarrow \mathsf{PIR.Rec}(t, \mathcal{P}_{rec}, \{\mathbf{RES}_i\}_{i \in \mathcal{P}_{rec}})$: *A deterministic algorithm executed by the client. It takes as input the threshold parameter t, a subset of surviving server indices $\mathcal{P}_{rec} \subseteq [p]$ (where $|\mathcal{P}_{rec}| = t$), and their corresponding response vectors. It reconstructs and outputs the target database record $y = \mathsf{DB}[\alpha]$.*

 doi:10.20944/preprints202605.0700.v1

- ***Correctness:*** *For any valid database* DB *of size* N*, any target index* $\alpha \in [N]$*, and any responsive subset of surviving servers* $\mathcal{P}_{rec} \subseteq [p]$ *with* $|\mathcal{P}_{rec}| = t$*, if the client computes* $(\mathbf{K}_1, \ldots, \mathbf{K}_p) \leftarrow \mathsf{PIR.Query}(t, p, 1^\lambda, N, \alpha)$*, then the reconstruction algorithm deterministically recovers the target record. Formally, it holds with probability 1 that:*

$$\mathsf{PIR.Rec}\Big(t, \mathcal{P}_{rec}, \{\mathsf{PIR.Eval}(i, \mathbf{K}_i, \mathsf{DB})\}_{i \in \mathcal{P}_{rec}}\Big) = \mathsf{DB}[\alpha] \tag{2}$$

- ***Privacy:*** *The protocol guarantees* $(t-1)$*-computational query privacy against semi-honest adversaries. For any unauthorized coalition of server indices* $\mathcal{I} \subset [p]$ *bounded by* $|\mathcal{I}| < t$*, there exists a PPT simulator* Sim *such that the joint distribution of the query key sets* $\{\mathbf{K}_i\}_{i \in \mathcal{I}}$ *generated by* PIR.Query *is computationally indistinguishable from a null distribution synthesized by* $\mathsf{Sim}(1^\lambda, N, \mathcal{I})$ *entirely without knowledge of the client's target index* α*.*

4. Hierarchical Construction of Threshold DPFs

This section presents our (t, p)-threshold fault-tolerant DPF. Unlike Shamir-based schemes that incur heavy finite field arithmetic, we achieve optimal fault tolerance via a novel *recursive conflict resolution* mechanism.

Our hierarchical construction proceeds in three progressive steps: Section 4.1 establishes the base (p, p) DPF scheme; Section 4.2 extends it to a $(2, p)$ framework via incremental extension; and Section 4.3 generalizes it to the arbitrary (t, p) setting, formalizing the recursive auxiliary key generation to circumvent the exponential share blowup inherent in deterministic FSS [18] architectures.

4.1. Base Construction: (p, p)-Threshold DPF

In this section, we propose an efficient (p, p)-threshold DPF construction based on the Boolean point function $f_{\alpha,1} : \{0,1\}^n \to \{0,1\}$. This construction is specifically tailored for typical Private Information Retrieval (PIR) scenarios requiring strictly binary outputs. Building upon the foundational paradigm and the polynomial-size random combinatorial design established by [20], our scheme introduces targeted architectural optimizations. By carefully adapting their graph-based distribution mechanism for the Boolean domain, we effectively reduce communication overhead and key size while maintaining optimal computational efficiency. The formal description is presented in Algorithm 1, and the core underlying mechanisms are elucidated as follows:

 doi:10.20944/preprints202605.0700.v1

Algorithm 1 (p, p)-Threshold DPF

Let n be the input length, set $u = 2^{n/2}$, and $\ell = 2^{p-1}$.
Let $\mathbf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{u}$ be a pseudorandom generator.
Let $\mathbf{e}_{\delta}$ denote the u-dimensional unit vector with a 1 at the δ-th position and 0 elsewhere.

DPF.Share(γ, p, u) :

for $g = 1$ to u **do**
  for $i = 1$ to p **do**
    $\mathbf{S}_i^g \leftarrow \emptyset$.
  end for
  if $g \neq \gamma$ **then**
    Sample a baseline graph G_0 according to Definition 3, denoted by $G_{0,g}$.
    for each right vertex $b \in [\ell]$ in $G_{0,g}$ **do**
      Sample seed $s_{0,g}^b \xleftarrow{\$} \{0,1\}^{\lambda}$.
      for $i = 1$ to p **do**
        if the b-th right vertex is connected to the i-th left vertex **then**
          $\mathbf{S}_i^g \leftarrow \mathbf{S}_i^g \cup \{s_{0,g}^b\}$.
        end if
      end for
    end for
  else
    Sample an entire graph G_1 according to Definition 3, denoted by $G_{1,g}$.
    for each right vertex $b \in [\ell]$ in $G_{1,g}$ **do**
      Sample seed $s_{1,g}^b \xleftarrow{\$} \{0,1\}^{\lambda}$.
      for $i = 1$ to p **do**
        if the b-th right vertex is connected to the i-th left vertex **then**
          $\mathbf{S}_i^g \leftarrow \mathbf{S}_i^g \cup \{s_{1,g}^b\}$.
        end if
      end for
    end for
  end if
end for
return $(\mathbf{S}_1^1, \ldots, \mathbf{S}_1^u), \ldots, (\mathbf{S}_p^1, \ldots, \mathbf{S}_p^u)$.

DPF.Gen$_{base}(1^{\lambda}, f_{\alpha,1})$:

Regard α as a pair $\alpha = (\gamma, \delta)$, where $\gamma, \delta \in [u]$.
Compute $(\mathbf{S}_i^1, \ldots, \mathbf{S}_i^u)_{i \in [p]} \leftarrow$ DPF.Share(γ, p, u).
Initialize $\mathbf{cw} \leftarrow \mathbf{e}_{\delta}$.
for $i = 1$ to p **do**
  Parse $\mathbf{S}_i^{\gamma}$ as a sequence of ℓ_i seeds $(s_i^{\gamma,1}, \ldots, s_i^{\gamma,\ell_i})$.
  $\mathbf{cw} \leftarrow \mathbf{cw} \oplus \left(\bigoplus_{b=1}^{\ell_i} \mathbf{PRG}(s_i^{\gamma,b})\right)$.
end for
for $i = 1$ to p **do**
  $k_i \leftarrow (\mathbf{cw} \parallel \mathbf{S}_i^1 \parallel \cdots \parallel \mathbf{S}_i^u)$.
end for
return $(k_1, \ldots, k_p)$.

DPF.Eval$_{base}(i, k_i, x)$:

Regard x as a pair $x = (\gamma', \delta')$, where $\gamma', \delta' \in [u]$.
Parse $k_i = (\mathbf{cw} \parallel \mathbf{S}_i^1 \parallel \cdots \parallel \mathbf{S}_i^u)$.
Parse $\mathbf{S}_i^{\gamma'}$ as seeds $s_1, \ldots, s_{\ell_i}$.
Compute $d_i \leftarrow (\ell_i \bmod 2) \cdot \mathbf{cw} \oplus \left(\bigoplus_{b=1}^{\ell_i} \mathbf{PRG}(s_b)\right)$.
return $y_i = d_i[\delta']$.

DPF.Rec$_{base}(y_1, \ldots, y_p)$:

return $y = \bigoplus_{i=1}^{p} y_i$ (where $y = f_{\alpha,1}(x)$).

Matrix Mapping and Coordinate Transformation

To significantly reduce storage and computational complexity, we embed the original N-dimensional output vector (where $N = 2^n$) into a $u \times u$ two-dimensional matrix, where $u = 2^{n/2}$. Under this mapping, the secret index α is precisely transformed into matrix coordinates $(\gamma, \delta) \in [u] \times [u]$,

 doi:10.20944/preprints202605.0700.v1

representing the target row and column, respectively. This architectural shift reduces the linear search across an N-dimensional space to localized operations on u-dimensional vectors, achieving a square-root compression of computational costs.

Seed Generation via Graph Distributions

Leveraging the randomized combinatorial design, we utilize two distinct bipartite graph distributions to generate pseudorandom seeds, thereby masking the target row γ. For the target row γ, the seed sets are sampled using the entire graph G_1, which mathematically guarantees an odd degree for specific valid subsets. For all non-target rows ($\gamma' \neq \gamma$), the baseline graph G_0 is utilized, strictly enforcing an even-degree constraint.

Boolean-Tailored PRG and Coordinate Hiding

In standard multiparty DPF constructions [20], the pseudorandom generator must stretch seeds to a domain of $\{0,1\}^{|\mathbb{F}|\cdot u}$ to accommodate general finite fields. Recognizing that PIR payload selection fundamentally requires only Boolean outputs ($|\mathbb{F}| = 1$), we elegantly specialize the $\mathbf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{u}$ such that its expansion length is strictly aligned with the matrix row width u. This targeted optimization allows servers to perform bitwise XOR operations directly on row vectors, eliminating the $|\mathbb{F}|$-fold expansion overhead. Furthermore, a correction word $\mathbf{cw} \in \{0,1\}^{u}$ is constructed to securely hide the specific column coordinate δ.

Evaluation and Structural Parity Logic

During the online evaluation phase, upon receiving a query point $x = (\gamma', \delta')$, each server evaluates the row vector. Crucially, instead of relying on explicit auxiliary control bits to activate the correction word, we embed the activation logic intrinsically within the graph's structural parity. Each server computes its share d_i by scaling the correction word with the parity of its seed count $(\ell_i \bmod 2)$:

$$d_i = (\ell_i \bmod 2) \cdot \mathbf{cw} \oplus \left(\bigoplus_{b=1}^{\ell_i} \mathbf{PRG}(s_b) \right) \tag{3}$$

The reconstruction logic strictly depends on this parity constraint:

- **Case 1: Row Index Match ($\gamma' = \gamma$).** The entire graph G_1 guarantees that the total number of surviving seeds is odd (i.e., $\sum \ell_i \equiv 1 \pmod 2$). The correction word $\mathbf{cw}$ is thus perfectly preserved after the XOR accumulation, recovering the u-dimensional unit vector $\mathbf{e}_\delta$. If the evaluated column matches ($\delta' = \delta$), the output yields 1; otherwise, 0.
- **Case 2: Row Index Mismatch ($\gamma' \neq \gamma$).** The baseline graph G_0 guarantees an even number of seed overlaps (i.e., $\sum \ell_i \equiv 0 \pmod 2$). Both the random components and the correction word are self-canceled entirely during the XOR aggregation, consistently yielding a zero vector $\mathbf{0}^u$. Thus, the output is fixed at 0 regardless of δ'.

Security Analysis. The privacy of our construction is fundamentally anchored in the simulation-based framework. Because any colluding subset of $p-1$ servers lacks at least one critical seed, the aggregated PRG sequences remain statistically indistinguishable from a uniform distribution. The core obfuscation relies on the statistical proximity of G_0 and G_1: from a localized adversarial perspective, the even-degree vertices in G_0 statistically mask the targeted odd-degree vertices planted in G_1. When the graph scale parameter ℓ satisfies a sufficient polynomial bound, the statistical distance is heavily bounded, robustly safeguarding the secret index α.

Efficiency Evaluation. Compared to the state-of-the-art framework by [20], our protocol achieves substantial performance gains in both communication overhead and key storage:

- *Dimension Compression:* By specializing in the Boolean domain, we directly compress the output length of the **PRG** and the size of **cw** from $O(|\mathbb{F}| \cdot \sqrt{N})$ to exactly $O(\sqrt{N})$ bits.

 doi:10.20944/preprints202605.0700.v1

- *Algorithmic Key Streamlining:* We completely eradicate the necessity of the $n \times m$ auxiliary random control matrix (e.g., the t_g^i bits) conventionally required to orchestrate row activation. By transferring this responsibility to the intrinsic $(\ell_i \bmod 2)$ parity multiplier, we mathematically drop the u-bit control matrix from the server's key state. This significantly reduces the constant factor associated with the total key size, strictly bounding it to $|k_i| = O(\sqrt{N} \cdot \mathrm{poly}(p) \cdot \lambda)$.
- *Optimal Online Complexity:* During the online phase, the strict computational independence between matrix rows guarantees an $O(\sqrt{N})$ evaluation time. Furthermore, the response share size achieves optimal constant complexity $|Y_i| = O(1)$ bit, making this architecture exceptionally well-suited for the real-time response requirements of large-scale PIR deployments.

4.2. Pairwise Construction: $(2, p)$*-Threshold DPF*

In this section, we construct an efficient $(2, p)$-threshold DPF scheme to address the single-point failure vulnerability inherent in strict full-participation PIR systems. This pairwise construction establishes a resilient architecture where the evaluation can be seamlessly reconstructed as long as any two servers survive. Building upon the base (p, p)-DPF presented in Section 4.1 and the 2-of-p incremental extension mechanism proposed by [22], our scheme bypasses the exponential share blowup of traditional combinatorial designs. By precisely assigning *patch keys* to resolve node conflicts during incremental expansion, the total number of shares per server reaches the theoretical minimum of $O(\log p)$. The formal description is detailed in Algorithm 2, with the core mechanisms delineated below:

Initialization and Key Extension

Instead of computing a monolithic sharing scheme, the dealer invokes the base generation algorithm Gen_{base} exactly $m = \lceil \log_2 p \rceil$ times. This pre-generates m fully independent base $(2,2)$-DPF key pairs, denoted as sequences $\mathbf{S}_1$ and $\mathbf{S}_2$. These orthogonal base keys serve as the fundamental patch components for the subsequent incremental allocation, effectively decoupling the redundancy across different recursive levels.

Incremental Share Allocation

The pre-generated base keys are distributed among the p servers using a dynamic, tree-like inheritance mechanism. The distribution begins by establishing a baseline $(2,2)$ association, assigning $\mathbf{S}_1[1]$ to P_1 and $\mathbf{S}_2[1]$ to P_2. For any newly appended server P_i ($i \geq 3$), the scheme computes its *conflict index* (i.e., its shadow ancestor) via $c = i - 2^{\lceil \log_2 i \rceil - 1}$. To inherit the functional capabilities of the network, P_i first seamlessly duplicates the entire key set of its shadow ancestor P_c. Crucially, to resolve the cryptographic conflict (as P_i and P_c now hold identical shares and cannot reconstruct a secret together), the dealer injects a fresh set of patch keys at the current recursive level $idx = \lceil \log_2 i \rceil$: $\mathbf{S}_1[idx]$ is appended to P_c, while $\mathbf{S}_2[idx]$ is assigned to P_i. This patching precisely establishes a new pairwise association, culminating in p tailored key sets $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$.

 doi:10.20944/preprints202605.0700.v1

Algorithm 2 $(2,p)$-Threshold DPF via Incremental Extension

Primitives: We instantiate the base $(2,2)$-threshold DPF from Algorithm 1, denoting its algorithms as $(\mathsf{Gen}_{base}, \mathsf{Eval}_{base}, \mathsf{Rec}_{base})$. Let $m = \lceil \log_2 p \rceil$.

$\mathsf{DPF.Init}_{2ofp}(p, 1^\lambda, f_{\alpha,1})$:

for $g = 1$ to m **do** ▷ Generate m independent base DPF instances
  $(k_1^{(g)}, k_2^{(g)}) \leftarrow \mathsf{Gen}_{base}(1^\lambda, f_{\alpha,1})$.
end for
Define sequences $\mathbf{S}_1 \leftarrow (k_1^{(1)}, \ldots, k_1^{(m)})$ and $\mathbf{S}_2 \leftarrow (k_2^{(1)}, \ldots, k_2^{(m)})$.
return $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2)$.

$\mathsf{DPF.Share}_{2ofp}(p, \mathbf{S})$:

Initialize key sets $\mathbf{K}_i \leftarrow \emptyset$ for $i \in [p]$.
$\mathbf{K}_1 \leftarrow \{\mathbf{S}_1[1]\}$, and $\mathbf{K}_2 \leftarrow \{\mathbf{S}_2[1]\}$.
for $i = 3$ to p **do**
  Compute the conflict index: $c \leftarrow i - 2^{\lceil \log_2 i \rceil - 1}$.
  Inherit keys: $\mathbf{K}_i \leftarrow \mathbf{K}_c$.
  Append new base keys to resolve the conflict:
    $\mathbf{K}_c \leftarrow \mathbf{K}_c \cup \{\mathbf{S}_1[\lceil \log_2 i \rceil]\}$.
    $\mathbf{K}_i \leftarrow \mathbf{K}_i \cup \{\mathbf{S}_2[\lceil \log_2 i \rceil]\}$.
end for
return $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$.

$\mathsf{DPF.Eval}_{2ofp}(j, \mathbf{K}_j, x)$:

Parse $\mathbf{K}_j$ as an ordered sequence of keys $(k_{j,1}, k_{j,2}, \ldots, k_{j,|\mathbf{K}_j|})$.
for $r = 1$ to $|\mathbf{K}_j|$ **do**
  Compute base share: $Y_{j,r} \leftarrow \mathsf{Eval}_{base}(j, k_{j,r}, x)$.
end for
return the share vector $\mathbf{Y}_j = (Y_{j,1}, Y_{j,2}, \ldots, Y_{j,|\mathbf{K}_j|})$.

$\mathsf{DPF.Rec}_{2ofp}(\mathcal{P}_{rec}, \{\mathbf{Y}_i\}_{i \in \mathcal{P}_{rec}})$:

Let $\mathcal{P}_{rec} = \{j_1, j_2\}$ be the two surviving indices, where $j_1 < j_2$.
Initialize evaluation output $\mathbf{y} \leftarrow \mathbf{0}$.
while true do
  if $j_2 == 2$ **then**
    $y \leftarrow \mathbf{Y}_{j_1}[1] \oplus \mathbf{Y}_{j_2}[1]$.
    break
  else
    Compute conflict index: $c \leftarrow j_2 - 2^{\lceil \log_2(j_2) \rceil - 1}$.
    if $c == j_1$ **then**
      Let $idx = \lceil \log_2(j_2) \rceil$.
      $\mathbf{y} \leftarrow \mathbf{Y}_{j_1}[idx] \oplus \mathbf{Y}_{j_2}[idx]$.
      break
    else
      Update $j_2 \leftarrow c$, and ensure $\{j_1, j_2\}$ remains sorted such that $j_1 < j_2$. ▷ perform shadow backtracking to the conflict ancestor
  end if
end if
end while
return y. ▷ reconstructed Boolean evaluation $y = f_{\alpha,1}(x)$

Evaluation and Shadow Backtracking Reconstruction

During the online phase, upon receiving a query x, each server P_j evaluates all base keys within its local set $\mathbf{K}_j$ in parallel, yielding a compact evaluation vector $\mathbf{Y}_j \in \{0,1\}^{|\mathbf{K}_j|}$.

The fault-tolerant reconstruction logic is executed by the client upon receiving vectors from any two surviving servers, P_{j_1} and P_{j_2} (assuming $j_1 < j_2$). If P_{j_1} is the direct shadow ancestor of P_{j_2} (i.e., $j_1 = c$), the conflict was resolved precisely at level $\lceil \log_2(j_2) \rceil$. The client simply extracts the corresponding bits from $\mathbf{Y}_{j_1}$ and $\mathbf{Y}_{j_2}$ and computes the XOR sum to reconstruct $f_{\alpha,1}(x)$. If they are not a direct conflict pair, the algorithm executes a *shadow backtracking* mechanism: it recursively replaces the larger index j_2 with its shadow ancestor c until the structural conflict is found. This elegant traversal guarantees successful reconstruction while avoiding complex interpolation operations.

 doi:10.20944/preprints202605.0700.v1

Security Analysis. The security of this pairwise construction is jointly guaranteed by the privacy of the underlying base DPF and the threshold properties of the incremental extension. The base $(2,2)$-DPF provides strict cryptographic privacy, ensuring that no single server can deduce the target index α. Furthermore, the 2-of-p access structure algorithmically bounds the view of any single compromised node. Even though servers hold overlapping key sets due to shadow inheritance, the patch key injection strictly ensures that recovering the full functional evaluation requires two orthogonal shares from a valid $(2,2)$ base pair. Consequently, any single colluding node observes only computationally indistinguishable pseudorandomness, guaranteeing 1-privacy against semi-honest adversaries.

Efficiency Evaluation. This hierarchical patching mechanism achieves exceptional performance metrics, successfully resolving single-point failures without incurring the catastrophic overhead of traditional RSS configurations:

- *Logarithmic Key Storage:* By leveraging conflict-driven duplication and targeted patch injection rather than storing all combinatorial possibilities, the maximum number of base keys held by any server is strictly bounded by $\lceil \log_2 p \rceil$. Consequently, the total key size is scaling gracefully as $|K_i| = O\left(\sqrt{N} \cdot \text{poly}(p) \cdot \lambda \cdot \log p\right)$.
- *Logarithmic Communication:* The response vector $\mathbf{Y}_j$ consists solely of the evaluated Boolean bits corresponding to the local base keys. Thus, the online communication complexity is bounded by $O(\log p)$ bits. This guarantees that as the fault-tolerant network scales, the bandwidth requirement grows only logarithmically, maintaining optimal practicality for low-latency PIR.

4.3. Generalized Construction: (t,p)-Threshold DPF

To accommodate diverse fault-tolerance requirements, we now generalize the pairwise construction to an arbitrary (t,p)-threshold scenario. By integrating the base (t,t)-DPF primitives with the t-of-p recursive patching mechanism proposed by [22], our architecture effectively circumvents the combinatorial explosion inherent in traditional threshold access structures. It systematically reduces high-order threshold conflicts into lower-order $(t-2)$ sub-problems through recursive dimensional degradation, guaranteeing strict $(t-1)$-privacy while maintaining polynomial scalability. The total number of keys in the key set $K[j]$ is determined by the following recursive function

$$F_{\text{maxl}}(t,p) = \begin{cases} \lceil \log_2 p \rceil & \text{if } t = 2 \\ p-2 & \text{if } t = 3 \\ 1 & \text{if } t = p \\ F_{\text{maxl}}(t,p-1) + F_{\text{maxl}}(t-2,p-2) & \text{otherwise} \end{cases} \tag{4}$$

A visual intuition of this architecture, illustrating the incremental expansion and fault-tolerant reconstruction for a concrete $(4,6)$ setting, is provided in Figure 3. The formal procedures are divided into two phases, detailed in Algorithms 3 and 4, with the core design paradigm delineated below:

 doi:10.20944/preprints202605.0700.v1

Algorithm 3 Generalized (t, p)-Threshold DPF: Setup and Sharing

Primitives: We instantiate the base (t, t)-DPF algorithms as $(\mathsf{Gen}^t_{base}, \mathsf{Eval}^t_{base}, \mathsf{Rec}^t_{base})$, and import the pairwise algorithms $\mathsf{DPF}_{2ofp} = (\mathsf{Share}_{2ofp}, \mathsf{Eval}_{2ofp}, \mathsf{Rec}_{2ofp})$ from Algorithm 2. Let $\mathcal{F}_{\max}(t, n)$ denote the maximum sharing level function defined in (4).

$\mathsf{DPF.Init}_{tofp}(t, p, 1^\lambda, f_{\alpha,1})$:

Let $M = p - t + 1$. Invoke $\mathsf{Gen}^t_{base}(1^\lambda, f_{\alpha,1})$ exactly M times to generate M groups of t-party base keys: $(k^1_i, k^2_i, \ldots, k^t_i)$ for $i \in [M]$. ▷ Generate baseline key sets for the (t, p) expansion
Group the components orthogonally: define sequences $\mathbf{S}_j \leftarrow (k^j_1, k^j_2, \ldots, k^j_M)$ for all $j \in [t]$.
return $\mathbf{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_t)$.

$\mathsf{DPF.Share}_{tofp}(t, p, \mathbf{S}, \mathcal{P}_{sub})$:

Initialize key sets $\mathbf{K}_i \leftarrow \emptyset$ for $i \in \mathcal{P}_{sub}$.
if $t == 2$ **then**
  return $\mathsf{Share}_{2ofp}(p, \mathbf{S})$.
else
  Assign foundational shares: add $\mathbf{S}_j[1]$ to $\mathbf{K}_j$ for all $1 \le j \le t$.
  for $i = t + 1$ to p **do**
    Compute conflict index: $c \leftarrow i - t$.
    Inherit existing structure: $\mathbf{K}_i \leftarrow \mathbf{K}_c$.
    Resolve primary conflict: $\mathbf{K}_c \leftarrow \mathbf{K}_c \cup \{\mathbf{S}_1[c + 1]\}$ and $\mathbf{K}_i \leftarrow \mathbf{K}_i \cup \{\mathbf{S}_2[c + 1]\}$.
    Define active recursion subset: $\mathcal{P}'_{sub} \leftarrow \{m \in \mathcal{P}_{sub} \mid m < i \text{ and } m \neq c\}$.
    for $b = 1$ to $t - 2$ **do** ▷ recursive degradation for the remaining $(t - 2)$ components
      Parse $\mathbf{S}_{b+2}[c + 1]$ into $\mathbf{cw}_b$ and PRG seeds.
      Apply combinatorial permutation to re-partition the seeds for $(t - 2)$ setting.
      Reassemble into c new sub-keys: $\{k^b_1, \ldots, k^b_c\}$.
    end for
    Let $\mathbf{S}^* = (\mathbf{S}^*_1, \ldots, \mathbf{S}^*_{t-2})$, where $\mathbf{S}^*_b \leftarrow (k^{b+2}_{c+1}, k^b_1, \ldots, k^b_c)$.
    Recursively embed sub-structures: execute $\mathsf{Share}_{tofp}(t - 2, i - 2, \mathbf{S}^*, \mathcal{P}'_{sub})$.
  end for
end if
return $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$.

Algorithm 4 Generalized (t, p)-Threshold DPF: Evaluation and Reconstruction

Note: This algorithm follows the key distribution established in Algorithm 3. $\mathcal{P}_{rec}$ is a sorted set of t surviving server indices.

$\mathsf{DPF.Eval}_{tofp}(j, \mathbf{K}_j, x)$:

$\mathbf{Y}_j \leftarrow \mathsf{Eval}_{2ofp}(j, \mathbf{K}_j, x)$. ▷ local evaluation is structurally identical to the pairwise setting
return $\mathbf{Y}_j$.

$\mathsf{DPF.Rec}_{tofp}(t, \mathcal{P}_{rec}, \{\mathbf{Y}_i\}_{i \in \mathcal{P}_{rec}})$:

Initialize evaluation output $y \leftarrow 0$.
if $t == 2$ **then**
  return $\mathsf{Rec}_{2ofp}(\mathcal{P}_{rec}, \{\mathbf{Y}_i\}_{i \in \mathcal{P}_{rec}})$.
else
  while true do
    Let j_{max} be the maximum index in $\mathcal{P}_{rec}$.
    if $j_{max} == t$ **then** ▷ evaluate directly upon reaching the minimal subset $\{1, \ldots, t\}$
      $y \leftarrow \bigoplus_{i=1}^t \mathbf{Y}_{\mathcal{P}_{rec}[i]}[1]$.
      break
    else
      Compute structural conflict index: $c \leftarrow j_{max} - t$.
      if $c \in \mathcal{P}_{rec}$ **then** ▷ handle direct conflict and reduce to the $(t - 2)$ case.
        Compute target activation level: $lvl \leftarrow \mathcal{F}_{max}(t, j_{max} - 1) + 1$.
        Extract primary resolution: $y_{conf} \leftarrow \mathbf{Y}_c[lvl] \oplus \mathbf{Y}_{j_{max}}[lvl]$.
        Extract secondary resolution recursively: $y_{rest} \leftarrow \mathsf{Rec}_{tofp}(t - 2, \mathcal{P}_{rec} \setminus \{c, j_{max}\}, \{\mathbf{Y}_i\})$.
        $y \leftarrow y_{conf} \oplus y_{rest}$.
        break
      else
        Replace j_{max} with c in $\mathcal{P}_{rec}$, maintaining sorted ascending order. ▷ shadow backtracking via ancestor replacement
      end if
    end if
  end while
end if
return y.

Initialization and Component Grouping (Algorithm 3)

Rather than relying on a single monolithic generation, the algorithm pre-computes $M = p - t + 1$ independent base (t, t)-DPF key sets: $\{(k^1_i, \ldots, k^t_i)\}_{i=1}^M$. These foundational blocks are orthogonally

 doi:10.20944/preprints202605.0700.v1

grouped into t sequences $\mathbf{S}_1, \ldots, \mathbf{S}_t$. Compared to the pairwise scheme, this high-dimensional initialization natively embeds the necessary baseline entropy to support t-party reconstruction before any fault-tolerant expansion begins.

Recursive Share Allocation (Algorithm 3)

The distribution leverages a dynamic inheritance and patching mechanism to output p customized key sets $\mathbf{K}_1, \ldots, \mathbf{K}_p$. For any incrementally added server P_i ($i > t$), the allocation proceeds in two critical phases: 1) *Conflict Inheritance and Resolution:* P_i first inherits the complete structural state of its shadow ancestor P_c (where $c = i - t$). Subsequently, fresh patch components from $\mathbf{S}_1$ and $\mathbf{S}_2$ are assigned to P_c and P_i respectively, resolving the primary cryptographic conflict at that level. 2) *Dimensional Degradation:* For the remaining $(t-2)$ components, the scheme applies a polynomial-size combinatorial permutation to re-partition the PRG seeds. By recursively invoking the allocation function, the architecture elegantly collapses the remaining high-order sharing requirements into a manageable $(t-2)$-of-$(i-2)$ sub-problem. As visualized in the left panel of Figure 3, this mechanism scales a base $(4,4)$ structure to a $(4,6)$ setting by elegantly leveraging underlying $(2,3)$ DPF instances (highlighted in the blue and green boxes) to generate the auxiliary patch keys.

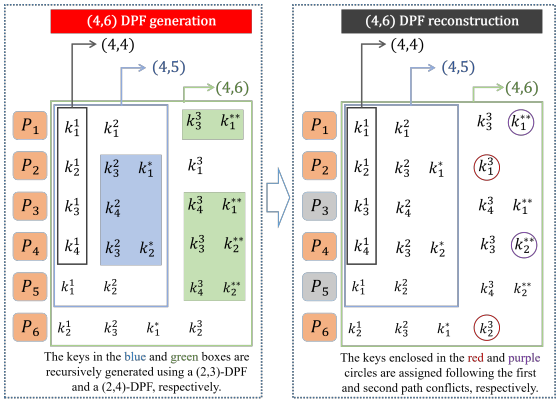

Figure 3. An illustrative example of the proposed $(4,6)$-threshold FT-DPF scheme. **Left (Generation):** The key distribution scales from a base $(4,4)$ structure to a $(4,6)$ configuration via the recursive patching mechanism. Specifically, auxiliary keys in the blue and green boxes are generated using underlying $(2,3)$-DPF and $(2,4)$-DPF instances, respectively. **Right (Reconstruction):** The fault-tolerant evaluation phase. In the presence of halting faults at P_3 and P_5 (greyed out), the surviving servers seamlessly recover the target by employing designated patch keys (enclosed in red and purple circles) to resolve sequential path conflicts.

Evaluation and Generalized Backtracking (Algorithm 4)

During the online phase, each server P_j evaluates all sub-keys within its local set $\mathbf{K}_j$ in parallel. Crucially, the reconstruction logic dynamically adapts to the surviving subset of t servers via a generalized *shadow backtracking* mechanism. Bounded by the maximum sharing level function $\mathcal{F}_{max}$, the algorithm identifies the maximum active index j_{max}. If a direct conflict pair exists in the surviving set, the client extracts the specifically activated shares to resolve it, and recursively evaluates the remaining $(t-2)$ subset. Otherwise, it iteratively replaces j_{max} with its shadow ancestor c until a valid topological match is found. Finally, the linear superposition property of the Boolean domain ensures that all non-target masks cancel out, precisely recovering $f_{\alpha,1}(x)$. As demonstrated in the right panel of Figure 3, even when encountering halting faults (e.g., servers P_3 and P_5), the surviving nodes seamlessly traverse the topology and utilize the designated patch keys (red and purple circles) to resolve sequential conflicts and reconstruct the target.

Security Analysis. The security of this generalized scheme is fundamentally anchored in the (t,t)-computational privacy of the base DPF and the restricted access structure of the recursive mechanism. According to the foundational proofs (see Section 2), any coalition of $t-1$ servers inevitably lacks at least one critical base share required to trigger the closed-loop XOR cancellation. Furthermore, the recursive combinatorial permutation ensures that the aggregated PRG streams observed by any

 doi:10.20944/preprints202605.0700.v1

unauthorized $(t-1)$ subset remain statistically indistinguishable from a uniform random distribution. Consequently, the scheme strictly guarantees $(t-1)$-privacy against semi-honest adversaries.

Efficiency Evaluation. Our generalized protocol successfully bypasses the catastrophic $O(\binom{p}{t})$ exponential explosion of share counts typically found in traditional combinatorial or replicated secret sharing frameworks.

- *Storage Overhead:* By employing the recursive degradation structure, the minimal redundancy coverage is strictly bounded. The overall key size held by each server is optimized to $|K_j| = O(\sqrt{N} \cdot \text{poly}(p,t) \cdot \lambda)$, where the $\text{poly}(p,t)$ factor reflects the theoretical minimum algebraic overhead required for recursive distribution.
- *Communication Overhead:* Driven by the localized evaluation capabilities, the online response dimension $|Y_j|$ depends exclusively on the logical path combinations and is completely independent of the database size N. The online bandwidth consumption is constrained to $O(\text{poly}(p,t))$ bits per server, ensuring near-constant extreme efficiency even as the cluster scale and threshold parameters increase.

5. The Proposed Fault-Tolerant PIR Scheme

5.1. Detailed Protocol Description

Based on the generalized (t,p)-threshold DPF architecture, we design and implement a Private Information Retrieval (PIR) protocol that efficiently supports customizable (t,p)-fault tolerance. In this system model, we assume a cluster comprising p servers, $P_1,\ldots,P_p$, where each server stores a full replica of the database DB. The database contains N records, formulated as $\mathsf{DB} = (\mathsf{DB}[1],\ldots,\mathsf{DB}[N])$, with each block $\mathsf{DB}[i] \in \{0,1\}^\ell$.

By integrating the recursive dimensional degradation mechanism proposed by [22], the protocol ensures robust data retrieval even if up to $p-t$ servers experience halting faults. The execution flow, formally encapsulated in Algorithm 5, is divided into three core phases:

Query Phase

When a client intends to retrieve the record at index α, they implicitly define a Boolean point function $f_{\alpha,1} : [N] \to \{0,1\}$. The client invokes the PIR.Query algorithm, which internally calls the Init_{tofp} and Share_{tofp} procedures (Algorithm 3) to generate p customized sets of query keys $\mathbf{K}_1,\ldots,\mathbf{K}_p$. Through dynamic inheritance and recursive patching, these key sets inherently embed the topological redundancy required for threshold recovery. Finally, the client securely distributes each key set $\mathbf{K}_j$ to the corresponding server P_j.

Evaluation Phase

Upon receiving the query set $\mathbf{K}_j$, server P_j executes a localized, vectorized computation across the entire database without any inter-server interaction. For every record index $i \in [N]$, the server computes the DPF share vector $\mathbf{Y}_j^{(i)} \leftarrow \mathsf{Eval}_{tofp}(j,\mathbf{K}_j,i)$. The server then accumulates the target payload by computing the XOR inner product between the database blocks and the DPF shares: $\mathbf{RES}_j = \bigoplus_{i=1}^{N} \mathsf{DB}[i] \cdot \mathbf{Y}_j^{(i)}$. This aggregated response vector $\mathbf{RES}_j$ is returned to the client.

Reconstruction Phase

Leveraging the t-threshold property, the client awaits response vectors $\{\mathbf{RES}_i\}_{i\in\mathcal{P}_{rec}}$ from any viable subset of t surviving servers (i.e., $|\mathcal{P}_{rec}| = t$). The client then invokes PIR.Rec, which executes the generalized shadow backtracking mechanism (Algorithm 4) to precisely locate and extract the active logical levels. The topological masks are deterministically canceled out via XOR aggregations, recovering the exact target record $\mathsf{DB}[\alpha]$.

 doi:10.20944/preprints202605.0700.v1

Algorithm 5 Fault-Tolerant PIR Based on Generalized (t, p)-Threshold DPF

Primitives: We utilize the generalized DPF algorithms $\mathsf{DPF}_{tofp} = (\mathsf{Init}_{tofp}, \mathsf{Share}_{tofp}, \mathsf{Eval}_{tofp}, \mathsf{Rec}_{tofp})$ established in Algorithms 3 and 4. The database contains N records, formulated as $\mathsf{DB} = (\mathsf{DB}[1], \ldots, \mathsf{DB}[N])$.

Input: Client's target index $\alpha \in [N]$.
Output: The retrieved record $\mathsf{DB}[\alpha]$.

$\mathsf{PIR.Query}(t, p, 1^\lambda, N, \alpha)$:

1: Generate base keys: $\mathbf{S} \leftarrow \mathsf{Init}_{tofp}(t, p, 1^\lambda, f_{\alpha,1})$.
2: Distribute via recursive patching: $(\mathbf{K}_1, \ldots, \mathbf{K}_p) \leftarrow \mathsf{Share}_{tofp}(t, p, \mathbf{S}, [p])$.
3: **return** query sets $(\mathbf{K}_1, \ldots, \mathbf{K}_p)$ to servers $P_1, \ldots, P_p$, respectively.

$\mathsf{PIR.Eval}(j, \mathbf{K}_j, \mathsf{DB})$:

1: Initialize response vector $\mathbf{RES}_j \leftarrow \mathbf{0}$.
2: **for** $i = 1$ to N **do**
3: Extract share vector for current index: $\mathbf{Y}_j^{(i)} \leftarrow \mathsf{Eval}_{tofp}(j, \mathbf{K}_j, i)$.
4: Accumulate: $\mathbf{RES}_j \leftarrow \mathbf{RES}_j \oplus \left(\mathsf{DB}[i] \cdot \mathbf{Y}_j^{(i)}\right)$.
5: **end for**
6: **return** $\mathbf{RES}_j$.

$\mathsf{PIR.Rec}(t, \mathcal{P}_{rec}, \{\mathbf{RES}_i\}_{i \in \mathcal{P}_{rec}})$:

1: Execute generalized backtracking: $y \leftarrow \mathsf{Rec}_{tofp}(t, \mathcal{P}_{rec}, \{\mathbf{RES}_i\}_{i \in \mathcal{P}_{rec}})$. ▷ the client reconstructs the target record using responses from any t surviving servers $\mathcal{P}_{rec}$.
2: **return** y. ▷ $y = \mathsf{DB}[\alpha]$

5.2. Correctness Analysis

The correctness of this fault-tolerant PIR protocol is mathematically anchored in the linear superposition property of the Boolean DPF and the structural parity logic of the recursive reconstruction.

Because the generalized reconstruction function Rec_{tofp} operates purely via bitwise XOR aggregations, it constitutes a linear operator over the Galois field $\mathrm{GF}(2)$. Therefore, the reconstruction over the accumulated response vectors mathematically commutes with the evaluation over individual indices. The client's final computation yields:

$$y = \mathsf{Rec}_{tofp}\left(t, \mathcal{P}_{rec}, \{\mathbf{RES}_j\}_{j \in \mathcal{P}_{rec}}\right) \tag{5}$$

Substituting the server-side evaluation expansion:

$$y = \bigoplus_{i=1}^{N} \mathsf{DB}[i] \cdot \left(\mathsf{Rec}_{tofp}\left(t, \mathcal{P}_{rec}, \{\mathbf{Y}_j^{(i)}\}_{j \in \mathcal{P}_{rec}}\right)\right) \tag{6}$$

According to the correctness of the underlying (t, p)-DPF, the inner reconstruction evaluates exactly to the point function $f_{\alpha,1}(i)$. Hence:

$$y = \bigoplus_{i=1}^{N} \mathsf{DB}[i] \cdot f_{\alpha,1}(i) = (\mathsf{DB}[\alpha] \cdot 1) \oplus \left(\bigoplus_{i \neq \alpha} \mathsf{DB}[i] \cdot 0\right) = \mathsf{DB}[\alpha] \tag{7}$$

This strictly proves that the protocol flawlessly recovers the target record.

5.3. Protocol Security

The proposed protocol satisfies the following dual security guarantees:
(t, p)-Fault Tolerance (Availability): The protocol establishes topological redundancy through the recursive degradation architecture. Because Share_{tofp} ensures that any subset of t surviving sub-key sets can seamlessly trigger the required shadow backtracking and complete the closed-loop reconstruction path, the system gracefully tolerates up to $p - t$ server failures (e.g., node crashes, network partitions, or offline status) without compromising retrieval availability.

 doi:10.20944/preprints202605.0700.v1

$(t-1)$-Collusion-Resistant Privacy (Confidentiality): The query privacy strictly inherits the access structure bounds of the generalized scheme. Even if up to $t-1$ semi-honest servers collude and share their views, their combined key components represent an algorithmic proper subset of the global entropy pool. Lacking the critical complementary patch keys required to satisfy the topological parity, their aggregated PRG streams remain computationally indistinguishable from a uniform random distribution. Consequently, the user's query index α is perfectly hidden from any unauthorized coalition.

5.4. Efficiency Evaluation

By systematically avoiding the combinatorial explosion of traditional threshold mechanisms, our protocol optimizes computational and network resource consumption while delivering high-order fault tolerance:

Computational Efficiency: The protocol completely eschews heavy cryptographic operations such as modular exponentiations or bilinear pairings. The server-side workload is dominated by highly parallelizable, hardware-friendly bitwise XOR operations and PRG expansions, achieving an optimal $O(N)$ scanning cost. The client-side reconstruction simply involves constant-time XOR backtracking, making it highly competitive for ultra-low latency requirements.

Storage Overhead: By deploying the recursive fractal construction, the scheme mathematically collapses the exponential key duplication. The storage complexity of the query keys $\mathbf{K}_j$ held by each server is strictly bounded to $O\left(\sqrt{N} \cdot \text{poly}(p,t) \cdot \lambda\right)$. This polynomial degradation significantly alleviates the memory footprint on the servers, breaking the scalability bottlenecks of standard RSS-based PIR.

Online Communication Overhead: The paramount advantage of this DPF-based approach is its sublinear download cost. The online response vector $\mathbf{RES}_j$ is strictly independent of the database capacity N. The communication complexity scales solely with the payload size and the topological path logic, yielding an optimal download overhead of $\mathcal{O}(\ell \cdot \text{poly}(p,t))$ bits. This ensures that even as the server cluster scales and the threshold rises, the bandwidth consumption maintains extremely low-order growth, maximizing real-world practicality.

6. Related Work

6.1. Single-Server PIR

Single-server PIR fundamentally operates under the computational privacy setting. Existing literature typically optimizes its performance bottleneck through two distinct technical routes: relying on homomorphic encryption to compress communication, or utilizing hint-based preprocessing to bypass heavy public-key computations.

6.1.1. HE-Based Schemes

Scheme [8] proposed an unkeyed doubly-efficient PIR (DEPIR) protocol based on the standard ring-LWE assumption, establishing a robust theoretical foundation for sublinear online query complexity. However, extending this construction to RAM-FHE necessitates an additional circular security assumption, and the protocol inherently incurs prohibitive ciphertext storage overhead. To mitigate the offline communication bottleneck and enhance server throughput, scheme [6] introduced "silent preprocessing" via LWE-to-RLWE homomorphic packing techniques and number theoretic transform (NTT) acceleration. Nevertheless, to successfully compress the downstream response, this approach inevitably inflates the client's online query size. To address the circuit depth and computational bottlenecks encountered during the preprocessing phase, scheme [7] utilized homomorphic Thorp permutations combined with a learning with rounding (LWR)-based PRG to construct a constant-depth preprocessing circuit. Although this architecture reduces offline processing time and achieves sublinear offline bandwidth, it imposes a strict bound on the maximum number of queries and fails to eliminate the substantial constant-factor overhead inherent in homomorphic evaluations. Consequently,

 doi:10.20944/preprints202605.0700.v1

the prohibitive cryptographic computational costs remain the primary impediment to the practical deployment of HE-based PIR frameworks.

6.1.2. Hint-Based Preprocessing Schemes

Distinct from HE-based approaches, another line of research leverages lightweight symmetric primitives via client-side preprocessing. Scheme [9] introduced puncturable pseudorandom sets to achieve sublinear online communication and computation under standard one-way function assumptions. Building on this, schemes [10] and [11] proposed hint-based constructions using pseudorandom functions and programmable pseudorandom sets, systematically reducing the online communication complexity to $\mathcal{O}(\sqrt{N})$ and $\mathcal{O}(N^{1/4})$, respectively. Furthermore, to optimize the parameter configurations for stateful clients, scheme [12] designed a dynamic data relocation mechanism. This approach precisely balances the client's local storage overhead (S) and the server's online computation time (T), achieving a strict space-time trade-off of $S \cdot T = \mathcal{O}(N)$. While these preprocessing protocols significantly improve online efficiency without relying on public-key operations, they fundamentally require an $\mathcal{O}(N)$ offline database scan to fetch hints. Additionally, the bounded nature of these pre-fetched hints necessitates costly full-scale resets upon depletion, which limits their applicability in continuous high-frequency retrieval scenarios.

6.2. Multi-Server PIR

To bypass the inherent computational and storage bottlenecks of single-server models, multi-server PIR architectures distribute queries across multiple nodes. By amortizing the workload, this architecture significantly improves retrieval efficiency. Existing multi-server protocols primarily evolve along two distinct technical trajectories: information-theoretic constructions based on algebraic coding, and lightweight cryptographic protocols utilizing FSS.

6.2.1. Algebraic and Coding-Based Schemes

Operating under the information-theoretic privacy setting, these schemes leverage polynomials and error-correcting codes to mask client queries against bounded server collusion [13]. Early foundational works established strict baselines: scheme [14] bounded 2-server communication at $\mathcal{O}(N^{1/3})$, while scheme [15] achieved optimal Byzantine robustness via algebraic decoding algorithms. Subsequent research successfully breached these initial limits. To optimize bandwidth, scheme [16] utilized matching vector codes to compress communication to a sub-polynomial level. Furthermore, to mitigate the $\mathcal{O}(N)$ server computation barrier, scheme [17] proposed a multi-server doubly efficient PIR paradigm, employing preprocessing to amortize online computation to a sublinear level. Despite these advancements, such schemes rely heavily on expensive algebraic operations over large finite fields and stateful preprocessing mechanisms. Consequently, they remain fundamentally constrained in practical deployments, struggling to simultaneously deliver logarithmic communication, stateless client operations, and efficient fault tolerance.

6.2.2. FSS-Based Schemes

Traditional PIR schemes typically incur prohibitively high communication overhead. To alleviate this bottleneck, schemes [18] and [19] introduced the DPF and FSS frameworks. By transmitting key shares generated via pseudorandom generators (PRGs), these primitives compress the client's upload communication overhead from linear to sublinear. Building upon this foundation, subsequent work [19] further optimized the tree-based key expansion mechanism in the two-server architecture, strictly bounding the request communication complexity to $\mathcal{O}(\log N)$ while maintaining a response communication of $\mathcal{O}(1)$. Such minimalist constructions established the theoretical optimum for two-party PIR communication efficiency. However, their applicability is restricted to the two-server model, rendering them vulnerable to more sophisticated collusion attacks.

To accommodate distributed environments with heightened security requirements, researchers have endeavored to extend FSS to multi-party ($p \geq 3$) architectures. However, early multi-party

 doi:10.20944/preprints202605.0700.v1

DPF protocols relying on deterministic constructions suffered from severe bandwidth inflation; their request overhead reached $\mathcal{O}(\sqrt{N} \cdot 2^p)$, which grows exponentially with the number of servers [18]. To circumvent this theoretical barrier, scheme [20] proposed an optimization scheme based on a randomized special combinatorial design, successfully compressing the key size of multi-party DPFs to $\mathcal{O}(p^3\sqrt{N})$. Despite this significant improvement in query efficiency, standard FSS-based PIR protocols heavily rely on the responses from all p servers during the reconstruction phase. Consequently, a single point of failure or network latency can result in the complete failure of the retrieval process. To enhance system availability, scheme [21] introduced a fault-tolerant PIR protocol that allows the client to recover the target data using only a subset of responses from active nodes. However, its underlying mechanism necessitates returning multiple verification blocks to the client during the evaluation phase, which severely erodes the inherent $\mathcal{O}(1)$ lightweight response advantage of standard DPF protocols.

7. Conclusions and Future Work

In this paper, we proposed a novel (t, p)-fault-tolerant PIR (FT-PIR) protocol to address the inherent single-point failure vulnerability in standard FSS-based architectures. Grounded in a newly designed generalized fault-tolerant distributed point function (FT-DPF), our scheme utilizes a hierarchical recursive patching mechanism to enable deterministic reconstruction from any t surviving servers. Recognizing that practical deployments treat the server cluster size as a small constant relative to massive datasets ($N \gg p \geq t$), our architecture completely decouples the downlink response overhead from N. In addition to the strict $\mathcal{O}(poly(t, p) \cdot \log p)$ communication bound, our protocol facilitates highly efficient client-side reconstruction by employing straightforward XOR aggregations rather than computationally expensive algebraic interpolations. Ultimately, the proposed protocol achieves optimal stateless client efficiency and robust fail-stop fault tolerance while strictly guaranteeing $(t-1)$-computational privacy under the semi-honest model.

Future research will focus on two critical extensions. First, we aim to achieve full Byzantine fault tolerance to ensure query correctness against malicious adversaries that return arbitrarily manipulated responses. Second, we will extend the recursive patching mechanism to more expressive FSS primitives, such as distributed comparison functions (DCF), to efficiently support privacy-preserving range queries and complex analytics over distributed databases.

References

1. Chor, B.; Kushilevitz, E.; Goldreich, O.; Sudan, M. Private information retrieval. *Journal of the ACM (JACM)* **1998**, *45*, 965–981.
2. Demmler, D.; Rindal, P.; Rosulek, M.; Trieu, N. PIR-PSI: Scaling Private Contact Discovery. *Proceedings on Privacy Enhancing Technologies* **2018**, *4*, 159–178.
3. Henzinger, A.; Dauterman, E.; Corrigan-Gibbs, H.; Zeldovich, N. Private web search with tiptoe. In Proceedings of the Proceedings of the 29th symposium on operating systems principles, 2023, pp. 396–416.
4. Corrigan-Gibbs, H.; Boneh, D.; Mazières, D. Riposte: An anonymous messaging system handling millions of users. In Proceedings of the 2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 321–338.
5. Dauterman, E.; Feng, E.; Luo, E.; Popa, R.A.; Stoica, I. {DORY}: An encrypted search system with distributed trust. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 2020, pp. 1101–1119.
6. Menon, S.J.; Wu, D.J. {YPIR}:{High-Throughput}{Single-Server}{PIR} with silent preprocessing. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 5985–6002.
7. Fisch, B.; Lazzaretti, A.; Liu, Z.; Papamanthou, C. ThorPIR: single server PIR via homomorphic thorp shuffles. In Proceedings of the Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 1448–1462.
8. Lin, W.K.; Mook, E.; Wichs, D. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Proceedings of the Proceedings of the 55th Annual ACM Symposium on Theory of Computing, 2023, pp. 595–608.

 doi:10.20944/preprints202605.0700.v1

9. Shi, E.; Aqeel, W.; Chandrasekaran, B.; Maggs, B. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In Proceedings of the Annual International Cryptology Conference. Springer, 2021, pp. 641–669.
10. Zhou, M.; Park, A.; Zheng, W.; Shi, E. Piano: extremely simple, single-server PIR with sublinear server computation. In Proceedings of the 2024 IEEE symposium on security and privacy (SP). IEEE, 2024, pp. 4296–4314.
11. Ghoshal, A.; Zhou, M.; Shi, E. Efficient pre-processing PIR without public-key cryptography. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2024, pp. 210–240.
12. Wang, Z.; Ren, L. Single-server client preprocessing PIR with tight space-time trade-off. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2025, pp. 94–122.
13. Eriguchi, R.; Nuida, K. A Survey on Private Information Retrieval: Information-Theoretic Constructions and Error-Correction Techniques. In *Mathematical Foundations for Post-Quantum Cryptography: Crypto-Math CREST*; Springer, 2025; pp. 475–494.
14. Woodruff, D.; Yekhanin, S. A geometric approach to information-theoretic private information retrieval. In Proceedings of the 20th Annual IEEE Conference on Computational Complexity (CCC'05). IEEE, 2005, pp. 275–284.
15. Devet, C.; Goldberg, I.; Heninger, N. Optimally robust private information retrieval. In Proceedings of the 21st USENIX Security Symposium (USENIX Security 12), 2012, pp. 269–283.
16. Dvir, Z.; Gopi, S. 2-server PIR with subpolynomial communication. *Journal of the ACM (JACM)* **2016**, *63*, 1–15.
17. Lazzaretti, A.; Liu, Z.; Fisch, B.; Miao, P.; Papamanthou, C. Multi-server Doubly Efficient PIR in the Classical Model and Beyond. In Proceedings of the Theory of Cryptography Conference. Springer, 2025, pp. 611–641.
18. Boyle, E.; Gilboa, N.; Ishai, Y. Function secret sharing. In Proceedings of the Annual international conference on the theory and applications of cryptographic techniques. Springer, 2015, pp. 337–367.
19. Boyle, E.; Gilboa, N.; Ishai, Y. Function secret sharing: Improvements and extensions. In Proceedings of the Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 1292–1303.
20. Goel, A.; Wang, M.; Wang, Z. Multiparty Distributed Point Functions. In Proceedings of the Annual International Cryptology Conference. Springer, 2025, pp. 140–173.
21. Park, A.; Leong, T.; Maturana, F.; Zheng, W.; Rashmi, K. Communication-efficient, fault tolerant PIR over erasure coded storage. In Proceedings of the 2024 IEEE symposium on security and privacy (SP). IEEE, 2024, pp. 4331–4347.
22. Guo, J.; Shuai, M.; Wang, Q.; Li, W.; Lin, J. Replicated additive secret sharing with the optimized number of shares. In Proceedings of the International Conference on Security and Privacy in Communication Systems. Springer, 2022, pp. 371–389.