Quantized Private Information Retrieval

A project submitted in fulfillment of the requirements

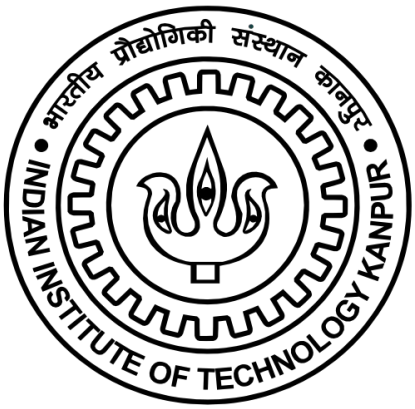
for the degree of Summer Internship

By

Rohit Mahesh

2540032

VELLORE INSTITUTE OF TECHNOLOGY



Under guidance of

Dr. Adithya Vadapalli

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CERTIFICATE

It is certified that the work contained in this project entitled "**Quantized Private Information Retrieval**" by Rohit Mahesh has been carried out under my supervision at the Indian Institute of Technology Kanpur.

Dr. Adithya Vadapalli
Department of Computer Science Engineering
Indian Institute of Technology,Kanpur

Acknowledgment

It is my proud privilege and duty to acknowledge the kind help and guidance received from several people in preparation for this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation, and guidance.

I am highly grateful to the Privacy Lab, Department of Computer Science Engineering, IIT Kanpur. The constant guidance and encouragement received from Dr. Adithya Vadapalli, Department of Computer Science Engineering, IIT Kanpur, have been of great help in carrying out the project work and are acknowledged with reverential thanks. I would like to express a deep sense of gratitude and thanks profusely to Dr. Sonu Sharma and all my labmates, for their invaluable help during my Summer Internship. My project, "**Quantized Private Information Retrieval**" has greatly benefited from their collective expertise and unwavering support. Their insightful discussions, technical assistance, and encouragement have been instrumental in the successful completion of this project.

Finally, I would like to express my sincere gratitude to the SURGE office at IIT Kanpur for providing me with this valuable opportunity to pursue research under the Summer Undergraduate Research Grant for Excellence program. Their initiative has offered a remarkable platform for academic growth, and I am thankful for the support and resources extended throughout the course of this project

Abstract

We design, analyze, and implement a framework for executing Private Information Retrieval (PIR) queries under a fixed computational budget, wherein the level of privacy is quantized along a continuum from no privacy to the highest achievable privacy under the given resource constraints. Our model introduces a partitioning mechanism that divides the database into a set of intervals. Accessing higher intervals incurs greater computational overhead but provides stronger privacy by increasing the server's uncertainty about the queried index.We achieve this by optimizing a parameterized weight function over the partitioned database, where each interval corresponds to a different degree of privacy. Tuning the weights gives users control over the trade-off between efficiency and privacy.This quantized PIR framework builds upon standard PIR by introducing a tunable, user-driven privacy model.Our construction leverages Distributed Point Functions (DPFs) to encode queries across multiple servers securely. It supports flexible retrieval of queries across multiple layers of ambiguity, making it particularly suitable for deployment in resource-constrained environments where privacy-performance trade-offs must be explicitly managed.

Keywords : Private Information Retrieval,Computational Budget,Distributed Point Functions.

1. Introduction

Private Information Retrieval (PIR) allows a user to retrieve an item from a database without revealing which item is being fetched. The problem was introduced by Chor et al.[1] in a multi-server, information-theoretic setting, where the database is replicated across multiple servers and privacy is guaranteed as long as the servers do not all collude. This model avoids cryptographic assumptions, instead relying on the presence of at least one honest server. In contrast, single-server PIR requires computational assumptions to achieve privacy. A key breakthrough by Kushilevitz and Ostrovsky [2] showed that homomorphic encryption could be leveraged to bypass the linear communication lower bound, enabling sublinear communication at the cost of heavy server-side computation. Since then, a range of protocols have been proposed to optimize communication or computation—though often not both simultaneously. On the information-

theoretic front, techniques using algebraic structures and coding theory (locally decodable codes) [3] have further reduced query sizes in multi-server PIR. However, most of these approaches do not attempt to quantify the level of privacy offered. They operate under a binary model either complete privacy is ensured, or none at all with no indication of how much privacy is actually being provided. In modern, resource-constrained environments, such an perspective is often impractical and lacks the flexibility needed to balance privacy with efficiency.To bridge this gap, we build upon secret-sharing-based PIR schemes, particularly those using Distributed Point Functions (DPFs), which naturally support multi-server PIR by distributing a user's query across servers in a privacy-preserving manner. This line of work motivates our Quantized PIR framework, which utilizes secret sharing to split the query index across servers and allows users to trade computational budgets for varying degrees of privacy. By quantizing privacy levels into discrete intervals and carefully tuning their associated computational cost, our scheme introduces a structured, tunable model that better reflects real-world constraints and user preferences.

2. Background

2.1 Secret-Sharing

Secret sharing is a foundational cryptographic primitive that enables a secret value to be divided into multiple fragments, known as *shares*, which are then distributed among several parties. The fundamental security guarantee is that only a qualified subset of these parties can collaborate to reconstruct the original secret, while any subset of insufficient size learns nothing about it. This mechanism is pivotal in secure multiparty computation (MPC), private information retrieval (PIR), and threshold cryptography [4-5].

One of the most practical and efficient variants of secret sharing is *additive secret sharing*. In this scheme, a secret value x over a finite field $\mathbb{F}_N$ is split into n shares $x_0, x_1, \ldots, x_{n-1}$ such that:

$$x = x_0 + x_1 + \cdots + x_{n-1} \bmod N$$

Each share appears uniformly random and is individually independent of the original secret. Unless all n shares are combined, no information about x is revealed, offering information-theoretic security.

2.2 Secret-Shared 1-Hot Vectors

Secret-shared 1-hot vectors form a foundational building block in Private Information Retrieval (PIR) systems and related privacy-preserving data access protocols. These vectors allow users to retrieve records from a distributed database without revealing which record is being requested, thus ensuring access pattern privacy [10].

A 1-hot vector $\mathbf{e}_i \in \{0,1\}^n$ encodes an index i such that:

$$\mathbf{e}_i[j] = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases}$$

This vector is then split using additive secret sharing into multiple components:

$$\mathbf{e}_i = \mathbf{e}_i^{(0)} + \mathbf{e}_i^{(1)} + \cdots + \mathbf{e}_i^{(t)} \bmod N$$

Each share $\mathbf{e}_i^{(k)}$ is sent to a server P_k, such that the original index remains hidden from every individual server.

2.3 Distributed Point Functions (DPFs)

Distributed Point Functions (DPFs) are a concise method for sharing a standard basis vector (1-hot vector) among multiple parties. First introduced by Gilboa and Ishai **[6]**, later improved by Boyle, Gilboa, and Ishai [7-8], DPFs enable privacy-preserving data access with compact representations.

2.3.1 Point Functions

A point function is defined as a function that evaluates to zero everywhere except at a target point i^*, where it evaluates to a target value M.

Definition 1 (Point Function)

A point function $p_{i^*,M}: [0, n) \to \{0,1\}^*$ satisfies:

$$p_{i^*,M}(i) = \begin{cases} M, & \text{if } i = i^* \\ 0, & \text{otherwise} \end{cases}$$

Point functions can be represented as a binary tree, where the target value appears at the target index leaf

2.3.2 Distributed Point Functions

DPFs **secret-share** a point function across m parties such that no coalition of fewer than t parties can learn the target point or value.

We focus on (2,2)-DPFs, sharing a point function among two parties securely.

Definition 2 ((2,2)-DPF)

A (2,2)-DPF consists of a pair of PPT algorithms (GEN, EVAL) that:

- **Inputs:**
  - Security parameter $\lambda \in \mathbb{N}$
  - Target point i^*
  - Target value M
- **Correctness:**
  If $(k_0, k_1) \leftarrow \mathrm{GEN}(1^\lambda, i^*, M)$, then for all $i \in [0, n)$,

$$\mathrm{EVAL}(k_0, i) + \mathrm{EVAL}(k_1, i) = \begin{cases} M, & \text{if } i = i^* \\ 0, & \text{otherwise} \end{cases}$$

- **Simulatability:**
  There exists a PPT simulator S such that for bit $b \in \{0, 1\}$, the distributions:

$$\{S(1^\lambda, b)\}_{\lambda \in \mathbb{N}} \quad \text{and} \quad \{k_b : (k_0, k_1) \leftarrow \mathrm{GEN}(1^\lambda, i^*, M)\}_{\lambda \in \mathbb{N}}$$

  are computationally indistinguishable.

The outputs k_b are called (2,2)-DPF keys.

2.3.3 DPF Construction

The most compact DPF construction uses a length-doubling PRG (denoted $G_{2\times}$) to build Goldreich-Goldwasser-Micali style PRFs **[9]**. Denote:

- Left half of $G_{2\times}(s)$ as $G_L(s)$
- Right half as $G_R(s)$.

Note:

$$G_L(s_0) = G_L(s_1) \quad \text{and} \quad G_R(s_0) = G_R(s_1) \quad \Leftrightarrow \quad s_0 = s_1$$

- ***Initialize Seeds:***

  Choose two random seeds:

$$s_0^{()}, \quad s_1^{()}$$

- ***Recursive Tree Construction:***

Use the length-doubling PRG ($G_{2\times}$) recursively to build two binary trees, which are equal everywhere except along the path to the target leaf i^*.At the target leaf, the node pair XORs to the target value M.

- ***Correction Words & Flag Bits:***

To correct the random trees into the required structure:

- Use correction words $cw^{(j)}$ at each tree level:
  - First Level**:**

$$cw^{(1)} \leftarrow G_L\left(s_0^{()}\right) \oplus G_L\left(s_1^{()}\right)$$

  - Second Level**:**

$$cw^{(2)} \leftarrow G_R\left(s_0^{(1)}\right) \oplus G_R\left(s_1^{(1)}\right)$$

  - Third Level**:**

$$cw^{(3)} \leftarrow G_L\left(s_0^{(10)}\right) \oplus G_L\left(s_1^{(10)}\right)$$

- Use flag bits for each node to control where each correction word is applied during the recursive expansion, ensuring only the path to the target leaf differs between the two trees.

- ***Final Correction Word***

At the final step, a final correction word ensures the leaf at the target index equals the target value:

$$\mathcal{F} \leftarrow M \oplus v_0\left[i^*\right] \oplus v_1\left[i^*\right]$$

so that across the two trees:

$$v_0[i] \oplus v_1[i] = \begin{cases} M, & \text{if } i = i^* \\ 0, & \text{otherwise} \end{cases}$$

3. Security Model

3.1 Definition

A quantized ℓ-server IR protocol (S, Q, A, R) with servers $S_1, \ldots, S_\ell$ implements the quantized interval-based privacy setting if it satisfies the following conditions:

3.1.1 Correctness

For every index $i \in I_j$:

If $\left(\Gamma, D^{(1)}, \ldots, D^{(\ell)}\right) \leftarrow S(D, S_1, \ldots, S_\ell),\ (Q_1, \ldots, Q_\ell) \leftarrow Q(i)$, and $A_j \leftarrow A\left(D^{(j)}, Q_j\right)$

for each $j = 1, \ldots, \ell$,

then

$$\Pr\left[R(A_1, \ldots, A_\ell) = D[i]\right] = 1\,.$$

3.1.2 Privacy

Let X be a random variable denoting the index of the record requested.

For every pair of indices $i, i' \in I_j$ in the same interval, the query distributions satisfy

$$\Pr[Q \mid X = i] = \Pr[Q \mid X = i']\,.$$

The quantized privacy level of interval I_j is defined as

$$P_j := \log_2\left(j \cdot \left|I_j\right|\right),$$

where j captures the interval depth, and $\left|I_j\right|$ reflects the ambiguity width within the interval.

This definition captures the number of equally likely retrieval candidates within an interval, reflecting the server's uncertainty about the true target record.

3.1.3 Non-triviality

For every constant $c > 0$, there exists a positive integer N such that for all $n > N$ and for every batch of q indices $\left(i_1, \ldots, i_q\right) \in [n]^q$:

If $\left(\Gamma, D^{(1)}, \ldots, D^{(\ell)}\right) \leftarrow S\left(D, S_1, \ldots, S_\ell\right),\ \left(Q_1, \ldots, Q_\ell\right) \leftarrow Q\left(i_1, \ldots, i_q\right)$, and $A_j \leftarrow A\left(D^{(j)}, Q_j\right)$

for each $j = 1, \ldots, \ell$, then the total communication cost satisfies

$$\left|\Gamma\right| + \left|Q_1\right| + \cdots + \left|Q_\ell\right| + \left|A_1\right| + \cdots + \left|A_\ell\right| < c \cdot \left|D\right| .$$

Note: Interval preference profile

The client defines an interval preference profile by selecting non-negative weights

$$a_j \in \mathbb{R}_{\geq 0}$$

for each interval I_j.

A linear program of the form

$$\max_{\left|I_1\right|,\ldots,\left|I_k\right|} \sum_{j=1}^{k} a_j \cdot \left|I_j\right|$$

subject to

$$\sum_{j=1}^{k} \left|I_j\right| = n, \quad \sum_{j=1}^{k} j \cdot \left|I_j\right| \leq \lambda$$

determines the allocation of records across intervals.This supports a graded notion of privacy, allowing the client to allocate resources to achieve different degrees of privacy from low to high based on the application's privacy requirements and available computational budget.

4. Construction

We describe the construction using two-server DPF-based PIR, supporting graded privacy via interval quantization and LP-based interval allocation.

- Let the database be:

  $$D = \left[D[0], D[1], \ldots, D[n-1]\right],$$

  where $n \in \mathbb{N}$ is arbitrary and application-dependent.

- Two non-colluding servers S_0, S_1 each hold a copy of the database D.

- The client aims to privately retrieve $D[i]$ for an index $i \in [0, n-1]$ while preserving quantized interval-based privacy.

4.1 Interval Partitioning

- The client specifies:
  - The number of intervals $k \in \mathbb{N}$.
  - A computational budget $\lambda \in \mathbb{N}$.
  - Weights $\alpha_1, \ldots, \alpha_k \in \mathbb{R}_{\geq 0}$ indicating privacy preference for each interval.
- The client solves the following linear program:

$$\max_{|I_1|,\ldots,|I_k|} \sum_{j=1}^{k} \alpha_j \cdot |I_j|$$

subject to:

$$\sum_{j=1}^{k} |I_j| = n, \quad \sum_{j=1}^{k} j \cdot |I_j| \leq \lambda$$

where $|I_j|$ denotes the size of interval I_j.

- The solution yields interval boundaries:

$$I_j = [s_j, e_j],$$

where:

$$s_j = \sum_{r=1}^{j-1} |I_r|, \quad e_j = s_j + |I_j| - 1 .$$

4.2 Target Index Selection

The client selects an index $i \in [0, n-1]$ to retrieve.Using:

$$j^* = \text{find\_interval}\left(i, \{I_j\}\right)$$

the client identifies the interval I_{j*} containing i.

Define:

- $s = s_{j*}$ (start of the interval),
- $e = e_{j*}$ (end of the interval),
- $w = \left|I_{j*}\right| = e - s + 1$ (width of the interval),
- $N =$ round_up_to_pow2(w).

4.3 DPF-Based Query Generation

The offset within the interval is:

$$o = i - s\,.$$

The client initializes:

- Scalar $\beta = 1$.
- PRF method selection (e.g., AES-based, denoted prf_method $= 0$).

The client generates a DPF key pair:

$$\left(k_0, k_1\right) = \text{GenerateSeedsAndCodewordsLog}\left(o, \beta, N, \text{rng}, \text{prf\_method}\right)$$

and flattens them for network transmission:

$$\text{sf}_0 = \text{Flatten}\left(k_0\right), \quad \text{sf}_1 = \text{Flatten}\left(k_1\right).$$

4.4. Server-Side Evaluation

Each server S_b ($b \in \{0, 1\}$):

- Receives:
    - The flattened DPF key sf_b.
    - The starting offset s.

- The domain size N.

For $j = 0, 1, \ldots, N-1$:

- Computes:

$$v_j = \text{EvaluateFlat}\left(\text{sf}_b, j, \text{prf\_method}\right)$$

- Aggregates:

$$\text{sum}_b \mathrel{+}= v_j \cdot D\left[s + j\right].$$

The server returns:

$$r_b = \text{sum}_b \quad \text{to the client.}$$

The client computes:

$$r = r_0 - r_1$$

to recover the target:

$$r = D[i].$$

4.5. Privacy Quantification

For the retrieved interval I_{j^*}:

$$P_{j^*} = \log_2\left(\left(j^* + 1\right) \cdot \left|I_{j^*}\right|\right)$$

quantifies the server's uncertainty about the exact queried index under interval quantization.

- A higher $\left|I_{j^*}\right|$ and higher j^* yield greater privacy.
- The client can adjust k, λ, α_j to trade off privacy vs. bandwidth and computation.

4.6 Properties of Protocol

- Correctness**:** Ensured by the correctness of DPF constructions:

$$\Pr\left[r = D[i]\right] = 1 .$$

- Privacy**:** Ensured within the interval I_{j^*}:

$$\Pr\left[Q \mid X = i\right] = \Pr\left[Q \mid X = i'\right] \text{ for all } i, i' \in I_{j^*}.$$

- Efficiency**:** The use of DPF reduces bandwidth significantly compared to naive PIR, and quantization allows graded privacy at lower computational cost.

5. Results and Discussion

Through implementing our proposed construction, we identified key interactions between the number of intervals ($\mathtt{k}$), computational budget (λ), calculated privacy (P_j).These parameters directly shape the privacy-performance trade-off by controlling how privacy is distributed across the database. Our experiments highlight how adjusting these factors impacts retrieval efficiency and privacy levels, highlighting the framework's responsiveness to trade-offs between privacy and efficiency

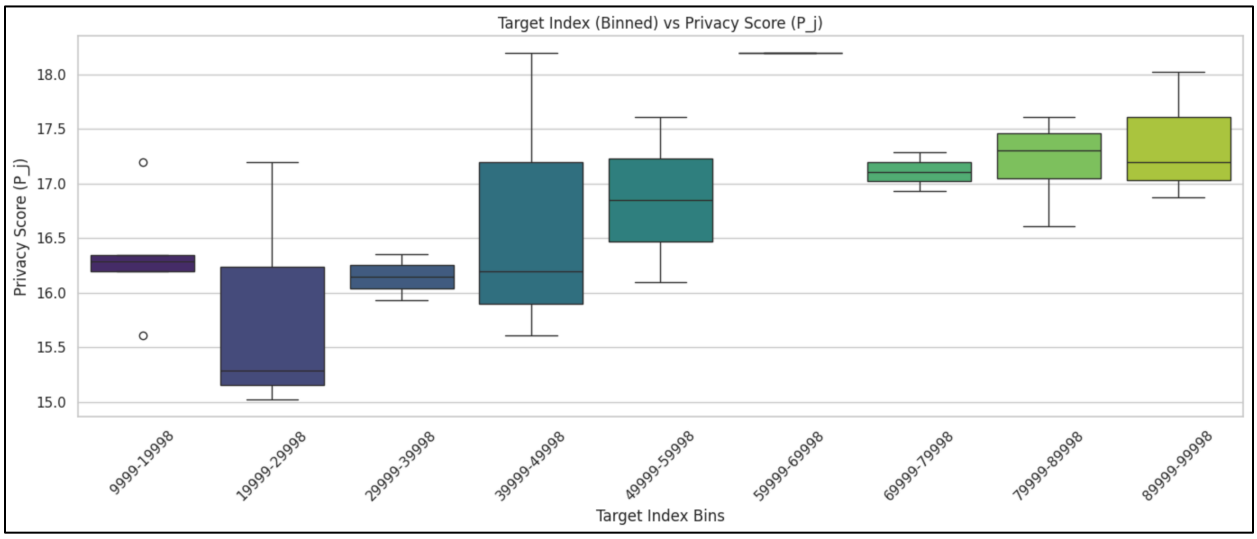


Figure 1.

Figure 1. reveals that privacy scores vary significantly across different index bins. Higher index bins consistently exhibit stronger and more stable privacy, while lower and mid-range bins show greater variance and lower median scores, indicating weaker and less consistent privacy guarantees. This disparity arises from uneven interval widths and suboptimal weight allocations

produced by the linear program. Additionally, the presence of outliers in the lower bins highlights that certain queries experience notably poor privacy. These observations underscore that privacy is influenced not only by the total computational budget or number of intervals, but critically by the placement and size of those intervals

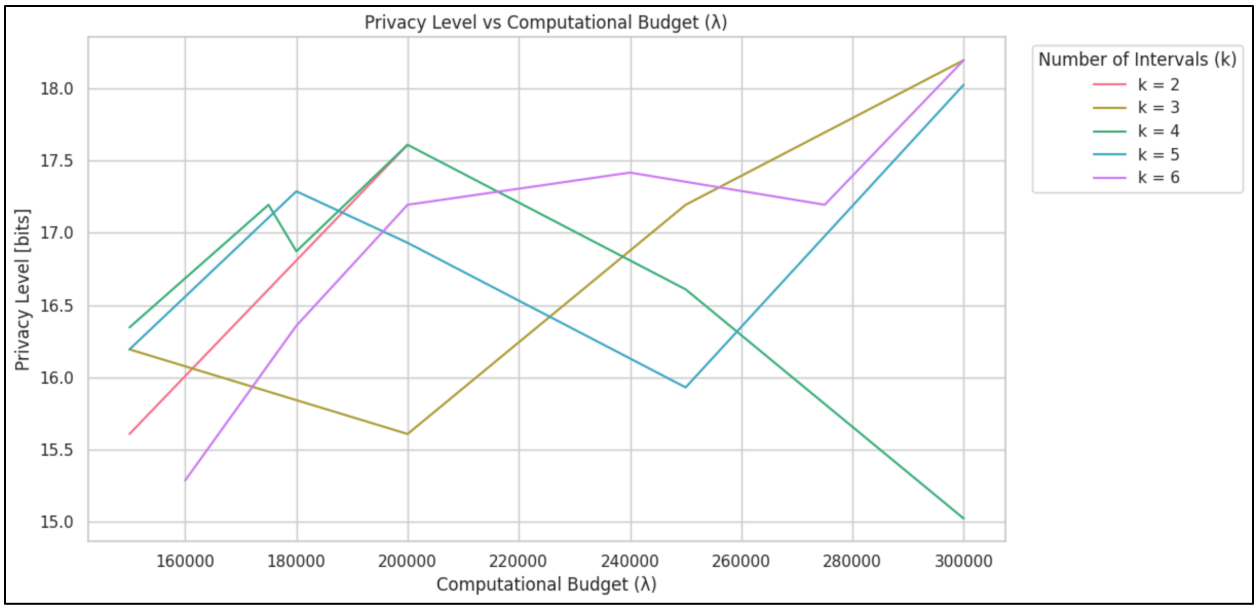
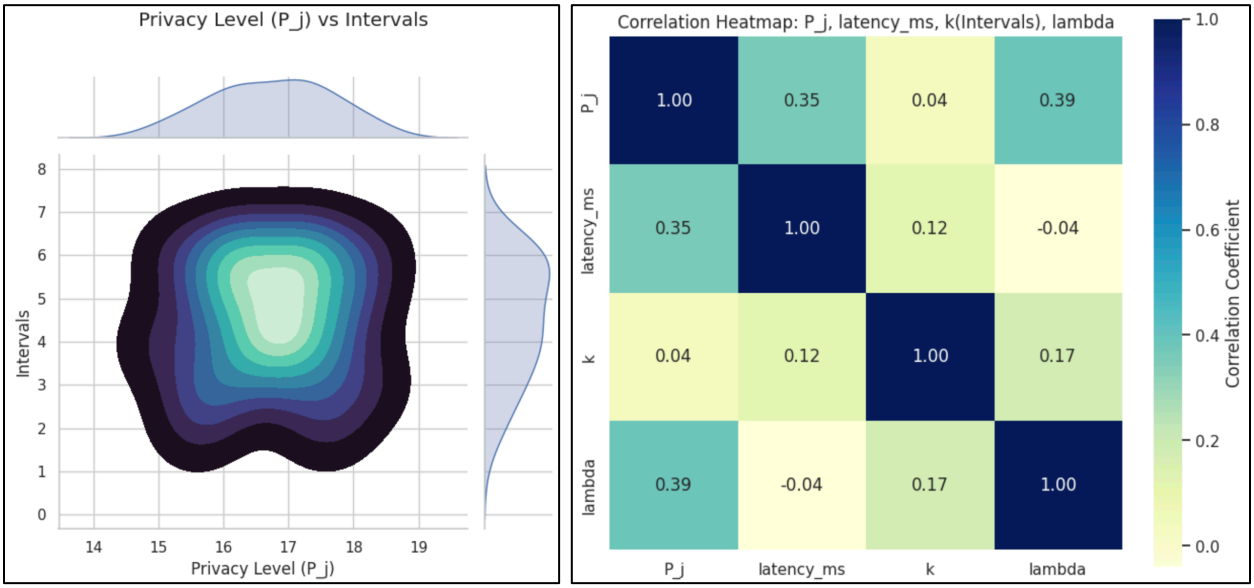


Figure 2.

Figure 2. shows that privacy level generally increases with computational budget, but the trend varies across different interval counts (k). Lower values of k tend to exhibit more predictable increases, while higher values show fluctuating or plateauing behavior. Some configurations even show a drop in privacy beyond a certain budget, indicating that simply increasing the budget or interval count does not guarantee better privacy. These patterns highlight the importance of jointly optimizing k and λ, as their interaction governs how effectively privacy is amplified.

In Figure 3.(a)The KDE plot shows that privacy levels are most densely concentrated around a moderate number of intervals, typically between 4 and 6. Configurations with very low or very high interval counts are less frequent and tend to yield less consistent privacy outcomes. This density concentration suggests that mid-range interval values offer the most balanced trade-off between privacy and computational cost. Importantly, the KDE plot highlights that privacy is not a linear function of interval count rather, it depends on how intervals are configured and interact with other system parameters like budget.The correlation heatmap Figure 3.(b) shows that privacy (P_j) has a moderate positive correlation with computational budget (λ) and latency, indicating their influence is present but not strictly proportional. Interval count (k) shows

minimal correlation with privacy, suggesting that interval structure and allocation matter more than quantity.



(a) (b)

Figure 3.

(**Note:** All visualizations were generated using data from a fixed database size of 100,000 entries. The full implementation used to produce these results, including code for quantized interval partitioning and DPF-based PIR query generation, is available on GitHub: https://github.com/RohitMahesh2004/Quantized-PIR-DPF.git.)

References

[1] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan.Private Information Retrieval.In Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS), pages 41–50. IEEE, 1995.

[2] Eyal Kushilevitz and Rafail Ostrovsky (1997). *Replication is Not Needed: Single Database, Computationally-Private Information Retrieval*. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 364–373. IEEE.

[3] Sergey Yekhanin (2010). *Private Information Retrieval*. *Communications of the ACM*, 53(4): 68–73.

[4] Adi Shamir. *How to share a secret*. Communications of the ACM, 22(11):612–613, 1979.

[5] Amos Beimel. *Secret-sharing schemes: A survey*. In Proceedings of the International Conference on Coding and Cryptology (ICCC), pages 11–46. Springer, 2011.

[6] Niv Gilboa and Yuval Ishai. Distributed Point Functions and Their Applications. In Advances in Cryptology - EUROCRYPT 2014, pages 640–658, 2014.

[7] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function Secret Sharing. In Advances in Cryptology - EUROCRYPT 2015, pages 337–367, 2015.

[8] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In CCS, pages 1292–1303, 2016.

[9] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. J. ACM, 33(4):792–807, 1986.