Preprocessed Private Function Evaluation: Achieving Sublinear Online Complexity for Lookup Tables

Tanping Zhou*
tanping2020@iscas.ac.cn
Engineering University of People's Armed Police
Xi'an, China

Xiaoyi Wang*
wangxiaoyi22@mails.ucas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China
University of Chinese Academy of Sciences
Beijing, China

Yi Qu
quyi23@mails.ucas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China
University of Chinese Academy of Sciences
Beijing, China

Wenchao Liu
liuwch3@alumni.sysu.edu.cn
Engineering University of People's Armed Police
Xi'an, China

Long Chen†
chenlong@iscas.ac.cn
Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software Chinese Academy of Sciences
Beijing, China

Zhenfeng Zhang†
zhenfeng@iscas.ac.cn
Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software Chinese Academy of Sciences
Beijing, China

Abstract

Private Function Evaluation (PFE) facilitates the secure computation of private functions on private inputs in an oblivious manner, ensuring that both the function and the inputs remain confidential throughout the entire computational process. PFE has garnered significant attention due to its critical applications in various domains, such as privacy-preserving healthcare systems and privacy-preserving credit checks, where safeguarding the confidentiality of the function itself is of paramount importance.

However, despite its broad applicability, existing PFE schemes often exhibit inefficiencies, even in relatively straightforward scenarios such as the evaluation of lookup tables. To mitigate these limitations, we propose a novel variant of PFE, termed Preprocessed Private Function Evaluation (PPFE), which leverages preprocessing techniques to significantly enhance the efficiency of online computations. Within this framework, we introduce a specialized construction tailored specifically for lookup table operations, achieving sublinear complexity during the online computation phase.

The efficacy of the proposed approach is demonstrated through experimental evaluations. For a lookup table of size 2^{24}, the online computation time required to process a single query is about 3 milliseconds, representing a performance improvement of more than an order of magnitude compared to existing results. Furthermore, the proposed scheme exhibits strong scalability, effectively handling thousands of adaptive queries within the same framework.

Keywords

Private Function Evaluation, Table Lookup, Sublinear Online Computation

*Both authors contributed equally to this research.
†Corresponding authors.

This is the full version of the paper appearing in the Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands. DOI: https://doi.org/10.1145/3830454.3832657.

1 Introduction

Private Function Evaluation (PFE) [1, 5, 7–9, 14, 15, 18, 20, 22, 27, 30, 33, 36, 37] allows a client P_1 holding a private input x and a server P_2 holding a private function f to securely compute $f(x)$. Unlike standard Secure Function Evaluation (SFE) where the function is public, PFE treats the function f itself as a private input. The protocol guarantees that P_2 learns nothing about x (input privacy) and P_1 learns nothing about f beyond the computation result (function privacy).

Private Function Evaluation for Lookup Tables (PFE-LUT). While general PFE remains computationally expensive [18, 22], many practical discrete functions can be efficiently represented as Lookup Tables (LUTs). In the PFE-LUT setting, P_2 holds a table $T = [T[0], \ldots, T[n-1]]$ of size n encoding the function, and P_1 holds a query index $x \in [0, n-1]$. The goal is for P_1 to retrieve $T[x]$ without revealing x to P_2, and without P_1 learning any other entries $T[i]$ where $i \neq x$. Although PFE-LUTs inherently require bounded input domains, they are highly practical for complex real-world tasks because inputs are typically discretized, compressed, or mapped to bounded indices. This abstraction captures a broad class of privacy-preserving applications, including:

- **Machine Learning & Healthcare.** In healthcare, LUTs efficiently handle privacy-sensitive matching tasks (e.g., as discussed in FLORAM [12]), where the domain is registered participant IDs. In privacy-preserving machine learning (PPML), LUTs are the standard mechanism for evaluating embedding layers (e.g., FABLE [38]). The typical item ID space for these models (10^4–10^6) perfectly fits our protocol's supported capacity (e.g., $n = 2^{20}$).
- **Proprietary Primitives & Security.** Cryptographic components, such as proprietary S-boxes in key derivation functions (KDFs), are natively small-domain LUTs (e.g., 8-bit

inputs). Complex proprietary functions can be securely evaluated by composing multiple small PFE-LUTs (e.g., MAESTRO [31]). Furthermore, in privacy-preserving Intrusion Detection Systemss (IDS), large or sparse malware signature domains can be efficiently mapped into smaller, effective index domains via Deterministic Finite Automaton (DFA) state compression [32], making LUT-based evaluation highly practical.

Scope. Our construction addresses PFE where the private function is represented as a lookup table, rather than arbitrary general-purpose PFE.

1.1 Related Work on PFE-LUT

Relationship to SPIR. PFE-LUT is functionally equivalent to single-server Symmetric Private Information Retrieval (SPIR): PIR protects the query index, while SPIR additionally requires protecting non-queried table entries. Classic paradigms for constructing SPIR protocols typically involve augmenting standard PIR with server-side privacy mechanisms [26] or utilizing Labeled Private Set Intersection (PSI) variants [25]. Our construction builds upon the preprocessing PIR framework of RMS [35] to achieve query index privacy. However, since RMS (and PIR in general) provides no server data privacy, we incorporate additional cryptographic mechanisms to fulfill the symmetric security requirement. To the best of our knowledge, no existing single-server SPIR achieves sublinear online costs for individual adaptive queries as efficiently as ours.

Circuit-based PFE (UC and GC). Circuit-based PFE approaches transform the LUT operation $y \leftarrow T[x]$ into a selection network (e.g., a multiplexer tree) and conceal the function's topology through one of two strategies: embedding the network inside a Universal Circuit (UC) [24] whose fixed structure hides the function, or directly encrypting the circuit's topology using homomorphic encryption and oblivious transfer [22]. In both cases, the hidden circuit is evaluated using garbled circuit techniques. These methods offer robust function privacy, but share a fundamental limitation: the online computational cost for both parties scales linearly with the table size ($O(n)$), since the entire selection network must be evaluated.

Fully Homomorphic Encryption (FHE). FHE-based approaches [2, 17] typically involve the client encrypting the index and the server performing homomorphic selection. While these methods achieve low communication overhead, they suffer from two major limitations: (1) **Linear Server Computation**: The server must process every table entry homomorphically, resulting in $O(n)$ complexity; (2) **Heavy Cryptographic Overhead**: Evaluating complex functions often necessitates bootstrapping or large parameters. Furthermore, function privacy typically requires noise flooding [16], which further increases computational costs.

Specialized 2PC and DORAM. Protocols like SP-LUT [11] and FLUTE [6] optimize 2PC for LUTs but generally exhibit linear communication costs ($O(n)$), limiting scalability. FABLE [38] improves amortized efficiency for batched queries but imposes significant overhead for single queries. Alternatively, Distributed Oblivious RAM (DORAM) schemes like FLORAM [12] and 2P-DUORAM [39] are designed for secure 2-party computation. While they successfully reduce communication complexity to sublinear levels (e.g., logarithmic), they fundamentally rely on Function Secret Sharing (FSS) or DPF techniques that require **linear local computation** ($O(n)$) by *both* parties to evaluate the access keys over the entire memory. In the PFE setting, this imposes a prohibitive computational burden on the client, contradicting the goal of a lightweight client. Furthermore, these protocols incur additional overheads to support read-write operations, which are unnecessary for static, read-only lookup tables.

Research Gap. Despite these advancements, a critical gap remains: no existing PFE-LUT protocol simultaneously achieves **sublinear online computation** for the server and **lightweight complexity** for the client. Current sublinear solutions (DORAM) burden the client with linear local operations, while server-efficient solutions (FHE/GC) scale linearly in computation or communication. This work bridges this gap by introducing a preprocessed framework that ensures sublinear complexity for both parties.

1.2 Our Contributions

In this paper, we introduce *Preprocessed Private Function Evaluation for Lookup Tables (PPFE-LUT)*, a novel framework that leverages offline preprocessing to achieve highly efficient online evaluation for lookup tables. Our specific contributions are as follows:

- **Sublinear and Lightweight PFE-LUT Framework.** We propose the first PFE-LUT protocol that simultaneously achieves **sublinear online computation** for the server and **lightweight complexity** for the client. Unlike FHE-based approaches (linear server cost) or DORAM-based approaches (linear local computation), our protocol utilizes a preprocessed lookup table and additive homomorphic encryption (AHE). This design decouples the function size from the online cost, ensuring that both communication and computation scale sublinearly with the table size n during the online phase.
- **Efficient Oblivious Decryption Mechanism.** We design a novel oblivious decryption protocol tailored for BFV-style ciphertexts. It splits the decryption process into a linear partial decryption and a non-linear final decryption. Crucially, instead of relying on *noise flooding*—which necessitates large parameters and degrades performance—we utilize a "large mask, small hint" strategy. This allows the server to add only a minimal perturbation noise during partial decryption to guarantee key privacy, achieving high efficiency without the heavy computational overhead associated with traditional noise flooding or bootstrapping.
- **Theoretical Foundation via Adaptive Hint-MLWE.** To rigorously prove the security of our protocol, we introduce and formalize the **Adaptive Hint-MLWE** problem. This new primitive models the security of Module-LWE when an adversary learns multiple linear combinations of the secret key and encryption errors, effectively capturing the leakage behavior in our setting. We bridge the theoretical gap between standard MLWE and Adaptive Hint-MLWE by proving that the latter is securely reducible to the standard

MLWE assumption. This reduction serves as the theoretical foundation for our oblivious decryption protocol.

- **Implementation and Evaluation.** We provide a full implementation of PPFE-LUT and compare it against state-of-the-art lookup-table protocols, including FABLE [38], SP-LUT [11], and DORAM variants. For a table with 2^{24} entries, PPFE-LUT achieves a **7.9× speedup** in WAN environments and **171× speedup** in LAN environments for online runtime compared to the evaluated 2PC-based LUT schemes. The protocol processes each query separately, so it supports adaptive query sequences without requiring future queries to be batched.

Table 1 provides a comprehensive comparison between our proposed PPFE and existing solutions. To the best of our knowledge, PPFE is the only solution that satisfies all key efficiency and functionality requirements for practical, privacy-preserving lookup table evaluation.

1.3 Technique Overview

Achieving sublinear complexity for Private Function Evaluation (PFE) on lookup tables is challenging, as it typically requires the server to scan the entire table to hide the access pattern. We address this by leveraging recent breakthroughs in preprocessing PIR (e.g., Piano [**?**], RMS [35]), which achieve sublinear online costs using simple additive operations.

From PIR to PFE: An Encrypted Database Approach. Observing that PIR and PFE differ primarily in server data privacy, we adopt an **"Encrypted Database"** approach. The server (P_2) encrypts the table T using Ring-LWE, and the client (P_1) executes the PIR protocol directly over the ciphertexts. This step is critical to enforce strict function privacy; it ensures that any non-queried table entries accessed or transmitted during the PIR process remain semantically secure, thereby preventing the client from learning any information outside of the intended query result. Crucially, since efficient PIR schemes rely solely on additive homomorphisms, our framework avoids computationally intensive operations like homomorphic multiplication or bootstrapping. This allows P_1 to retrieve an encrypted result $\mathbf{ct}_{res} = \mathsf{Enc}(T[x])$ while inheriting the sublinear efficiency of the underlying PIR.

The Core Challenge: The Decryption Deadlock. This approach introduces a critical deadlock regarding $\mathbf{ct}_{res}$: P_1 cannot decrypt it without the secret key, while P_2 cannot be allowed to decrypt it, as doing so would reveal the plaintext $T[x]$ and the access pattern. The main technical contribution of this work is resolving this deadlock without compromising efficiency or privacy.

A Trivial but Insecure Approach. A naive attempt to solve this is a simple interactive decryption: P_1 homomorphically adds a random plaintext mask r_{pt} to $\mathbf{ct}_{res}$, sends it to P_2 for decryption, and then subtracts r_{pt} from the result. However, this method is fundamentally insecure due to the structure of Ring-LWE ciphertexts, which roughly are pairs $(b = -a \cdot s + e + m, a)$. Since P_2 generated the encrypted database, they know the polynomial component a_i in the ciphertext for all entries $i \in [n]$ (where n is the table size). The a_{res} component in the result ciphertext is a linear combination of these public a_i's. If P_2 observes a_{res}, they can solve the linear system to deduce the combination coefficients, instantly revealing the client's access pattern x.

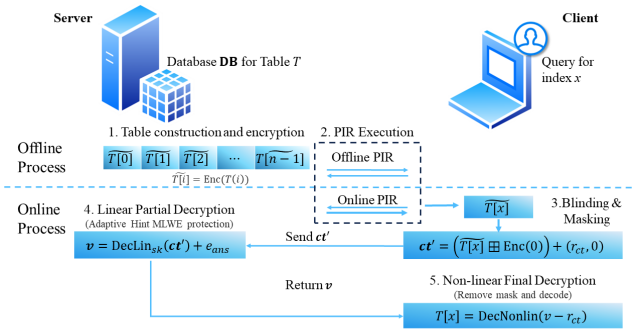

Figure 1: PPFE Protocol Overview.

First Fix: Ciphertext Blinding. To block the leakage from a, we introduce **Ciphertext Blinding**. Before sending anything to P_2, P_1 homomorphically adds a fresh encryption of zero:

$$\mathbf{ct}_{blinded} = \mathbf{ct}_{res} \boxplus \mathsf{Enc}(0)$$

Since $\mathsf{Enc}(0)$ contains a freshly generated a_{zero} unknown to P_2, this operation completely randomizes the a component, preventing P_2 from tracing the linear combination.

The Persistence of Error Leakage. While blinding protects the a component, it does not solve the problem entirely. The **Error Term (e)** within $\mathbf{ct}_{blinded}$ still retains the accumulated noise from the PIR computation path. This creates a new deadlock:

- **Risk to Client Privacy:** If P_2 fully decrypts the ciphertext, they will observe the raw error term e. Since the magnitude of e correlates with the homomorphic operations, this can leak the access pattern.

A standard countermeasure is **Noise Flooding**, where a massive noise is added to "drown out" the error term. However, this forces the use of exponentially larger encryption parameters, rendering the protocol impractical.

Our Intuition: The "One-Time Pad" in Ciphertext Space. Instead of trying to "drown out" the error with more noise, we observe that the server only needs to perform a *linear operation* on the secret key to assist decryption. This allows us to use a more elegant solution: **Ciphertext Blinding**. The client adds a uniform random mask r_{ct} (in the ciphertext modulus space) directly to the ciphertext component b.

- **Why this is efficient:** Unlike noise flooding, r_{ct} is a uniform element in the ciphertext modulus space $\mathcal{R}_q$. It acts as a one-time pad that *perfectly* hides both the message and the error e from the server without requiring any increase in the noise budget or the modulus q.
- **The Trade-off:** The cost of this perfect hiding is that the server can no longer "decrypt" the message to completion. It only sees a uniform random string.

Table 1: Comparison of Our Scheme with State-of-the-Art PFE and LUT Protocols. Online Comp.: Server/Client online computation complexity. Online Comm.: Online communication complexity. Offline Comp.: Total Server/Client computation in one offline/preprocessing phase (for PPFE-LUT: total cost per phase). Privacy: Symmetric privacy (both function and input protected). Adaptive: Supports adaptive queries where each query can depend on results of previous queries. Rounds: Interaction rounds in Offline (Off) and Online (On) phases. (•: Yes, ○: No; n: table size)

Protocol	Online Comp.		Online Comm.	Offline Comp.		Functionality		Rounds (Off/On)
	Server	Client		Server	Client	Privacy	Adaptive	
FHE-based PFE	$O(n)$	$O(\log n)$	$O(\log n)$	–	–	•	•	0/2
GC-based PFE [22]	$O(n)$	$O(n)$	$O(n)$	–	–	•	•	0/3
SP-LUT [11]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	○	•	2/2
FLUTE [6]	$O(n^2)$	$O(n^2)$	$O(\sigma)$[a]	$O(n)$	$O(n)$	○	•	$O(\log\log n)$/1
FLORAM [12]	$O(n)$	$O(n)$	$O(\log n)$	–	–	•	•	0/$O(\log n)$[c]
DUORAM [39]	$O(n)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	•	•	2/2[d]
FABLE [38]	$O(n)$[b]	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(n)$	–	•	○	0/4
PPFE-LUT (This Work)	$\mathbf{O(\sqrt{n})}$	$\mathbf{O(\sqrt{n})}$	$\mathbf{O(\sqrt{n})}$	$\mathbf{O(n)}$	$\mathbf{O(n)}$	•	•	**1/2**

[a] σ denotes output bits. [b] Using non-preprocessing PIRANA; server performs $O(n)$ online work. [c] Using FLORAM with CPRG optimization; basic FLORAM requires 3 rounds.

[d] Based on its 2-party read-only (CSPIR) variant. Its general ORAM mode requires $O(logn)$ offline rounds.

PPFE-LUT: One preprocessing phase supports $O(\sqrt{n})$ queries, yielding $O(n)/O(\sqrt{n})$ amortized offline computation cost. The server-side computation $O(n)$ in the offline phase consists of table encryption, which is a **one-time setup**. This encrypted database can be reused across multiple preprocessing sessions and by different clients as long as f remains static.

Refining the Protection: Oblivious Decryption via Hint-MLWE. To overcome the server's inability to fully decrypt, we introduce **Oblivious Decryption**, as shown in Fig. 1. The server computes the partial decryption result $v = b_{masked} + a \cdot s \pmod q$ and returns it to the client. Since the client knows the mask r_{ct}, they can subtract it to recover the result.

However, this creates a new leakage: the value $v - r_{ct}$ (which is message + key-dependent error terms) is now in the client's hands. To prevent the client from extracting the server's secret key s from this "hint", the server adds a *small* perturbation noise e_{ans} before returning the result. Crucially, while e_{ans} is small and does not affect the LWE parameters, its security is rigorously guaranteed by the **Adaptive Hint-MLWE assumption**. This allows us to maintain standard, efficient LWE parameters while achieving full PFE security.

Theoretical Contribution: Adaptive Hint-MLWE. Providing a rigorous security proof for Oblivious Decryption is challenging because the partial decryption result provides a "hint" representing a linear combination of the server's secret key and encryption errors, where the coefficients of this combination are explicitly known to the client. Crucially, standard MLWE assumptions do not account for such leakages, forcing conventional protocols to rely on costly *noise flooding* to achieve circuit privacy. To bridge this gap, we formalize the **Adaptive Hint-MLWE** problem, which specifically models the security of MLWE instances when exposed to multiple linear hints (with small hint noise). We prove that this variant is reducible to the standard MLWE problem. This provides the mathematical justification for using a small perturbation noise e_{ans} to protect the secret key against such leakages, thereby achieving high efficiency without compromising theoretical rigor.

Optimizations for Efficiency. To ensure practical performance, we incorporate two key optimizations:

(1) **Batching with Rotate-and-Extract:** We utilize the SIMD property of Ring-LWE to pack multiple table entries. To retrieve a specific entry, P_1 homomorphically rotates the target slot to the constant term position. The server is designed to always decrypt and return only the constant term, significantly reducing communication.
(2) **Round Compression:** We merge the PIR finalization with the Client Masking step. In protocols like RMS, the client holds an intermediate state while the server holds an offset. We combine the mask addition with the offset handling, reducing the online interaction to just **2 rounds**.

2 Preliminaries

This section introduces the basic cryptographic primitives and notation used in our construction. For detailed descriptions of the RMS PIR protocol and pseudorandom functions (PRF), we refer readers to Appendix A.

2.1 Notation

Let N be a power of two and q be an integer. We denote the polynomial rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$. For a polynomial $f = \sum_{i=0}^{N-1} f_i X^i \in \mathcal{R}$, its ℓ^p norm ($p \geq 1$) is defined coefficient-wise as $\|f\|_p := (\sum_{i=0}^{N-1} |f_i|^p)^{1/p}$. For a vector of polynomials $\mathbf{f} = (f^{(0)}, \ldots, f^{(m-1)}) \in \mathcal{R}^m$, its ℓ^p norm is defined as

$\|\mathbf{f}\|_p := (\sum_{j=0}^{m-1} \|f^{(j)}\|_p^p)^{1/p}$. For a real-valued matrix $\mathbf{A} \in \mathbb{R}^{m\times N}$, $\|\mathbf{A}\|_2$ denotes its spectral norm (i.e., the largest singular value), while $\sigma_{\max}(\mathbf{A})$ and $\sigma_{\min}(\mathbf{A})$ represent its largest and smallest singular values, respectively.

2.2 Probabilistic Distributions

Discrete Gaussians. For a lattice $\Lambda \subset \mathbb{R}^N$, center $\mathbf{c}$, and positive-definite matrix Σ, the discrete Gaussian distribution $\mathcal{D}_{\Lambda,\mathbf{c},\sqrt{\Sigma}}$ assigns probability proportional to $\rho_{\mathbf{c},\sqrt{\Sigma}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^\top \Sigma^{-1}(\mathbf{x}-\mathbf{c}))$ for each $\mathbf{x} \in \Lambda$. We denote $\mathcal{D}_{\mathbb{Z}^N,\sigma}$ for spherical variance $\sigma^2\mathbf{I}$. The smoothing parameter $\eta_\varepsilon(\Lambda)$ [29] is the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \le \varepsilon$. It is bounded by $\eta_\varepsilon(\Lambda) \le \lambda_N(\Lambda) \cdot \sqrt{\ln(2N(1+1/\varepsilon))/\pi}$.

Convolution Lemma. Our security analysis relies on the fact that the sum of independent discrete Gaussian variables approximates a discrete Gaussian with combined covariance [34]. Specifically, let Σ_1, Σ_2 be positive definite and define Σ_3 such that $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1}$. If $\|\Sigma_3^{-1}\|_2 \le \eta_\varepsilon(\mathbb{Z}^N)^{-2}$ (which implies smoothness [23]), then for $\mathbf{x}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sqrt{\Sigma_1}}$ and $\mathbf{x}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c},\sqrt{\Sigma_2}}$, the sum $\mathbf{x}_1 + \mathbf{x}_2$ is statistically indistinguishable (within distance 2ε) from $\mathcal{D}_{\mathbb{Z}^N,\mathbf{c},\sqrt{\Sigma_1+\Sigma_2}}$.

2.3 Homomorphic Encryption

We employ the BFV scheme [3, 13] over the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N+1)$ with coefficient encoding. The scheme consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda) \to pp$: Generates parameters $pp = (N, p, q, \chi_{\mathsf{key}}, \chi_{\mathsf{err}}, \chi_{\mathsf{enc}})$.
- $\mathsf{KeyGen}(pp) \to (pk, sk)$: Sample $s \leftarrow \chi_{key}$, $a \leftarrow \mathcal{U}(\mathcal{R}_q)$, $e \leftarrow \chi_{err}$. Set $sk = s$ and $pk = (b = -as + e, a) \in \mathcal{R}_q^2$.
- $\mathsf{Enc}(m)$: Encodes vector $\mathbf{m} \in \mathbb{Z}_p^N$ to polynomial m. For secret-key encryption, output $\mathbf{ct} = (-as + \lfloor q/p \rfloor m + e, a)$. For public-key encryption, sample $u \leftarrow \chi_{enc}, e_0 \leftarrow \chi_{err}, e_1 \leftarrow \chi_{err}$, output $\mathbf{ct} = (bu + \lfloor q/p \rfloor m + e_0, au + e_1)$.
- $\mathsf{Dec}_{sk}(\mathbf{ct})$: We decompose decryption into two steps:
  - $\mathsf{DecLin}_{sk}(\mathbf{ct})$: Given $\mathbf{ct} = (c_0, c_1)$, compute $m' = c_0 + c_1 \cdot s \pmod q$.
  - $\mathsf{DecNonlin}(m')$: Scale and round to recover message: $m = \lfloor \frac{p}{q} \cdot m' \rceil \pmod p$.
- $\mathsf{Eval}(\mathbf{ct}, \dots)$: We utilize homomorphic addition $\mathbf{ct}_{add} \leftarrow \mathbf{ct} \boxplus \mathbf{ct}'$ (component-wise addition) and scalar multiplication. Specifically, multiplication by a monomial $\mathsf{CMul}(\mathbf{ct}, X^{-k})$ performs a homomorphic negacyclic rotation, shifting the plaintext coefficient vector by k positions.

Hardness Assumption. The security relies on the Module-LWE (MLWE) problem [4].

Definition 2.1 (MLWE Problem). Let χ be an error distribution over $\mathcal{R}$. The $\mathrm{MLWE}_{d,m,q,\chi}$ assumption states that it is computationally hard to distinguish $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ from $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times d})$, $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$, $\mathbf{s} \leftarrow \chi^d$, and $\mathbf{e} \leftarrow \chi^m$.

Note that the Ring-LWE (RLWE) problem [28] corresponds to the special case of MLWE where the module rank $d = 1$.

2.4 Preprocessing PIR

Private Information Retrieval (PIR) is a fundamental cryptographic primitive that enables a client to retrieve a specific entry from a database held by a server without revealing the retrieved index. Formally, let the database be denoted as a vector $T = (T[0], T[1], \dots, T[n-1]) \in \mathbb{D}^n$, where $\mathbb{D}$ is the domain of the entries. The client, holding an index $x \in \{0, \dots, n-1\}$, interacts with the server to compute $T[x]$ such that the server gains no information about x.

In this work, we build upon the **RMS** protocol proposed by Ren et al. [35], which is currently among the most efficient single-server PIR schemes. RMS achieves sublinear online computation and communication by leveraging an offline preprocessing phase. While a more recent proposal, Plinko [19], offers theoretically improved asymptotics, it relies on invertible pseudorandom functions (iPRFs), which introduce significant concrete computational overhead. In contrast, RMS relies only on standard lightweight primitives, making it a more practical foundation for constructing efficient Private Function Evaluation (PFE) protocols. The RMS framework treats the database as a matrix $\mathbf{T} \in \mathbb{D}^{\sqrt{n}\times\sqrt{n}}$ with $\sqrt{n}$ rows, each containing $\sqrt{n}$ entries. The workflow, illustrated in Fig. 7, proceeds in two phases (see Appendix A for details):

Offline Preprocessing. The client downloads the database in a streaming fashion to generate a local state consisting of *hints*.

(1) **Hint Generation:** The client generates a collection of primary hints $\mathcal{H}_{pri}$. Each hint contains a random subset of indices $S \subset \{0, \dots, n-1\}$, which we refer to as a hint set. RMS constructs S by selecting $\sqrt{n}/2 + 1$ rows and picking one random index per row.[1]
(2) **Parity Storage:** For each set S, the client also computes and stores the parity $p = \sum_{j\in S} T[j]$. A hint is thus the tuple (S, p). Additionally, the client generates a set of *backup hints* $\mathcal{H}_{bak}$ to support multi-query capabilities.

Online Querying. To retrieve the element $T[x]$:

(1) **Hint Selection:** The client searches its local storage for a primary hint (S, p) such that $x \in S$.
(2) **Query Construction (Dummy Subset Technique):** The client partitions the random indices from different rows into two sets to hide x.
   - It constructs the *real set* $S' = S \setminus \{x\}$. This set contains indices from the rows covered by the hint, excluding the row containing x.
   - It constructs a *dummy set* S^* by selecting one uniformly random index from every row *not* represented in S'.

   Crucially, $S' \cup S^*$ covers exactly one random index from every row in the database, each containing $\sqrt{n}/2$ indices. The client sends the pair $\{S', S^*\}$ to the server in a random order, denoted as (S_0, S_1). We note that the distribution of (S_0, S_1) is independent of the query target x.
(3) **Server Response:** The server computes the sum of elements for each received set: $a_b = \sum_{j\in S_b} T[j]$ for $b \in \{0, 1\}$, and returns (a_0, a_1).
(4) **Reconstruction:** The client identifies the response a_{real} corresponding to S'. The target value is recovered as $T[x] = p - a_{real}$.

[1] To save storage, S is succinctly represented by a PRF seed. This "at most one index per row" structure allows the client to efficiently determine an index's membership in S via PRF evaluations.

Hint Refreshing. A critical requirement for security is that the primary hints used for queries must follow a specific stationary distribution. However, utilizing a hint consumes it (since reusing it for the same query pattern would leak information). To support unlimited queries, the protocol employs a refresh mechanism. After each query, the client utilizes the pre-generated *backup hints* to replenish the consumed primary hint. This ensures that the distribution of available hints remains consistent across queries, thereby preventing the server from inferring information through statistical analysis of the query patterns.

3 Adaptive Hint-MLWE

To facilitate the security analysis of our PFE protocol, where the client adaptively selects query indices, we introduce a variant termed *Adaptive Hint-MLWE*. This formulation extends the Hint-MLWE framework established in [23] in two key dimensions. First, it allows the adversary to select any challenge polynomials (satisfying the norm bound), whereas the original problem assumes challenges are sampled from a fixed distribution.[2] Second, it adopts an inner-product structure for the hints to accommodate the vectorized secrets and errors inherent in our protocol.

Definition 3.1 (Adaptive Hint-MLWE Problem). Let d, m, Q be positive integers, χ be a distribution over $\mathcal{R}^{d+m}$ and ξ be a distribution over $\mathcal{R}^{Q}$. Let $B > 0$ be a real number bound. The Adaptive Hint-MLWE problem, denoted by $\text{A-HintMLWE}^{Q,\xi,B}_{R,d,m,q,\chi}$, is defined by the following game between a challenger C and an adversary $\mathcal{A}$:

(1) **Setup:** The challenger samples $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times d})$, $\mathbf{r} \leftarrow \chi$, and $\mathbf{y} \leftarrow \xi$.
(2) **Query:** The adversary $\mathcal{A}$ outputs a sequence of challenge vectors $(\gamma_0, \ldots, \gamma_{Q-1})$, where $\gamma_i = (\gamma_{i,0}, \cdots, \gamma_{i,d+m-1}) \in \mathcal{R}^{d+m}$ is a vector of polynomials, subject to the norm constraint $\sum_{i=0}^{Q-1} \sum_{j=0}^{d+m-1} \|\gamma_{i,j}\|_1^2 \le B$.
(3) **Response and Challenge:** The challenger computes the hints $z_i = \gamma_i^\top \mathbf{r} + y_i$ for $0 \le i < Q$.
Then, the challenger samples a bit $b \leftarrow \{0,1\}$.
- If $b = 0$ (Real), set **target** $= [\mathbf{I}_m|\mathbf{A}]\mathbf{r}$.
- If $b = 1$ (Random), set **target** $= \mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$.

The challenger sends $(\mathbf{A}, \textbf{target})$ and $(z_0, \ldots, z_{Q-1})$ to $\mathcal{A}$.
(4) **Guess:** The adversary outputs a bit b'.

The adversary's advantage is defined as $|\Pr[b' = b] - 1/2|$. We call the $d = 1$ case Adaptive Hint-RLWE.

Despite these modifications, we demonstrate that the hardness of Adaptive Hint-MLWE remains reducible to the standard MLWE assumption.

Lemma 3.2. *Let $Q > 0$ be an integer and $\sigma_1, \sigma_2 > 0$ be reals. For $\gamma_0, \cdots, \gamma_Q - 1 \in R^K$, where $\gamma_i = (\gamma_{i,0}, \cdots, \gamma_{i,K-1})$, we denote the corresponding negacyclic matrix of $\gamma_{i,j}$ as $\mathbf{M}(\gamma_{i,j})$, and we let $\mathbf{\Gamma}_i = [\mathbf{M}(\gamma_{i,0})|\mathbf{M}(\gamma_{i,1})|\cdots|\mathbf{M}(\gamma_{i,K-1})]$. We let $\mathbf{\Sigma}_0 = (\frac{1}{\sigma_1^2}\mathbf{I}_{NK} + \frac{1}{\sigma_2^2}\sum_{i=0}^{Q-1} \mathbf{\Gamma}_i^\top \mathbf{\Gamma}_i)^{-1}$. Then, the following two distributions over R^{K+Q} are statistically identical.*

$$\left\{(\mathbf{r}, z_0, \ldots, z_{Q-1}) \;\middle|\; \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{NK},\sigma_1},\ y_i \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_2},\ z_i = \gamma_i^\top \mathbf{r} + y_i\right\}$$

$$\left\{(\hat{\mathbf{r}}, z_0, \ldots, z_{Q-1}) \;\middle|\; \begin{array}{l} \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{NK},\sigma_1},\ y_i \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_2},\ z_i = \gamma_i^\top \mathbf{r} + y_i, \\ \mathbf{c} = \frac{1}{\sigma_2^2}\mathbf{\Sigma}_0 \cdot \sum_{i=0}^{Q-1} \mathbf{\Gamma}_i^\top z_i,\ \hat{\mathbf{r}} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c},\sqrt{\Sigma_0}} \end{array}\right\}$$

The proof is provided in Appendix B.1.

Theorem 3.3 (Hardness of Adaptive Hint-MLWE). *Let d, m, Q, q be positive integers. Let $B > 0$ be a real number. Let $\chi = \mathcal{D}^{d+m}_{\mathbb{Z}^N,\sigma_1}$ and $\xi = \mathcal{D}^{Q}_{\mathbb{Z}^N,\sigma_2}$ be discrete Gaussian distributions for the secret and hint errors, respectively.*

Let $\sigma > 0$ be a real number satisfying the following condition:

$$\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_1^2} + \frac{B}{\sigma_2^2}\right). \tag{1}$$

If $\sigma \ge \sqrt{2}\cdot\eta_\varepsilon(\mathbb{Z}^{N(d+m)})$ for some negligible $\varepsilon \in (0, 1/2)$, then there exists an efficient reduction from $\text{MLWE}_{R,d,m,q,\sigma}$ to $\text{A-HintMLWE}^{Q,\xi,B}_{R,d,m,q,\chi}$ that reduces the advantage by at most 2ε.

Specifically, for any adversary $\mathcal{A}$ that solves the Adaptive Hint-MLWE problem with a set of queries (or challenge vectors) γ satisfying $\sum_{i=0}^{Q-1}\sum_{j=0}^{d+m-1}\|\gamma_{i,j}\|_1^2 \le B$, there exists a simulator that solves the standard MLWE problem.

The proof is provided in Appendix B.2.

4 Preprocessed Private Function Evaluation

In this section, we provide the formal definition of the Preprocessed Private Function Evaluation (PPFE) scheme.

We denote the computational security parameter by λ and the statistical security parameter by κ. We use $\text{negl}(\cdot)$ to denote a negligible function. Let $f : \{0,1\}^\ell \to \{0,1\}^{\ell'}$ be the function to be evaluated, where $n = 2^l$ denotes the size of the input domain (i.e., the size of the lookup table). Let $\mathbf{x} = \{x_t \in \{0,1\}^\ell\}_{t\in\{1,\ldots,Q\}}$ denote the sequence of inputs for Q queries. While denoted as a static sequence for notational convenience, the queries can be strictly adaptive, meaning each index x_t may be dynamically chosen based on the retrieved plaintexts $\{f(x_1), \ldots, f(x_{t-1})\}$ from previous queries.

4.1 Syntax and Correctness

We define a PPFE scheme as a two-party protocol between a client P_1 and a server P_2. The scheme consists of two phases: an offline setup phase and an online evaluation phase.

Definition 4.1 (Preprocessed PFE). A Preprocessed PFE scheme $\Pi = (\text{Setup}, \text{Evaluate})$ is defined as follows:

- **Phase 1: Offline Setup** (Setup). This protocol is executed once. P_1 and P_2 input the security parameter 1^λ and the statistical security parameter 1^κ. P_2 inputs the function f. At the end of the protocol, P_1 obtains an initial state $\text{st}_1^{(0)}$, and P_2 obtains an initial state $\text{st}_2^{(0)}$.
- **Phase 2: Online Evaluation** (Evaluate). This protocol is executed for a sequence of queries $t = 1, \ldots, Q$. In the t-th execution:
  – P_1 inputs its current state $\text{st}_1^{(t-1)}$ and a query $x_t \in \mathbf{x}$.

[2] This is essential. It allows the adversary to select challenge polynomials that model adaptive query choices in the PFE protocol, where subsequent challenges may depend on previously retrieved results.

– P_2 inputs its current state $\text{st}_2^{(t-1)}$.

At the end of the protocol, P_1 obtains the output y_t and updates its state to $\text{st}_1^{(t)}$. P_2 updates its state to $\text{st}_2^{(t)}$.

Correctness. The scheme Π is correct if for any function f, any sequence of inputs $\mathbf{x} = \{x_t\}_{t=1}^{Q}$, and any valid execution of the protocols, the probability that P_1 outputs $y_t = f(x_t)$ for all $t \in \{1, \ldots, Q\}$ is at least $1 - \text{negl}(\lambda) - \text{negl}(\kappa)$.

4.2 Security Definition

We adhere to the standard simulation-based security definition for semi-honest adversaries. Intuitively, the server's view should reveal nothing about the client's inputs $\mathbf{x}$, and the client's view should reveal nothing about the function f beyond what can be inferred from the outputs $\{y_t\}$.

Let $\text{VIEW}_i^{\Pi}(1^\lambda, \mathbf{x}, f)$ denote the view of party P_i during the execution of Π. The view includes the party's input, random tape, and all messages received during both the Setup and Evaluate phases.

Definition 4.2 (Semi-Honest Security). Protocol Π is a secure preprocessed PFE protocol for semi-honest adversaries if there exist probabilistic polynomial-time simulators S_1, S_2 such that:

$$S_1(1^\lambda, 1^\kappa, \mathbf{x}, \{f(x_t)\}_{t\in\{1,\ldots,Q\}}) \stackrel{c}{\equiv} \text{VIEW}_1^{\Pi}(1^\lambda, 1^\kappa, \mathbf{x}, f) \tag{2}$$

$$S_2(1^\lambda, 1^\kappa, f) \stackrel{c}{\equiv} \text{VIEW}_2^{\Pi}(1^\lambda, 1^\kappa, \mathbf{x}, f) \tag{3}$$

where f is a function that maps $\{0,1\}^\ell$ to $\{0,1\}^{\ell'}$.

Remark. It is important to emphasize that the views VIEW_1^{Π} and VIEW_2^{Π} encompass the **entire life-cycle** of the protocol, including the one-time offline preprocessing phase and the sequence of Q adaptive online queries. By defining security over the joint distribution of all phases, we ensure that the server's function privacy and the client's input privacy are maintained across the total execution, rather than just in an isolated query.

5 PPFE-LUT Construction

Both PIR and PFE protocols for lookup tables enable a client to query data from a server-side database while preserving privacy guarantees. Preprocessing PIR protocols, as shown in prior works [35**?**], achieve notable efficiency gains. Inspired by this, we propose an efficient PPFE protocol incorporating similar preprocessing techniques.

5.1 Notation and System Parameters

In this section, we specify the system parameters and notations used in our concrete construction.

General Parameters. Let λ denote the computational security parameter and κ denote the statistical security parameter. We use $\alpha(\kappa)$ to denote an arbitrarily small super-constant function. Let $f : \{0,1\}^\ell \to \{0,1\}^{\ell'}$ be the private function, represented as a lookup table T of size $n = 2^\ell$. Let N denote the polynomial dimension of the homomorphic encryption scheme, which represents the number of elements per row when using batching techniques.

RMS-based Parameters. Following the underlying RMS framework [35], we view the database as consisting of n/N rows (chunks), each of size N. We define the number of hints required for the protocol as follows:

- **Primary Hints (M_1):** The number of primary hints is set to $M_1 = O(N) \log \kappa \cdot \alpha(\kappa)$.
- **Backup Hints (M_2):** The number of backup hints is set to $M_2 = C \cdot M_1$, where C is a constant (set to $C = 0.5$ in our implementation).

Data Structures. The client maintains a local state consisting of the following hint structures:

- **Primary Hint Tuple:** Each primary hint is stored as a tuple $(j, \hat{v}_j, e_j, b_j, \tilde{p}_j)$, where:
  - j: A unique hint ID.
  - $\hat{v}_j$: The cutoff median value used for chunk selection.
  - e_j: An extra index used for the hint set in the primary hints.
  - $\tilde{p}_j$: The encrypted parity (ciphertext) of the hint set.
  - Note: The first three components $(j, \hat{v}_j, e_j)$ of the primary hint tuple serve as a compact representation of the pseudorandom hint set S. This design allows the client to deterministically reconstruct the large hint set S from these few parameters, significantly reducing client storage requirements.
- **Backup Hint Tuple:** Each backup hint is stored as $(j, \hat{v}_j, \tilde{p}_j, \tilde{p}'_j)$, containing two encrypted parities corresponding to the pair of backup hint sets used for refreshing.

5.2 Oblivious Decryption Protocol

As outlined in Section 1.3, we design an *oblivious decryption* protocol that decomposes BFV decryption into a linear step $\text{DecLin}_{\text{sk}}(\cdot)$ and a nonlinear final-decryption step $\text{DecNonlin}(\cdot)$. The key property enabling our construction is the linearity of DecLin: for any ciphertext $\mathbf{ct}'$ and mask $r \in \mathcal{R}_q$,

$$\text{DecNonlin}(\text{DecLin}_{\text{sk}}(\mathbf{ct}' + (r, 0)) - r) = \text{DecNonlin}(\text{DecLin}_{\text{sk}}(\mathbf{ct}')).$$

This allows P_1 to mask the ciphertext before sending it to P_2, and then remove the mask after receiving the partial decryption result without affecting correctness. Note that in our actual protocol execution, while the addition of $\tilde{0}$ and e_{ans} introduces extra noise terms into the linear partial decryption result, the non-linear final decryption step (DecNonlin) successfully removes them as long as the total noise remains within the decryption budget

We now present the concrete protocol instantiated with the BFV scheme. In this setting, P_1 holds a ciphertext $\mathbf{ct}$ (with plaintext m and noise e), pk, and public parameters $pp = (N, \chi_{\text{key}}, \chi_{\text{err}}, \chi_{\text{enc}}, p, q)$; P_2 holds the private key sk and pp. The goal is for P_1 to obtain m while P_1 learns nothing about sk and P_2 learns nothing about $\mathbf{ct}$, m or e.

Here, we denote the components of $\mathbf{ct}$ by $(b, a) = (-a \cdot sk + \lfloor q/p \rfloor m + e, a)$. The public key is denoted as $pk = (b_p, a_p) = (-a_p \cdot sk + e_p, a_p)$.

Oblivious Decryption Protocol

(1) **Ciphertext masking:** P_1 samples a random mask $r \in \mathcal{R}_q$ and generates a ciphertext $\tilde{0}$ of 0. It computes $\mathbf{ct}' = (\mathbf{ct} \boxplus$

$\tilde{0}) + (r, 0)$ and sends $\mathbf{ct}'$ to P_2. Suppose that the ciphertext $\mathbf{ct}'$ has the form $(b_q, a_q) = (b + b_p u_q + e'_q + r, a + a_p u_q + e_q)$, where $u_q, e_q, e'_q \leftarrow \chi_{err}$.

(2) **Partial decryption with noise:** P_2 performs the linear decryption step and adds a small perturbation noise: $\mathsf{out}_2 = \mathsf{DecLin}_{\mathsf{sk}}(\mathbf{ct}') + e_{ans}$, where $e_{ans} \leftarrow \chi^N_{\mathsf{ans}}$. P_2 sends out_2 to P_1.

(3) **Unmasking and final decryption:** P_1 removes the mask r and applies the nonlinear step: $m = \mathsf{DecNonlin}(\mathsf{out}_2 - r)$.

Correctness: By the linearity of DecLin, we have $\mathsf{out}_2 = \mathsf{DecLin}_{\mathsf{sk}}(\mathbf{ct}') + e_{\mathsf{ans}} = b_q + a_q \cdot sk + e_{\mathsf{ans}}$. After unmasking, $\mathsf{out}_2 - r = \lfloor q/p \rfloor m + e + u_q e_p + e'_q + e_q \cdot sk + e_{ans}$. When $|e + u_q e_p + e'_q + e_q \cdot sk + e_{ans}| < \frac{q}{2p}$, $\mathsf{DecNonlin}(\mathsf{out}_2 - r)$ outputs the correct plaintext m.

Privacy and Security: The protocol ensures privacy for both parties. For P_2, the ciphertext $\mathbf{ct}'$ received is masked by r and a ciphertext of 0, so P_2 cannot distinguish the noise distribution of the original ciphertext from a random ciphertext, and thus cannot obtain any information about the original ciphertext $\mathbf{ct}$ or the noise e embedded in it. For P_1, the partial decryption result out_2 is protected by the perturbation noise e_{ans}. Specifically, the unmasked value $\mathsf{out}_2 - r$ can be expressed as:

$$\mathsf{out}_2 - r = \lfloor q/p \rfloor m + e'_q + \langle \gamma, \mathbf{r} \rangle + e_{ans}$$

where $\mathbf{r} = (sk, e_p, e)$ is the secret vector and $\gamma = (e_q, u_q, 1)$ is the challenge vector sampled by P_1. Unlike traditional noise flooding which requires a massive e_{ans} to statistically drown out the secret-dependent terms, our protocol relies on the **Adaptive Hint-MLWE assumption**. This assumption asserts that even if an adversary (P_1) chooses the challenge γ, the secret sk remains computationally hidden, and the 'hints' (partial decryption results) do not help in distinguishing the BFV ciphertexts from random strings. This ensures that a small e_{ans} is sufficient to prevent P_1 from recovering the server's private key. (See Section 5.5 for the formal reduction).

5.3 The Construction of PPFE

In this section, we present the detailed construction of our PPFE scheme, which synergistically integrates the RMS PIR protocol [35] with the BFV homomorphic encryption scheme [3, 13]. This construction leverages batching techniques to optimize performance and incorporates an oblivious decryption mechanism to ensure security for both parties.

Preprocessed PFE Protocol

System Setup. The server holds a database table T with input space $[n]$, and the client intends to compute $T(x)$ for multiple inputs x. The protocol is instantiated using the symmetric BFV scheme. The server executes $\mathsf{Setup}(1^\lambda)$ to generate public parameters pp, which are shared with the client. Subsequently, the server runs $\mathsf{KeyGen}(pp)$ to generate a secret key sk and a public key $pk = (b_p, a_p) = (-a_p \cdot sk + e_p, a_p)$. The homomorphic encryption scheme operates over a plaintext space $\mathbb{Z}_p^N$ and a ciphertext space $\mathcal{R}_q^2$, where N denotes the polynomial ring dimension.

5.3.1 Phase 1: Offline Preprocessing. The offline phase prepares the encrypted database and constructs efficient hint structures, enabling the client to perform sub-linear online queries.

(1) **Table Encoding.** To exploit BFV batching capabilities, the server organizes the function values $\{f(i)\}_{i\in[n]}$ into an $n/N \times N$ matrix $\mathbf{T} \in \mathbb{Z}_p^{n/N \times N}$. Each row $k \in [n/N]$ represents a batch of N consecutive values, encoded as the coefficients of a single polynomial in $\mathcal{R}_p$.

(2) **Database Encryption.** The server encrypts each row k of the matrix $\mathbf{T}$ individually. Specifically, it computes $\tilde{\mathbf{T}}[k] \leftarrow \mathsf{Enc}(\mathbf{T}[k])$, where $\mathbf{T}[k]$ is the polynomial encoding of the k-th row. The resulting collection of n/N ciphertexts is transmitted to the client. We explicitly represent $\tilde{\mathbf{T}}[k]$ as (b_k, a_k), where $b_k = -a_k \cdot sk + \lfloor q/p \rfloor \mathbf{T}[k] + e_{ct_k}$.

(3) **Client Key Generation.** To efficiently manage large index sets without explicit storage, the client adopts PRF-based hint representations following the RMS framework. The client samples a PRF key $K_{\text{hint}} \in \{0,1\}^\lambda$. For brevity, we denote $\mathsf{PRF}_{K_{\text{hint}}}(\cdot)$ as $\mathsf{PRF}(\cdot)$.

(4) **Hint Initialization.** The client initializes data structures for both primary and backup hints to support multiple queries with replenishment:

(a) **ID Assignment:** Assign primary hint IDs $\{0, \ldots, M_1 - 1\}$ and backup hint IDs $\{M_1, \ldots, M_1 + M_2 - 1\}$.

(b) **Hint Set Generation:** [3] For each hint $j \in [M_1 + M_2]$, the client generates a vector of random values $V_j = [v_{j,0}, \ldots, v_{j,n/N-1}]$ where $v_{j,k} = \mathsf{PRF}(\text{"select"} \parallel j \parallel k)$. It computes the median $\hat{v}_j$ of V_j to serve as the selection threshold, i.e., for k-th row, if $v_{j,k} < \hat{v_j}$, then the hint set with ID j selects an index (with offset $r_{j,k} = \mathsf{PRF}(\text{"offset"} \parallel j \parallel k)$) within the k-th row. For primary hints ($j < M_1$), an extra index e_j is uniformly sampled from the indices whose rows are not selected by the threshold logic.

(c) **Parity Initialization:** Initialize primary hint parities $\{\tilde{p}_j\}_{j\in[M_1]}$ and backup hint parities $\{\tilde{p}_j, \tilde{p}'_j\}_{j\in\{M_1,\ldots,M_1+M_2-1\}}$ as trivial ciphertexts of zero (i.e., $(0,0) \in \mathcal{R}_q^2$).

(5) **Streaming Processing.** To minimize client-side storage, the client processes the encrypted database in a streaming fashion. Upon receiving the k-th ciphertext $\tilde{\mathbf{T}}[k]$:

- **Update Primary Hints:** For each $j \in [M_1]$, if $v_{j,k} < \hat{v}_j$, the client homomorphically rotates $\tilde{\mathbf{T}}[k]$ by an offset $r_{j,k} = \mathsf{PRF}(\text{"offset"} \parallel j \parallel k)$ and aggregates it into $\tilde{p}_j$. If the row k contains the extra index e_j, the corresponding element is also rotated and added to $\tilde{p}_j$.
- **Update Backup Hints:** For each backup hint $j \in \{M_1, \ldots, M_1 + M_2 - 1\}$, the client updates either $\tilde{p}_j$ or $\tilde{p}'_j$ based on the comparison between $v_{j,k}$ and $\hat{v}_j$, effectively maintaining two complementary partial sums.
- **Cleanup:** The ciphertext $\tilde{\mathbf{T}}[k]$ is discarded immediately after processing to conserve memory.

Remark: The homomorphic rotations align the target elements to the constant term of the polynomial. Consequently, the constant term of the decrypted parity polynomial corresponds to the sum of the selected database values.

[3] Each primary hint set is conceptually formed by selecting one random index from each of $n/2N + 1$ distinct rows. Each backup hint consists of two complementary sets (S_0, S_1). Conceptually, the n/N rows are partitioned into two equal-sized groups; S_0 and S_1 are formed by selecting one random index from each row in the first and second group, respectively.

(6) **State Storage.** The client stores the primary hints $H = \{(j, \hat{v}_j, e_j, \tilde{p}_j)\}_{j\in[M_1]}$ and backup hints

$$H' = \{(j, \hat{v}_j, \tilde{p}_j, \tilde{p}'_j)\}_{j\in\{M_1,\ldots,M_1+M_2-1\}}.$$

Additionally, a bit vector $\{b_j\}$ is initialized to zeros to track 'whether to redefine '<' as '>' in row selection for hint j' during hint replenishment.

Post-Preprocessing State. The client's state is $\mathrm{st}_1^{(0)} = (K_{\mathrm{hint}}, H, H')$, and the server's state is $\mathrm{st}_2^{(0)} = (f, \mathrm{sk})$. The public parameters pp serve as the CRS.

5.3.2 Phase 2: Online Querying. For the t-th query ($t = 1, \ldots, Q$), with client input x_t and current states $\mathrm{st}_1^{(t-1)}$ and $\mathrm{st}_2^{(t-1)}$, the protocol proceeds as follows:

(1) **Hint Selection.** The client searches for a primary hint $j \in H$ that covers the target index x_t. A hint j covers x_t if $x_t = e_j$ or if the row containing x_t (denoted c_{x_t}) satisfies the selection criteria defined by $v_{j,c_{x_t}} < \hat{v}_j$ (adjusted by b_j) and the offset matches. We denote the chosen hint set as S_t, and we denote the corresponding (encrypted) parity as $\tilde{p}_j = (b_{S_t}, a_{S_t})$.

(2) **Query Formulation.**
- **Set Construction:** If a valid hint j is found, the client reconstructs the hint's index set S_t using the PRF and removes x_t from it, such that $S'_t = S_t \setminus \{x_t\}$. A dummy set S^*_t is also constructed by selecting a random index from each row not covered by S'_t. If no hint is found, the client generates two random dummy sets S'_t and S^*_t by first partitioning the n/N rows into two equal-sized groups; then S'_t and S^*_t are formed by selecting one random index from each row in the first and second group, respectively.
- **Ciphertext Masking:** To secure the interactive decryption, the client prepares a masked ciphertext $\tilde{\beta}''_t$. It samples a random masking polynomial $r_t \leftarrow \mathcal{U}(\mathcal{R}_q)$ and computes $\tilde{\beta}''_t = (\tilde{p}_j \boxplus \tilde{0}) + (r_t, 0)$, where $\tilde{0}$ is a fresh encryption of zero. Specifically, the client computes $\tilde{\beta}''_t := (b_{q_t}, a_{q_t}) = (b_{S_t} + b_p u_{q_t} + e'_{q_t} + r_t, a_{S_t} + a_p u_{q_t} + e_{q_t})$, where $u_{q_t}, e_{q_t}, e'_{q_t}$ are freshly sampled from χ_{err}.
- **Transmission:** The client sends the query $((S_{t,0}, S_{t,1}), \tilde{\beta}''_t)$ to the server, where $(S_{t,0}, S_{t,1})$ is a random permutation of (S'_t, S^*_t).

(3) **Server Computation.** Upon receiving $((S'_t, S^*_t), \tilde{\beta}''_t)$, the server performs the following for $i \in \{0, 1\}$:
- **Partial Decryption:** It computes the noisy linear decryption $\overline{\beta''_t} = \mathsf{DecLin}_{\mathsf{sk}}(\tilde{\beta}''_t) \in \mathcal{R}_q$.
- **Response Generation:** The server calculates $\overline{\beta'_{t,i}} = \overline{\beta''_t}[0] - q_{t,i} + e_{\mathsf{ans}_t} \pmod q$, where $q_{t,i} = \lfloor \frac{q}{p} \sum_{k\in S_{t,i}} f(k) \rceil$ scales the plaintext sum to the ciphertext modulus, and $e_{\mathsf{ans}_t} \leftarrow \chi_{\mathsf{ans}}$ is a fresh noise term.
- **Oblivious Transfer:** To prevent the client from learning the parity of the dummy set in addition to the desired response, the server returns the two candidate responses through a 1-out-of-2 OT. This differs from the original RMS PIR protocol, where returning both responses is safe because RMS does not aim to hide the database from the client; in PPFE-LUT, revealing the dummy-set evaluation would violate table privacy.

(4) **Result Extraction.** The client extracts the function value by removing the mask: $\beta_t = \mathrm{DecNonlin}(\overline{\beta'_t} - r'_t[0]) \pmod p$, where $r'_t[0]$ is the constant term of the masking polynomial r_t. If the query was successful, the client outputs $y_t = \beta_t$.

(5) **Hint Replenishment.** To sustain query capability, the client promotes a backup hint to a primary hint. It selects the next available backup hint J, updates its parity with the retrieved value y_t (computing $\tilde{p}_J \boxplus y_t$ or $\tilde{p}'_J \boxplus y_t$ depending on whether c_{x_t} does not fall in the left or right set), and adds it to the primary hint set H. The used backup hint is removed from H'.

Continuous Operation. One preprocessing phase supplies $M_2 = \widetilde{O}(\sqrt{n})$ backup hints and therefore supports that many online queries. Before these backup hints are exhausted, the client begins a background refresh: it re-downloads the encrypted database and computes only the partial sums needed for a new, disjoint set of backup hints. The existing primary hints and long-term keys remain in place, so this is not a full setup restart; nevertheless, downloading the encrypted table is necessary because new backup parities depend on the table. Pipelining this refresh with the remaining online queries provides continuous service when the background bandwidth is available.

5.4 Correctness Analysis

The PPFE protocol can be seen as an encrypted database variant of the RMS scheme with added masking/unmasking steps. Thus, we derive the following correctness lemma.

Lemma 5.1 (Correctness). *Assume n is bounded by $\mathrm{poly}(\lambda)$ and $\mathrm{poly}(\kappa)$, and set the parameters of the BFV scheme such that it can support the homomorphic operations of the protocol. Let $\alpha(\kappa)$ be any super-constant function, i.e., $\alpha(\kappa) = \omega(1)$. Let N denote the polynomial dimension of the BFV scheme. Assume that $N/\sqrt{n} = O(1)$, Set $M_1 = O(N)\ln\kappa\alpha(\kappa)$ and $M_2 = 0.5 \cdot O(N)\ln\kappa\alpha(\kappa)$. For all $Q = N\ln\kappa\alpha(\kappa)$ $(Q \le M_2)$ queries, there exists a probability of at least $1 - \mathrm{negl}(\lambda) - \mathrm{negl}(\kappa)$ to correctly respond to all queries for some negligible function $\mathrm{negl}(\cdot)$.*

The proof follows the same structure as in the RMS scheme. We assume the client uses a true random oracle to sample sets rather than a PRF. Due to the PRF pseudorandomness, this assumption has a negligible impact on the failure probability. There are only two types of failure events:

The oblivious decryption protocol fails to decrypt the ciphertext correctly, which happens with negligible chance $\mathrm{negl}(\lambda)$, due to the noise in the masked ciphertext being larger than the threshold $\frac{q}{2p}$. **The client cannot find a set containing query index x_t.** To prevent the first type of failure, we set the BFV parameters such that the noise budget is sufficient to accommodate all homomorphic operations and the perturbation noise e_{ans}. Then, Lemma 1 in the RMS scheme has established the following fact: Each set contains the query with probability more than $1/2N$ (since there are n/N rows, and each set selects $n/2N + 1$ of them). Moreover, when replacing the PRF with a true random oracle, the M_1 hint sets in the primary hint can be considered independent. Therefore, the probability that a specific query x is not included in any set is

Client (P_1)		**Server** (P_2)
Phase 1: Offline		
Gen $K_{\text{hint}}, K_{\text{mask}}$		Setup (pk, sk), pp
Init Hints H, H'		Matrix $\mathbf{T} \in \mathbb{Z}_p^{n/N \times N}$
	$\xleftarrow[\text{for } k=0,\ldots,n/N-1]{\widetilde{\mathbf{T}}[k]=\text{Enc}(\mathbf{T}[k])}$	Encrypt Rows
Update H, H' via homomorphic rot/add		
Store $\text{st}_1^{(0)} = (K_{\text{hint}}, H, H')$		Store $\text{st}_2^{(0)} = (f, sk)$
Phase 2: Online (Query x_t)		
1. Set Finding:		
Find hint j covering x_t		
2. Query Gen:		
Generate corresponding hint set S_t $\quad S'_t \leftarrow S_t \setminus \{x_t\}$		
$S_t^* \leftarrow$ Dummy Set		
Sample mask $r_t \in \mathcal{R}_q$		
$\widetilde{\beta}''_t = (\widetilde{p}_j \boxplus \text{Enc}(0)) + (r_t, 0)$		
Permute $(S'_t, S_t^*) \to (S_{t,0}, S_{t,1})$	$\xrightarrow{((S_{t,0},S_{t,1}),\widetilde{\beta}''_t)}$	
3. Server Comp:		
		$\overline{\beta''_t} = \text{DecLin}_{sk}(\widetilde{\beta}''_t)$
		For $i \in \{0,1\}$:
		$q_{t,i} = \lfloor \frac{q}{p} \sum_{k \in S_{t,i}} f(k) \rceil$
		$\overline{\beta'_{t,i}} = \overline{\beta''_t}[0] - q_{t,i} + e_{\text{ans}_t}$
4. Oblivious Transfer:		
Select σ s.t. $S_{t,\sigma} = S'_t$	$\xleftarrow[\text{OT Response}]{\overline{\beta'_t}=\overline{\beta'_{t,\sigma}}}$	Run $\text{OT}(\overline{\beta'_{t,0}}, \overline{\beta'_{t,1}})$
5. Response Processing:		
$r'_t = \text{const_term}(r_t)$		
$\beta_t = \text{DecNonlin}(\overline{\beta'_t} - r'_t)$		
$y_t = \beta_t \bmod p$		
6. Refresh:		
Update H using H', y_t, x_t		

Figure 2: Preprocessed Private Function Evaluation (PPFE) Protocol

bounded by:

$$\left(1 - \frac{1}{2N}\right)^{M_1} = \left(1 - \frac{1}{2N}\right)^{O(N)\ln\kappa\cdot\alpha(\kappa)} \leq (1/e)^{\ln\kappa\alpha(\kappa)/2} = \kappa^{-\alpha(\kappa)/2}$$

With the union bound, for all $Q = N\ln\kappa\alpha(\kappa)$ queries(where $Q \leq M_2$), the probability of any query cannot find a set is bounded by $N\ln\kappa\alpha(\kappa)\kappa^{-\alpha(\kappa)/2}$, which is negligible in κ since N is bounded by $poly(\kappa)$.

5.5 Security Analysis

We prove that our PPFE-LUT construction from Section 5.3 is a secure PPFE-LUT protocol for semi-honest adversaries in the $\mathcal{F}_{\text{OT}}$-hybrid model. In our analysis, the 'client' represents party P_1, who holds input 1^λ and the query indices $\mathbf{x} = \{x_1, x_2, \ldots, x_Q\}$, while the 'server' represents party P_2, who holds input 1^λ and the function f.

We note that establishing security in the $\mathcal{F}_{\text{OT}}$-hybrid model is sufficient, as replacing the ideal functionality with a secure OT protocol preserves security in the real model. Specifically, the total adversarial advantage is at most the advantage in the hybrid model plus the advantage against the underlying OT instantiation.

Theorem 5.2 (Security of PPFE-LUT in the Hybrid Model). *Assuming the hardness of the A-HintMLWE problem, the hardness of decision-RLWE problem and the pseudorandomness of the Pseudorandom Function (PRF), our protocol Π described in Section 5.3 is a secure PPFE-LUT protocol in the $\mathcal{F}_{OT}$-hybrid model against semi-honest adversaries.*

We prove this theorem by establishing the following two claims that demonstrate the security against both parties.

Theorem 5.3 (Server Security). *Under the same assumptions as in Theorem 5.2, there exists a probabilistic polynomial-time (PPT) simulator S_1 in the $\mathcal{F}_{OT}$-hybrid model such that*

$$S_1(1^\lambda, 1^\kappa, \mathbf{x}, \{f(x_t)\}_{t\in\{1,2,\ldots,Q\}}) \overset{c}{\equiv} VIEW_1^\Pi(1^\lambda, 1^\kappa, \mathbf{x}, f)$$

Specifically, for every PPT adversary $\mathcal{A}$ that distinguishes the simulated view from the real view with advantage $Adv^{\mathcal{A}}_{\Pi,S_1}(\lambda)$, there exists a PPT adversary $\mathcal{B}$ against the Adaptive Hint-MLWE problem such that:

$$Adv^{\mathcal{A}}_{\Pi,S_1}(\lambda) \leq Adv^{\mathcal{B}}_{A\text{-}HintMLWE}(\lambda) + negl(\lambda)$$

Proof Intuition.

The security of the server relies on ensuring that the client's view—comprising the public key pk, the encrypted database $\{\tilde{m}_k\}$, and the decryption transcripts $\{\overline{\beta'_t}\}$—leaks no information about the function f beyond the legitimate outputs $\{f(x_t)\}$. The core of our proof lies in observing that the partial decryption result $\overline{\beta'_t}$ (after removing the client-side mask r_t) reveals a linear leakage of the server's secret vector $\mathbf{r} = (sk, e_p, e_0, \dots)$. Specifically, this leakage is structured as an inner product $[\langle \gamma_t, \mathbf{r} \rangle]_0 + e_{ans,t}$, where γ_t is a challenge vector derived from the client's query.

This structure maps directly to the **Adaptive Hint-MLWE** problem. We construct a reduction where the server's public parameters and database ciphertexts are treated as the Adaptive Hint-MLWE **secret** r, and the noisy decryption results are treated as the **hints**. The adaptivity of the query indices x_t is captured by the adversary's ability to choose γ_t. Under the Adaptive Hint-MLWE assumption, even with access to these hints, the adversary cannot distinguish the valid BFV public key and ciphertexts from uniform random elements. Thus, the real view is computationally indistinguishable from a simulated view where the database is replaced by random noise, ensuring function privacy. The formal proof is provided in Appendix B.3.

Theorem 5.4 (Client Security). *Under the same assumptions as in Theorem 5.2, there exists a probabilistic polynomial-time (PPT) simulator S_2 in the $\mathcal{F}_{OT}$-hybrid model such that*

$$S_2(1^\lambda, 1^\kappa, f) \stackrel{c}{\equiv} VIEW_2^{\Pi}(1^\lambda, 1^\kappa, \mathbf{x}, f)$$

Specifically, for every PPT adversary $\mathcal{A}$ that distinguishes the simulated view from the real view with advantage $Adv^{\mathcal{A}}_{\Pi,S_2}$, the advantage is bounded by

$$Adv^{\mathcal{A}}_{\Pi,S_2} \leq Adv_{PRF}(\lambda) + Q \cdot Adv_{RLWE}(\lambda) + negl(\lambda)$$

where $Adv_{PRF}(\lambda)$ and $Adv_{RLWE}(\lambda)$ denote the maximum advantages of any PPT adversary against the PRF and the Decision-RLWE problem, respectively. Q is the total number of online queries, and is bounded by $poly(\lambda)$.

Proof Intuition. The client security ensures that the server learns nothing about the client's input $\mathbf{x}$. The server's view consists of the query sets (indices) and the associated ciphertexts. Our proof relies on two main pillars: the pseudorandomness of the query set construction and the semantic security of the RLWE encryption. First, for the query sets, the protocol mixes the "real" hint set (containing the target index) with a "dummy" set. Due to the properties of the underlying RMS scheme [35], these two sets, when combined and randomly permuted, appear as a uniform random partition of the database indices, statistically hiding which set is real. Second, for the ciphertexts sent by the client, we use a hybrid argument to show that they are computationally indistinguishable from random 'ciphertexts'. This is guaranteed by the Decision-RLWE assumption. Therefore, we can construct a simulator that generates random query sets and random ciphertexts without knowing $\mathbf{x}$. Since the server cannot distinguish this simulated random view from the real protocol execution, the client's input remains private. The formal proof is provided in Appendix B.4.

Combining the two theorems above, we establish that our PPFE-LUT protocol satisfies the security requirements of Definition 4.2, completing the proof of Theorem 5.2.

6 Evaluation

We evaluate the efficiency and practicality of our Preprocessed Private Function Evaluation (PPFE) scheme, focusing on computational and communication costs across varying table sizes and network settings. We compare our scheme against four state-of-the-art baselines: SP-LUT [11], FLORAM [12], 2P-DUORAM [39], and FABLE [38]. We provide the complete implementation at https://anonymous.4open.science/r/PPFE-E178/ for reproducibility.

6.1 Experimental Setup

Implementation & Environment. All experiments were conducted on an Alibaba Cloud server equipped with Intel® Xeon® Cooper Lake processors (3.3 GHz) and 128 GB RAM, running Ubuntu 24.04. The execution is single-threaded. Our PPFE implementation builds upon the RMS scheme [35] using C++ with the Troy-Nova library for BFV operations[13]. Baseline schemes were adapted from their official repositories (SP-LUT derived from FABLE). We simulated network conditions using Linux `tc-netem`: **LAN** (3 Gbps, 0.8 ms RTT) and **WAN** (200 Mbps, 40 ms RTT).

Parameter Selection. Table 2 summarizes our parameter configuration. We configure BFV for 128-bit computational security ($\lambda = 128$). The plaintext modulus $p = 2^{20}$ supports 20-bit function outputs, while the ciphertext modulus $q \approx 2^{54}$ ensures correctness. The polynomial degree N is selected as 2048 or 4096 depending on the LUT size. For RMS, we set statistical security $\kappa = 40$.

Table 2: Parameter Configuration Summary.

Category	Parameter	Value/Range
HE	Security Level (λ)	128 bits
	Plaintext Modulus (p)	2^{20}
	Ciphertext Modulus (q)	$\approx 2^{54}$
	Polynomial Degree (N)	2048 / 4096
	Key/Error Dist. (χ_{key}, χ_{err})	$\mathcal{D}_{\mathbb{Z}^N,\sigma\approx 16.4}$
RMS	Statistical Security (κ)	40 bits
	Primary Hints (M_1)	$\sqrt{n}\ln\kappa \cdot \omega(1)$
	Backup Hints (M_2)	$0.5\sqrt{n}\ln\kappa \cdot \omega(1)$
Masking	Mask Polynomial (r)	$U(\mathcal{R}_q)$
	Answer Noise (χ_{ans})	$\mathcal{D}_{\mathbb{Z},\sigma_{\text{ans}}=22}$

6.2 Performance of PPFE

We first analyze the detailed performance breakdown of PPFE. Table 3 presents the amortized costs for each protocol phase across different LUT sizes.

Amortization, adaptivity, and offline cost. Throughout this section, "amortized" has a specific meaning: we divide the one-time offline hint-generation cost by the number of online queries supported by that preprocessing phase. It does *not* mean that PPFE-LUT batches online queries. Each online measurement executes an individual lookup, and query $t+1$ may be chosen after learning the result of previous queries. In contrast, FABLE's reported amortized online performance uses a batch of 1024 queries; this advantage is unavailable for such an adaptive sequence.

Table 3: Comparison of Metrics for Different LUT Sizes. The performance metrics in this table represent amortized costs per query, where offline phase values (time and communication) are calculated by dividing the total protocol execution cost of a single offline phase—which supports a max of $Q = \sqrt{n} \ln \kappa \alpha(\kappa)$ queries—by this max query count Q, while online phase values are derived from executing $\sqrt{n}$ queries and averaging results per query. Client Time aggregates the client's computation time (green) and communication time (red), Server Time combines the server's computation time (blue) with communication time (red), and End-to-End time represents the complete protocol duration across all 9 phases.

	LUT Size	2^{18}			2^{20}			2^{22}		
	Metric	Amortized Time (ms)		Amortized Comm. (KB)	Amortized Time (ms)		Amortized Comm. (KB)	Amortized Time (ms)		Amortized Comm. (KB)
	Network	LAN	WAN		LAN	WAN		LAN	WAN	
Offline	(Server) LUT Encryption	0.09	0.09	-	0.09	0.09	-	0.09	0.09	-
	(Both) Preprocessing Comm.	0.01	0.34	2.07	0.01	0.33	2.07	0.01	0.33	2.07
	(Client) Hint Generation	15.66	15.28	-	25.69	25.31	-	46.01	45.10	-
	Total Offline Phase	15.76	15.71	2.07	25.79	25.73	2.07	46.11	45.52	2.07
Online	(Client) Query Generation	0.06	0.07	-	0.11	0.12	-	0.20	0.22	-
	(Both) Query Comm.	0.04	0.07	34.19	0.04	0.07	36.25	0.05	0.08	40.37
	(Server) Query Response	0.01	0.01	-	0.01	0.01	-	0.26	0.01	-
	(Both) Query Response	2.38	82.50	0.12	2.40	82.55	0.12	2.44	82.76	0.12
	(Client) Query Result Gen.	0.00	0.00	-	0.00	0.00	-	0.00	0.00	-
	(Client) Refresh Phase	0.02	0.03	-	0.02	0.04	-	0.04	0.05	-
	Total Online Phase	2.49	82.68	34.31	2.58	82.79	36.37	2.96	83.21	40.49
	Client Time	18.17	98.29	-	28.27	108.42	-	48.77	128.54	-
	Server Time	2.52	83.01	-	2.54	83.04	-	2.56	83.26	-
	End-to-End	18.27	98.39	36.38	28.37	108.52	38.44	49.07	128.73	42.56

PPFE-LUT deliberately trades heavier offline work for a fast adaptive online path. Table 3 reports this cost amortized over the supported queries, while the comparison figures report only online costs.

As shown in Fig. 3 and Table 3, the amortized online query time remains remarkably stable regardless of the LUT size. In LAN settings, query time stays around 2.5 ms, while in WAN settings, it hovers around 83 ms. The pure computational overhead is minimal (< 0.3 ms for the server), confirming that network latency dominates the online phase. The client's offline hint generation time scales with $\sqrt{n}$, but this cost is amortized and performed prior to the time-critical query phase.

Communication & Storage. Fig. 4 illustrates the communication efficiency. As the LUT grows from 2^{10} to 2^{20}, communication increases only marginally from 32 KB to 36 KB. This is because the traffic is dominated by the fixed-size BFV ciphertext (approx. 25 KB for $N = 2048$). Additionally, for client storage, we optimize by converting stored RLWE hints (2 polynomials) into LWE ciphertexts (1 polynomial vector) after the offline phase, reducing storage requirements by approximately 50% without affecting functionality.

6.3 Comparison with State-of-the-Art

We compare PPFE with SP-LUT, FLORAM, 2P-DUORAM, and FABLE. All baselines are executed single-threaded. Notably, FABLE is evaluated using a batch size of 1024, whereas PPFE uses an adaptive (single-query) approach.

Computation Efficiency. As shown in Fig. 5, PPFE achieves the lowest amortized time per query across all LUT sizes. The advantage widens as n increases. At $n = 2^{24}$, PPFE is approximately 7.9× to 171× faster than FABLE (the fastest baseline). Even in WAN settings, PPFE maintains real-time performance (≈ 85 ms). This efficiency stems from our streamlined design: we avoid heavy primitives like FHE bootstrapping or Oblivious RAM read/write overheads, relying instead on lightweight homomorphic additions and PRFs.

Communication Efficiency. Fig. 6 highlights the decisive advantage of PPFE in bandwidth. For a 2^{20} size LUT, PPFE requires only 36 KB per query. This is merely 3.2% of FABLE (1.11 MB), 1.6% of 2P-DUORAM (2.19 MB), and 0.24% of FLORAM (14.59 MB). This superiority is attributed to our 2-round interaction pattern. By offloading computation to the server via AHE and aggregating results into a single response, we avoid the multi-round communication inherent in GC or DORAM-based approaches.

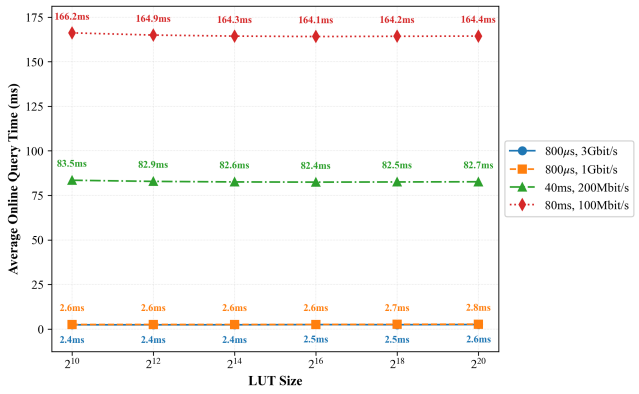

Figure 3: Amortized Query Time Under Different LUT Size and Different Network Environments. The computational overhead is minimal (<0.3 ms); so the network latency dominates the online phase.

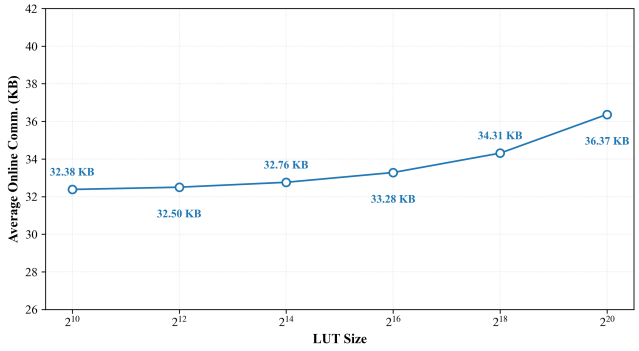

Figure 4: Amortized Online Communication Overhead Under Different LUT Size.

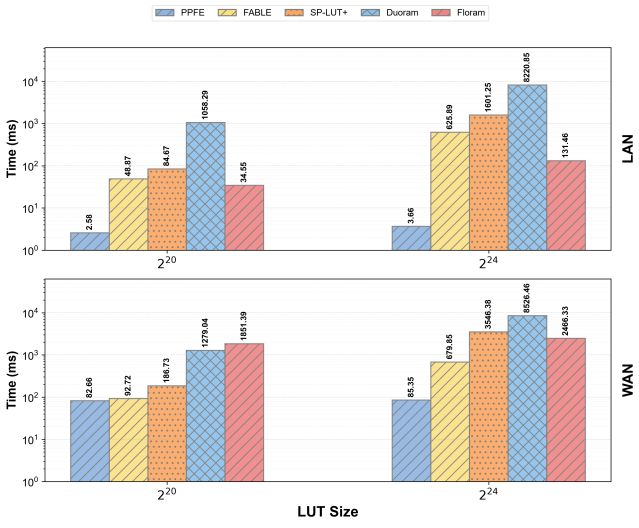

Figure 5: Computation Performance Comparison

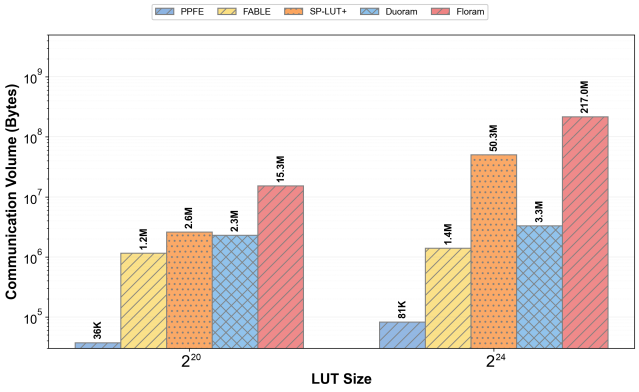

Figure 6: Communication Performance Comparison

7 Conclusion

In this paper, we presented **PPFE-LUT**, a preprocessing-based protocol for **private lookup table evaluation**. Functionally, PPFE-LUT realizes single-server SPIR for a private server table. By leveraging an encrypted database, it achieves sublinear online computation and communication with a lightweight client. Its core technical contribution is *oblivious decryption*: the server performs linear partial decryption, while the client performs the nonlinear final-decryption step. We formalized Adaptive Hint-MLWE and reduced it to standard MLWE, establishing that the partial-decryption artifacts remain secure even when queries are adaptively chosen.

Experimental evaluations confirm that PPFE-LUT substantially improves online performance for individual adaptive lookups. For a lookup table with 2^{24} entries, the online query processing time is **3 milliseconds**, providing more than an order-of-magnitude improvement over the evaluated baselines. Unlike approaches whose reported per-query costs rely on batching a set of queries known in advance, PPFE-LUT processes every online query individually; hence a later index may depend on an earlier result.

8 Ethical Considerations

This work focuses on the design and implementation of cryptographic protocols and does not involve human subjects, user studies, or the collection of personally identifiable information (PII). All experiments presented in this paper were conducted using synthetic datasets and standard benchmarks; therefore, no Institutional Review Board (IRB) approval was required.

Societal Impact. The primary goal of the proposed PPFE framework is to enhance privacy in sensitive digital interactions. By enabling secure lookup table evaluations, our work facilitates critical applications—such as privacy-preserving medical diagnostics and secure malware detection—where data confidentiality is paramount. We anticipate that this technology will empower individuals to utilize digital services without compromising their personal privacy, while simultaneously allowing organizations to protect their intellectual property.

Potential Misuse. We acknowledge that, like many cryptographic tools, privacy-preserving computation could theoretically be employed to conceal malicious queries. However, we believe that

the societal benefits of protecting sensitive user data (e.g., genomic information or financial status) and enabling secure collaboration significantly outweigh the generic risks associated with the dual-use nature of encryption technologies.

9 Generative AI Usage

In this work, we employed generative AI tools, including ChatGPT, Gemini, and Claude, primarily to support the refinement and presentation of the manuscript as well as certain aspects of code development. We stress that the **core conceptual ideas, theoretical analyses, and mathematical proofs** remain entirely the original contributions of the authors.

The scope of AI assistance can be summarized as follows:

- **Manuscript Writing and Refinement:** ChatGPT and Gemini were used to improve language clarity, enhance readability, and provide suggestions on structuring the manuscript, especially in the Introduction (Section 1).
- **Code Assistance:** Claude was employed to assist with code editing, debugging suggestions, and improving code readability. All actual implementation, testing, and analysis were carried out by the authors.
- **Presentation and Visualization:** AI provided advice on the flow of the Experiment section (Section 6), and Figure 3 was revised and redrawn based on an initial draft generated by Gemini.

All AI-assisted content, including text and code suggestions, was carefully reviewed, verified, and finalized by the authors to ensure correctness and consistency with the original research contributions.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2023YFB3106100; the National Cryptologic Science Fund of China under No. 2025NCSF01007; the National Natural Science Foundation of China Key Program under Grant No. 92270204; and the CAS Project for Young Scientists in Basic Research under Grant No. YSBR-035.

References

[1] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. 2009. Secure evaluation of private linear branching programs with medical applications. In *Computer Security–ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*. Springer, 424–439.

[2] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. 2019. nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM workshop on encrypted computing & applied homomorphic cryptography*. 45–56.

[3] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual cryptology conference*. Springer, 868–886.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 309–325.

[5] Justin Brickell, Donald E Porter, Vitaly Shmatikov, and Emmett Witchel. 2007. Privacy-preserving remote diagnostics. In *Proceedings of the 14th ACM conference on Computer and communications security*. 498–507.

[6] Andreas Brüggemann, Robin Hundt, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2023. FLUTE: fast and secure lookup table evaluations. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 515–533.

[7] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller. 2000. One-round secure computation and secure autonomous mobile agents. In *Automata, Languages and Programming: 27th International Colloquium, ICALP 2000 Geneva, Switzerland, July 9–15, 2000 Proceedings 27*. Springer, 512–523.

[8] Ran Canetti, Yuval Ishai, Ravi Kumar, Michael K Reiter, Ronitt Rubinfeld, and Rebecca N Wright. 2001. Selective private function evaluation with applications to private statistics. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. 293–304.

[9] Yan-Cheng Chang and Chi-Jen Lu. 2001. Oblivious polynomial evaluation and oblivious neural learning. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*. Springer, 369–384.

[10] Tung Chou and Claudio Orlandi. 2015. The simplest protocol for oblivious transfer. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America*. Springer, 40–58. Available at: https://eprint.iacr.org/2015/267.pdf.

[11] Ghada Dessouky, Farinaz Koushanfar, Ahmad-Reza Sadeghi, Thomas Schneider, Shaza Zeitouni, and Michael Zohner. 2017. Pushing the Communication Barrier in Secure Computation using Lookup Tables. In *Proceedings 2017 Network and Distributed System Security Symposium*. Internet Society.

[12] Jack Doerner and Abhi Shelat. 2017. Scaling ORAM for secure computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 523–535.

[13] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).

[14] Keith Frikken, Mikhail Atallah, and Jiangtao Li. 2004. Hidden access control policies with hidden credentials. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. 27–27.

[15] Keith B Frikken, Jiangtao Li, and Mikhail J Atallah. 2006. Trust Negotiation with Hidden Credentials, Hidden Policies, and Policy Cycles.. In *NDSS*, Vol. 6. 157–172.

[16] Craig Gentry, Shai Halevi, and Nigel P Smart. 2012. Homomorphic Evaluation of the AES Circuit. In *Annual Cryptology Conference*. Springer, 850–867.

[17] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*. PMLR, 201–210.

[18] Marco Holz, Ágnes Kiss, Deevashwer Rathee, and Thomas Schneider. 2020. Linear-complexity private function evaluation is practical. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*. Springer, 401–420.

[19] Alexander Hoover, Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. 2025. Plinko: Single-Server PIR with Efficient Updates via Invertible PRFs. In *Advances in Cryptology – EUROCRYPT 2025*, Serge Fehr and Pierre-Alain Fouque (Eds.). Springer Nature Switzerland, Cham, 3–33.

[20] Yuval Ishai and Anat Paskin. 2007. Evaluating branching programs on encrypted data. In *Theory of Cryptography Conference*. Springer, 575–594.

[21] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC.

[22] Jonathan Katz and Lior Malka. 2011. Constant-round private function evaluation with linear complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 556–571.

[23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. 2023. Toward practical lattice-based proof of knowledge from hint-MLWE. In *Annual International Cryptology Conference*. Springer, 549–580.

[24] Vladimir Kolesnikov and Thomas Schneider. 2008. A practical universal circuit construction and secure evaluation of private functions. In *International Conference on Financial Cryptography and Data Security*. Springer, 83–97.

[25] Shuaishuai Li, Liqiang Peng, Weiran Liu, Cong Zhang, Zhen Gu, and Dongdai Lin. 2025. BitBatSPIR: Efficient Batch Symmetric Private Information Retrieval From PSI. *IEEE Transactions on Dependable and Secure Computing* 22, 6 (2025), 6028–6039. doi:10.1109/TDSC.2025.3579365

[26] Chengyu Lin, Zeyu Liu, and Tal Malkin. 2022. XSPIR: Efficient Symmetrically Private Information Retrieval from Ring-LWE. In *Computer Security – ESORICS 2022*, Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng (Eds.). Springer International Publishing, Cham, 217–236.

[27] Yi Liu, Qi Wang, and Siu-Ming Yiu. 2022. Making private function evaluation safer, faster, and simpler. In *IACR International Conference on Public-Key Cryptography*. Springer, 349–378.

[28] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 1–23.

[29] Daniele Micciancio and Oded Regev. 2007. Worst-case to average-case reductions based on Gaussian measures. *SIAM journal on computing* 37, 1 (2007), 267–302.

[30] Payman Mohassel and Saeed Sadeghian. 2013. How to hide circuits in MPC an efficient framework for private function evaluation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 557–574.
[31] Hiraku Morita, Erik Pohle, Kunihiko Sadakane, Peter Scholl, Kazunari Tozawa, and Daniel Tschudi. 2024. MAESTRO: Multi-party AES using Lookup Tables. Cryptology ePrint Archive, Paper 2024/1317. https://eprint.iacr.org/2024/1317
[32] Salman Niksefat, Babak Sadeghiyan, Payman Mohassel, and Seyed Saeed Sadeghian. 2014. ZIDS: A Privacy-Preserving Intrusion Detection System Using Secure Two-Party Computation Protocols. *Comput. J.* 57, 12 (2014), 1832–1847.
[33] Annika Paus, Ahmad-Reza Sadeghi, and Thomas Schneider. 2009. Practical secure evaluation of semi-private functions. In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*. Springer, 89–106.
[34] Chris Peikert. 2010. An efficient and parallel Gaussian sampler for lattices. In *Annual Cryptology Conference*. Springer, 80–97.
[35] Ling Ren, Muhammad Haris Mughees, and I Sun. 2024. Simple and practical amortized sublinear private information retrieval using dummy subsets. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1420–1433.
[36] Ahmad-Reza Sadeghi and Thomas Schneider. 2008. Generalized universal circuits for secure evaluation of private functions with application to data classification. In *International Conference on Information Security and Cryptology*. Springer, 336–353.
[37] Tomas Sander, Adam Young, and Moti Yung. 1999. Non-interactive cryptocomputing for nc/sup 1. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 554–566.
[38] Zhengyuan Su, Qi Pang, Simon Beyzerov, and Wenting Zheng. 2025. FABLE: Batched Evaluation on Confidential Lookup Tables in 2PC. In *34nd USENIX Security Symposium (USENIX Security 25)*. 1945–1964.
[39] Adithya Vadapalli, Ryan Henry, and Ian Goldberg. 2023. Duoram: A {Bandwidth-Efficient} distributed {ORAM} for 2-and 3-party computation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 3907–3924.

A Tools

A.1 Pseudorandom Function (PRF)

A keyed function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ takes a key k and an input x, and is defined as $F_k(x) = F(k,x)$. Assume F is length-preserving, meaning $|F_k(x)| = |x| = |k|$. By randomly choosing a key $k \in \{0,1\}^\lambda$, we obtain a function F_k, whose behavior is compared against f_λ, a function uniformly chosen from the set of all random functions mapping λ-bit strings to λ-bit strings. If F_k is indistinguishable from f_λ for any polynomial-time distinguisher D, then F is called a pseudorandom function.

Definition A.1. [21] Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving keyed function. F is a pseudorandom function if, for any probabilistic polynomial-time distinguisher D, there exists a negligible function negl such that:

$$|\Pr[D^{F_k}(1^\lambda) = 1] - \Pr[D^{f_\lambda}(1^\lambda) = 1]| \leq \text{negl}(\lambda)$$

where $k \in \{0,1\}^\lambda$ is a randomly chosen key, and f_λ is uniformly chosen from the set of functions mapping λ-bit strings to λ-bit strings. In this case, F is called a pseudorandom function.

Generalization for Arbitrary Domains and Codomains. The foundational definition above describes a PRF mapping λ-bit strings to λ-bit strings. In practice, and in our protocol, we require PRFs with arbitrary input domains and specific output codomains. Such functions can be securely constructed from a standard PRF.

For a PRF with an arbitrary input space $\mathcal{X}$ (e.g., concatenated strings and integers), a standard construction is to apply a collision-resistant hash function H first, i.e., $F'_k(x) = F_k(H(x))$ for $x \in \mathcal{X}$, where F_k is a fixed-input-length PRF. For an output space that is a specific range, such as $[M]$ or $\mathbb{Z}_p$, a standard technique is to take the output of a PRF that produces a sufficiently long string and reduce it modulo M or p. While this may introduce a statistically negligible bias if the size of the original output range is not a multiple of M or p, its pseudorandomness is preserved computationally.

A.2 Oblivious Transfer

Oblivious Transfer (OT) is a fundamental cryptographic primitive that enables secure two-party computation. In this work, we model the 1-out-of-2 OT using the ideal functionality $\mathcal{F}_{\text{OT}}$.

Ideal Functionality $\mathcal{F}_{OT}$. The functionality interacts with a sender $\mathcal{S}$ and a receiver $\mathcal{R}$.

- **Inputs:** $\mathcal{S}$ sends two messages (M_0, M_1) (where $M_i \in \{0,1\}^\ell$) to $\mathcal{F}_{\text{OT}}$. $\mathcal{R}$ sends a selection bit $c \in \{0,1\}$ to $\mathcal{F}_{\text{OT}}$.
- **Outputs:** Upon receiving the inputs, $\mathcal{F}_{\text{OT}}$ sends M_c to $\mathcal{R}$ and outputs nothing to $\mathcal{S}$.

Our protocol construction and security analysis are presented in the $\mathcal{F}_{\text{OT}}$-hybrid model, assuming access to this ideal functionality.

Instantiation. For the concrete implementation, we instantiate $\mathcal{F}_{\text{OT}}$ using the efficient OT protocol proposed by Chou and Orlandi [10], which is based on a tweak of the Diffie-Hellman key exchange protocol. This protocol achieves high efficiency with only $2 + 3m$ full exponentiations for m OTs, making it suitable for practical applications. The protocol provides security against semi-honest adversaries and can be extended to achieve security against active adversaries through appropriate encoding and verification mechanisms.

A.3 The RMS PIR Protocol Details

Single-Server Scheme

The server has a database of size n ($T[0], \ldots, T[n-1]$), which is divided into $\sqrt{n}$ chunks (rows) each of size $\sqrt{n}$. For convenience, we assume $\sqrt{n}$ is an even integer. The client wants to query the x-th element in the database. We define the chunk (row) index of x as $\mathsf{chunk}(x)$.

1. Offline Preprocessing

(1) **Key Generation:** The client samples a PRF key $K_{\text{hint}} \in \{0,1\}^\lambda$ for generating primary and backup hints. To streamline our notation, we define a function instance by fixing the key:

$$\mathsf{PRF}_{K_{\text{hint}}}(\cdot) = \mathsf{PRF}(K_{\text{hint}}, \cdot)$$

For the remainder of this section, we will use the shorthands $\mathsf{PRF}(x)$ to denote $\mathsf{PRF}_{K_{\text{hint}}}(x)$.

(2) **Hint Initialization:** The client initializes the hints as follows:

- **1.** Initialize primary hint IDs as $\{0, 1, \ldots, M_1 - 1\}$, and the backup hint IDs as $\{M_1, M_1 + 1, \ldots, M_1 + M_2 - 1\}$.
- **2.** For $j = 0, 1, ..., M_1+M_2-1$, compute $V_j = [v_{j,0}, v_{j,1}, \ldots, v_{j,\sqrt{n}-1}]$ where $v_{j,k} = \mathsf{PRF}(\text{"select"} \parallel j \parallel k)$, then find and store the median $\hat{v}_j$ of V_j as the cutoff for chunk selection. If $j < M_1$, set the extra index e_j to a random index from a random chunk not in $\{k | v_{j,k} < \hat{v}_j\}$. Specifically, we can randomly select an additional chunk $c_j = \mathsf{PRF}(\text{"select-chunk"} \parallel j \parallel \text{nonce})$ until $v_{j,c_j} < \hat{v}_j$ no longer holds (expected number of iterations is 2), then compute $e_j = c_j\sqrt{n} + \mathsf{PRF}(\text{"offset"} \parallel$

$j \| c_j$) as the extra index. If $j \geq M_1$, set the extra index e_j to $\perp$.

- **3.** Initialize primary hint parities as $\{p_i = 0\}_{i\in[M_1]}$. For the backup table, initialize parities as $\{p'_j = 0\}_{j\in\{M_1,M_1+1,\ldots,M_1+M_2-1\}}$.

(3) **Streaming Download:** The client downloads the encrypted database in a streaming manner. When processing the k-th chunk with elements $T[k\sqrt{n}], \cdots, T[(k+1)\sqrt{n}-1]$:

- **Update primary hints:** For $j \in [M_1]$, compute $v_{j,k} = \mathsf{PRF}(\text{"select"} \| j \| k)$. If $v_{j,k} < \hat{v}_j$, then compute $p_j \leftarrow p_j \oplus (T[r_{j,k} + k\sqrt{n}])$, where $r_{j,k} = \mathsf{PRF}(\text{"offset"} \| j \| k)$. If $\lfloor e_j/\sqrt{n} \rfloor = k$, then $p_j \leftarrow p_j \oplus (T[e_j])$.
- **Update backup hints:** For $j \in \{M_1, M_1+1, \ldots, M_1+M_2-1\}$, compute $v_{j,k} = \mathsf{PRF}(\text{"select"} \| j \| k)$. If $v_{j,k} < \hat{v}_j$, then compute $p_j \leftarrow p_j \oplus (T[r_{j,k} + k\sqrt{n}])$, otherwise let $p'_j \leftarrow p'_j \oplus (T[r_{j,k}+k\sqrt{n}])$, where $r_{j,k} = \mathsf{PRF}(\text{"offset"} \| j \| k)$.
- **Clean up:** Delete processed data from local storage.

(4) **Hint Storage:** The client stores primary hints $H := \{(j, \hat{v}_j, e_j, p_j)\}_{j\in[M_1]}$ and backup hints $H' := \{(j, \hat{v}_j, p_j, p'_j)\}_{j\in\{M_1,M_1+1,\ldots,M_1+M_2-1\}}$. For each hint, the client additionally stores a bit $b_j (j \in [M_1 + M_2])$ to indicate whether it is necessary to redefine '<' as '>' for this hint. These bits are initialized to 0.

2. Online Process: Query $x \in \{0, 1, \ldots, n-1\}$

(1) **Set Finding:** Let x be the queried index and $c_x = \mathsf{chunk}(x)$. The client finds a primary hint j such that either of the following holds: $x = e_j$, or both following conditions hold: $v_{j,c_x} < \hat{v}_j$ and $r_{j,c_x} = x \bmod \sqrt{n}$, where $v_{j,c_x} = \mathsf{PRF}(\text{"select"} \| j \| c_x)$ and $r_{j,c_x} = \mathsf{PRF}(\text{"offset"} \| j \| c_x)$.

(2) **Query Transmission:** If step (2-1) succeeded, the client initializes two sets $S = \emptyset$ and $S' = \emptyset$. For $k = 0, 1, \ldots, \sqrt{n}-1$, if $v_{j,k} < \hat{v}_j$, then $S = S \cup \{r_{j,k} + k\sqrt{n}\}$; if $\mathsf{chunk}(e_j) = k$, then $S = S \cup \{e_j\}$; if neither holds, then $S' = S' \cup \{\mathsf{PRF}(\text{"dummy-offset"} \| j \| k) + k\sqrt{n}\}$.The resulting set S is precisely the index set corresponding to hint j, while S' is a dummy set constructed by selecting one random index from each chunk not covered by S. Then the client lets $S = S \setminus \{x\}$ and $S' = S' \cup \{\mathsf{PRF}(\text{"dummy-offset"} \| j \| c_x) + c_x\sqrt{n}\}$. The client sends (S, S') or (S', S) to the server with probability $1/2$ (we denote the query content received by the server as (S_0, S_1)). If step (2-1) failed, the client partitions the $\sqrt{n}$ chunks into two equal-sized subsets. For each subset, it constructs set S_i ($i \in 0, 1$) by selecting one uniform random index per chunk and sends (S_0, S_1) to the server.

(3) **Server Computation:** Upon receiving the query (S_0, S_1), for $i = 0, 1$, the server computes $q_i = \sum_{k\in S_i} T[k]$.

(4) **Response Processing:** Upon receiving q_0 and q_1 (the client can recognize which one corresponds to S; for simplicity, we assume that it is q_0), the client outputs $T[x] = q_0 \oplus p_j$ if step (2-1) succeeded.

(5) **Refresh:** Let J be the ID of the next unused pair of backup hints, the client computes v_{J,c_x}. If $v_{J,c_x} > \hat{v}_J$ then the client sets $p_j \leftarrow p'_j$ and set b_j to 1 to redefine "<" to be "greater than" for this hint. If step (2-1) succeeded, the client replaces hint j with backup hint $(J, \hat{v}_J, x, p_j \oplus T[x])$.

B Deferred Proofs

B.1 Proof of lemma 3.2

Lemma B.1 (Gaussian Posterior(restated)). *Let $Q > 0$ be an integer and $\sigma_1, \sigma_2 > 0$ be reals. For $\gamma_0, \cdots, \gamma_Q - 1 \in R^K$, where $\gamma_i = (\gamma_{i,0}, \cdots, \gamma_{i,K-1})$, we denote the corresponding negacyclic matrix of $\gamma_{i,j}$ as $\mathbf{M}(\gamma_{i,j})$, and we let $\mathbf{\Gamma}_i = [\mathbf{M}(\gamma_{i,0})|\mathbf{M}(\gamma_{i,1})|\cdots|\mathbf{M}(\gamma_{i,K-1})]$. We let $\Sigma_0 = (\frac{1}{\sigma_1^2}\mathbf{I}_{NK} + \frac{1}{\sigma_2^2}\sum_{i=0}^{Q-1}\mathbf{\Gamma}_i^\top\mathbf{\Gamma}_i)^{-1}$. Then, the following two distributions over R^{K+Q} are statistically identical.*

$$\left\{(\mathbf{r}, z_0, \ldots, z_{Q-1}) \;\middle|\; \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{NK},\sigma_1},\ y_i \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_2},\ z_i = \gamma_i^\top\mathbf{r} + y_i\right\}$$

$$\left\{(\hat{\mathbf{r}}, z_0, \ldots, z_{Q-1}) \;\middle|\; \begin{array}{l}\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{NK},\sigma_1},\ y_i \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_2},\ z_i = \gamma_i^\top\mathbf{r} + y_i, \\ \mathbf{c} = \frac{1}{\sigma_2^2}\Sigma_0 \cdot \sum_{i=0}^{Q-1}\mathbf{\Gamma}_i^\top\mathbf{z}_i,\ \hat{\mathbf{r}} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c},\sqrt{\Sigma_0}}\end{array}\right\}$$

Proof. We treat the secret $\mathbf{r}$ as a vector in $\mathbb{Z}^{NK}$ and the hints z_i as vectors in $\mathbb{Z}^N$. Therefore, we can rewrite $z_i = \gamma_i^\top\mathbf{r} + y_i$ as $\mathbf{z}_i = \mathbf{\Gamma}_i \cdot \mathbf{r} + \mathbf{y}_i$, where the variables now denote their corresponding integer vector representations.

Let $\mathbf{z} = (\mathbf{z}_0, \ldots, \mathbf{z}_{Q-1})$ be the collection of hints. We analyze the joint probability mass function of the secret $\mathbf{r}$ and the hints $\mathbf{z}$.

The probability that the honest execution outputs $(\mathbf{r}, \mathbf{z})$, denoted as $\mathbf{v} \in \mathbb{Z}^{NK}$ and $\mathbf{w} = (\mathbf{w}_0, \ldots, \mathbf{w}_{Q-1}) \in (\mathbb{Z}^N)^Q$, is given by:

$$\begin{aligned}\Pr[\mathbf{r} = \mathbf{v}, \mathbf{z} = \mathbf{w}] &= \Pr[\mathbf{r} = \mathbf{v}] \cdot \Pr[\mathbf{z} = \mathbf{w} \mid \mathbf{r} = \mathbf{v}] \\ &= \mathcal{D}_{\mathbb{Z}^{NK},\sigma_1}(\mathbf{v}) \cdot \prod_{i=0}^{Q-1}\mathcal{D}_{\mathbb{Z}^N,\sigma_2}(\mathbf{w}_i - \mathbf{\Gamma}_i\mathbf{v}) \\ &\propto \exp\left(-\pi\left(\frac{\mathbf{v}^\top\mathbf{v}}{\sigma_1^2} + \sum_{i=0}^{Q-1}\frac{(\mathbf{w}_i - \mathbf{\Gamma}_i\mathbf{v})^\top(\mathbf{w}_i - \mathbf{\Gamma}_i\mathbf{v})}{\sigma_2^2}\right)\right)\end{aligned}$$

By expanding the quadratic terms in the exponent and completing the square with respect to $\mathbf{v}$, we obtain:

$$\begin{aligned}&= \exp\left(-\pi\left(\mathbf{v}^\top\underbrace{\left(\frac{1}{\sigma_1^2}\mathbf{I}_{NK} + \frac{1}{\sigma_2^2}\sum_{i=0}^{Q-1}\mathbf{\Gamma}_i^\top\mathbf{\Gamma}_i\right)}_{\Sigma_0^{-1}}\mathbf{v} - 2\mathbf{v}^\top\underbrace{\left(\frac{1}{\sigma_2^2}\sum_{i=0}^{Q-1}\mathbf{\Gamma}_i^\top\mathbf{w}_i\right)}_{\Sigma_0^{-1}\mathbf{c}} + C(\mathbf{w})\right)\right) \\ &= \exp\left(-\pi((\mathbf{v}-\mathbf{c})^\top\Sigma_0^{-1}(\mathbf{v}-\mathbf{c}) + C'(\mathbf{w}))\right)\end{aligned}$$

where $C(\mathbf{w})$ and $C'(\mathbf{w})$ are terms independent of $\mathbf{v}$, and $\mathbf{c} = \frac{1}{\sigma_2^2}\Sigma_0 \cdot \sum_{i=0}^{Q-1}\mathbf{\Gamma}_i^\top\mathbf{w}_i$. Thus, the joint distribution can be rewritten as:

$$\Pr[\mathbf{r} = \mathbf{v}, \mathbf{z} = \mathbf{w}] \propto \rho_{\sqrt{\Sigma_0}}(\mathbf{v} - \mathbf{c}) \cdot \text{marginal}(\mathbf{w}) \tag{4}$$

This factorization implies that the conditional distribution of the secret $\mathbf{r}$ given the hints $\mathbf{z}$ is a discrete Gaussian distribution centered at $\mathbf{c}$ with covariance Σ_0. Specifically:

$$\mathbf{r} \mid \mathbf{z} \sim \mathcal{D}_{\mathbb{Z}^{NK},\mathbf{c},\sqrt{\Sigma_0}} \tag{5}$$

This completes the characterization of the posterior distribution. □

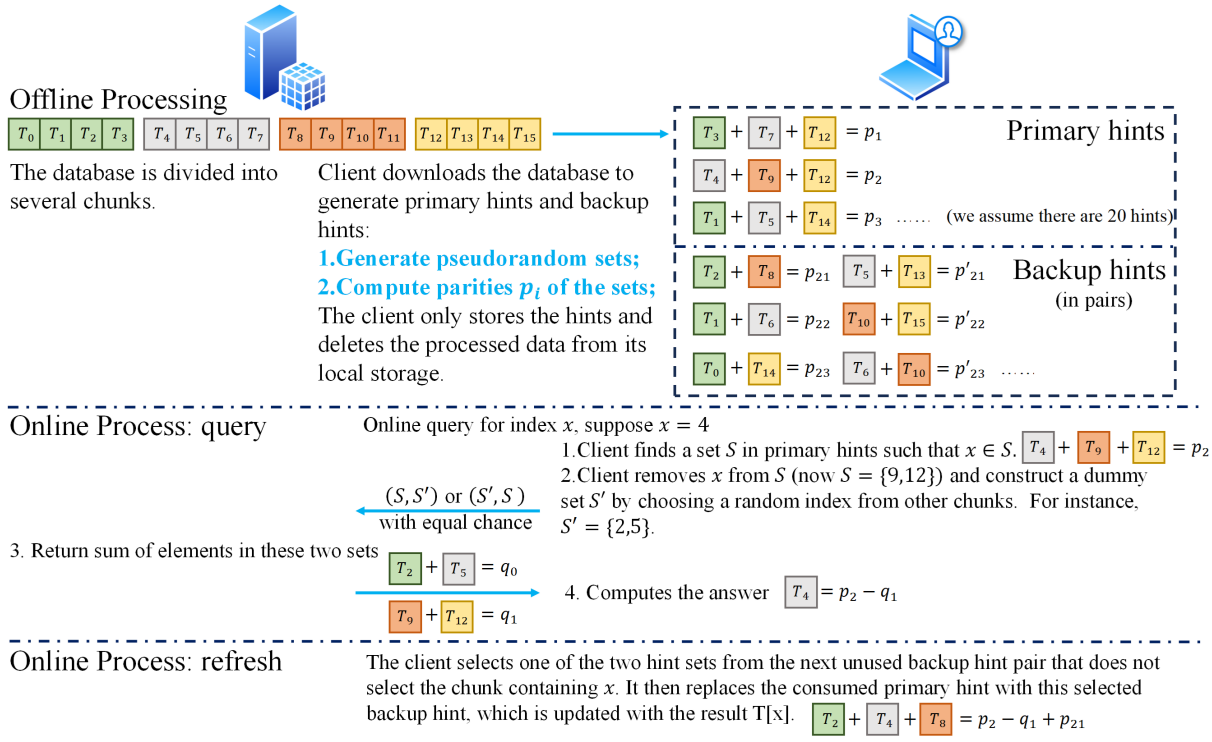

Figure 7: Overview of the RMS protocol workflow [35], illustrating the separation between the offline hint generation and the online query process using the dummy subset technique.

B.2 Proof of theorem 3.3

Theorem B.2 (Hardness of Adaptive Hint-MLWE(restated)). *Let d, m, Q, q be positive integers. Let $B > 0$ be a real number. Let $\chi = \mathcal{D}^{d+m}_{\mathbb{Z}^N,\sigma_1}$ and $\xi = \mathcal{D}^{Q}_{\mathbb{Z}^N,\sigma_2}$ be discrete Gaussian distributions for the secret and hint errors, respectively.*

Let $\sigma > 0$ be a real number satisfying the following condition:

$$\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_1^2} + \frac{B}{\sigma_2^2}\right). \tag{6}$$

If $\sigma \geq \sqrt{2}\cdot\eta_\varepsilon(\mathbb{Z}^{N(d+m)})$ for some negligible $\varepsilon \in (0, 1/2)$, then there exists an efficient reduction from $\mathrm{MLWE}_{R,d,m,q,\sigma}$ to $\text{A-HintMLWE}^{Q,\xi,B}_{R,d,m,q,\chi}$ that reduces the advantage by at most 2ε.

Specifically, for any adversary $\mathcal{A}$ that solves the Adaptive Hint-MLWE problem with a set of queries (or challenge vectors) γ satisfying $\sum_{i=0}^{Q-1}\sum_{j=0}^{d+m-1}\|\gamma_{i,j}\|_1^2 \leq B$, there exists a simulator that solves the standard MLWE problem.

Proof. Let $(\mathbf{A}, \mathbf{b}) \in \mathcal{R}_q^{m\times d}\times\mathcal{R}_q^m$ be an instance of the $\mathrm{MLWE}_{R,d,m,q,\sigma}$ problem. We construct a reduction $\mathcal{B}$ that simulates the A-HintMLWE game for an adversary $\mathcal{A}$.

1. Setup and Query: $\mathcal{B}$ receives the MLWE instance $(\mathbf{A}, \mathbf{b})$ and initiates the adversary $\mathcal{A}$. $\mathcal{A}$ first outputs a sequence of challenge vectors $\gamma = (\gamma_0, \ldots, \gamma_{Q-1})$ satisfying $\sum_{i,j}\|\gamma_{i,j}\|_1^2 \leq B$.

Upon receiving γ, $\mathcal{B}$ internally samples a dummy secret $\mathbf{r}_{sim} \leftarrow \mathcal{D}_{\mathbb{Z}^{N(d+m)},\sigma_1}$ and dummy errors $y_{i,sim} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_2}$ to compute the simulated hints $z_i = \gamma_i^\top \mathbf{r}_{sim} + y_{i,sim}$ for $0 \leq i < Q$. Based on the received γ and the generated z_i, $\mathcal{B}$ defines the posterior covariance matrix $\Sigma_0^{-1} = \sigma_1^{-2}\mathbf{I} + \sigma_2^{-2}\sum\boldsymbol{\Gamma}_i^\top\boldsymbol{\Gamma}_i$ and the mean vector $\mathbf{c} = \Sigma_0(\sigma_2^{-2}\sum\boldsymbol{\Gamma}_i^\top\mathbf{z}_i)$ according to Lemma 3.2. At this stage, $\mathcal{B}$ holds $(z_0, \ldots, z_{Q-1})$ temporarily without sending them to $\mathcal{A}$.

2. Positive Definiteness and Norm Estimation: To ensure the validity of the perturbation that will be used later, we analyze the spectral norm of $\sum_{i=0}^{Q-1}\boldsymbol{\Gamma}_i^\top\boldsymbol{\Gamma}_i$. For $\boldsymbol{\Gamma}_i = [\mathbf{M}(\gamma_{i,0}) \mid \cdots \mid \mathbf{M}(\gamma_{i,d+m-1})]$, we utilize the property that the non-zero eigenvalues of $\boldsymbol{\Gamma}_i^\top\boldsymbol{\Gamma}_i$ and $\boldsymbol{\Gamma}_i\boldsymbol{\Gamma}_i^\top = \sum_j \mathbf{M}(\gamma_{i,j})\mathbf{M}(\gamma_{i,j})^\top$ are identical. By the triangle inequality and $\|\mathbf{M}(a)\|_2 \leq \|a\|_1$:

$$\|\sum_{i=0}^{Q-1}\boldsymbol{\Gamma}_i^\top\boldsymbol{\Gamma}_i\|_2 \leq \sum_{i=0}^{Q-1}\|\boldsymbol{\Gamma}_i\boldsymbol{\Gamma}_i^\top\|_2 \leq \sum_{i=0}^{Q-1}\sum_{j=0}^{d+m-1}\|\mathbf{M}(\gamma_{i,j})\|_2^2 \leq \sum_{i,j}\|\gamma_{i,j}\|_1^2 \leq B. \tag{7}$$

From the theorem condition $\sigma^{-2} = 2(\sigma_1^{-2} + B\sigma_2^{-2})$, it follows that:

$$\|\Sigma_0^{-1}\|_2 \leq \sigma_1^{-2} + \sigma_2^{-2}\|\sum_{i=0}^{Q-1}\boldsymbol{\Gamma}_i^\top\boldsymbol{\Gamma}_i\|_2 \leq \sigma_1^{-2} + B\sigma_2^{-2} = \frac{1}{2\sigma^2} < \frac{1}{\sigma^2}. \tag{8}$$

This implies $\Sigma_0 \succ 2\sigma^2\mathbf{I} \succ \sigma^2\mathbf{I}$, thus $\Sigma_{pert} = \Sigma_0 - \sigma^2\mathbf{I}$ is strictly positive definite.

3. Distribution Transformation: $\mathcal{B}$ samples $\mathbf{t} \leftarrow \mathcal{D}_{\mathbb{Z}^{N(d+m)},\mathbf{c},\sqrt{\Sigma_{pert}}}$. Then, $\mathcal{B}$ constructs the A-HintMLWE target by shifting the MLWE challenge vector $\mathbf{b}$ directly:

$$\mathbf{target} = \mathbf{b} + [\mathbf{I}_m|\mathbf{A}]\mathbf{t}$$

Finally, $\mathcal{B}$ reveals the tuple $(\mathbf{A}, \mathbf{target}, z_0, \ldots, z_{Q-1})$ to the adversary $\mathcal{A}$. $\mathcal{B}$ outputs whatever bit $\mathcal{A}$ outputs.

To justify the correctness of this reduction, we analyze the case where the underlying MLWE instance is Real (i.e., $\mathbf{b} = [\mathbf{I}_m|\mathbf{A}]\mathbf{r}'$ for $\mathbf{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^{N(d+m)},\sigma}$). In this case, the constructed target can be expressed as:

$$\mathbf{target} = [\mathbf{I}_m|\mathbf{A}]\mathbf{r}' + [\mathbf{I}_m|\mathbf{A}]\mathbf{t} = [\mathbf{I}_m|\mathbf{A}](\mathbf{r}' + \mathbf{t})$$

Let $\mathbf{r}_{out} = \mathbf{r}' + \mathbf{t}$ be the effective secret. To utilize the Convolution Lemma 2.2 to prove $\mathbf{r}_{out} \approx \mathcal{D}_{\mathbf{c},\sqrt{\Sigma_0}}$, we define Σ_3 such that $\Sigma_3^{-1} = (\sigma^2\mathbf{I})^{-1} + \Sigma_{pert}^{-1}$. Since $\Sigma_{pert} \succ \sigma^2\mathbf{I}$, we have:

$$\begin{aligned} \|\Sigma_3^{-1}\|_2 &= \|\sigma^{-2}\mathbf{I} + (\Sigma_0 - \sigma^2\mathbf{I})^{-1}\|_2 \le \sigma^{-2} + (\sigma_{min}(\Sigma_0) - \sigma^2)^{-1} \\ &\le \sigma^{-2} + \sigma^{-2} = 2\sigma^{-2}. \end{aligned} \tag{9}$$

Since $\sigma_{min}(\Sigma_3) = \|\Sigma_3^{-1}\|_2^{-1} \ge \sigma^2/2$, the condition $\sigma/\sqrt{2} \ge \eta_\varepsilon$ (or $\sigma \ge \sqrt{2}\eta_\varepsilon$) ensures that $\mathbf{r}_{out}$ is within statistical distance 2ε of $\mathcal{D}_{\mathbf{c},\sqrt{\Sigma_0}}$.

Therefore, the distribution of the tuple

$$(\mathbf{A}, \mathbf{b} + [\mathbf{I}_m|\mathbf{A}]\mathbf{t}, \gamma_0, \ldots, \gamma_{Q-1}, z_0, \ldots, z_{Q-1}) \tag{10}$$

is within statistical distance 2ε of the distribution

$$(\mathbf{A}, [\mathbf{I}_m|\mathbf{A}]\hat{\mathbf{r}}, \gamma_0, \ldots, \gamma_{Q-1}, z_0, \ldots, z_{Q-1}) \tag{11}$$

where $\hat{\mathbf{r}}$ is sampled from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{N(d+m)},\mathbf{c},\sqrt{\Sigma_0}}$.

4. Statistical Indistinguishability: As the last step, we apply the statistical identity established in Lemma 3.2 to the joint distribution of $(\hat{\mathbf{r}}, z_0, \ldots, z_{Q-1})$. Lemma 3.2 states that the distribution of $(\hat{\mathbf{r}}, z_0, \ldots, z_{Q-1})$ is identical to that of $(\mathbf{r}, z_0, \ldots, z_{Q-1})$ where $\mathbf{r} \leftarrow \chi$, $y_i \leftarrow \xi$, and $z_i = \gamma_i^\top \mathbf{r} + y_i$. We now analyze the two cases of the MLWE instance:

- **Case** $b = 0$ **(Real):** The input is $\mathbf{b} = [\mathbf{I}_m|\mathbf{A}]\mathbf{r}'$ for $\mathbf{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^{N(d+m)},\sigma}$. From the observation above, the simulated output $(\mathbf{A}, \mathbf{b} + [\mathbf{I}_m|\mathbf{A}]\mathbf{t}, \gamma_0, \ldots, \gamma_{Q-1}, z_0, \ldots, z_{Q-1})$ is within statistical distance 2ε of
$$(\mathbf{A}, [\mathbf{I}_m|\mathbf{A}]\mathbf{r}, \gamma_0, \ldots, \gamma_{Q-1}, z_0, \ldots, z_{Q-1}), \tag{12}$$
which exactly follows the Real distribution of samples from $\text{A-HintMLWE}^{Q,\xi,B}_{R,d,m,q,\chi}$.
- **Case** $b = 1$ **(Random):** The input is $\mathbf{b} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$. Then the output target $\mathbf{b} + [\mathbf{I}_m|\mathbf{A}]\mathbf{t}$ follows the same uniform distribution over $\mathcal{R}_q^m$, which is independent of the challenges γ_i and hints z_i. Thus, the simulated tuple follows the Random distribution of A-HintMLWE:
$$(\mathbf{A}, \mathbf{u}, \gamma_0, \ldots, \gamma_{Q-1}, z_0, \ldots, z_{Q-1}) \quad \text{where } \mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m). \tag{13}$$

Therefore, the reduction is correct and reduces the advantage by at most 2ε. □

B.3 Proof of Theorem B.3

Theorem B.3 (Server Security(restated)). *Under the same assumptions as in Theorem 5.2, there exists a probabilistic polynomial-time (PPT) simulator S_1 in the $\mathcal{F}_{OT}$-hybrid model such that*

$$S_1(1^\lambda, 1^\kappa, \mathbf{x}, \{f(x_t)\}_{t\in\{1,2,\ldots,Q\}}) \overset{c}{\equiv} \mathit{VIEW}_1^\Pi(1^\lambda, 1^\kappa, \mathbf{x}, f)$$

Specifically, for every PPT adversary $\mathcal{A}$ that distinguishes the simulated view from the real view with advantage $Adv^{\mathcal{A}}_{\Pi,S_1}(\lambda)$, there exists a PPT adversary $\mathcal{B}$ against the Adaptive Hint-MLWE problem such that:

$$Adv^{\mathcal{A}}_{\Pi,S_1}(\lambda) \le Adv^{\mathcal{B}}_{\text{A-HintMLWE}}(\lambda) + negl(\lambda)$$

Proof. The view of the client in an execution of Π is defined as:

$$\text{VIEW}_1^\Pi(1^\lambda, 1^\kappa, \mathbf{x}, f) = (\mathbf{x}, \{f(x_t)\}_{t=1}^Q, pk, rt, \{\tilde{\mathbf{T}}[k]\}_{k=0}^{n/N-1}, \{\overline{\beta'_t}\}_{t=1}^Q)$$

where $\text{pk} = (b_p, a_p)$ is the server's public key, rt is the client's internal random tape (determining masks r_t, random sets S_t, and blinding noise $u_{q_t}, e_{q_t}, e'_{q_t}$). $\overline{\beta'_t}$ is the message received from the $\mathcal{F}_{\text{OT}}$ functionality in the t-th query.

We prove this claim by constructing a simulator S_1 and use a hybrid argument to reduce the distinguishability of the simulation to the hardness of the Adaptive Hint-MLWE problem.

We define the simulator S_1 as follows:

Simulator $S_1(1^\lambda, \mathbf{x}, \{y_t\}_{t=1}^Q)$**:**

(1) **Setup Simulation:** S_1 generates the client's random tape rt uniformly at random. This determines the selected hint sets S_t (we note that the distribution of S_t depends on the choice of x_t), masks r_t, and blinding noises $u_{q_t}, e_{q_t}, e'_{q_t}$ for all $t \in \{1, \ldots, Q\}$. Instead of performing BFV Setup and Encryption, S_1 samples the public key and the encrypted database from the uniform distribution:

$$pk^* \leftarrow \mathcal{U}(\mathcal{R}_q \times \mathcal{R}_q), \quad \tilde{\mathbf{T}}[k]^* \leftarrow \mathcal{U}(\mathcal{R}_q \times \mathcal{R}_q) \quad \forall k \in \{0, \ldots, n/N-1\}$$

(2) **Internal Secret Sampling:** To simulate the query responses consistently, S_1 samples a dummy secret vector $\mathbf{r}^*$ and dummy query noises $\{e^*_{ans_t}\}_{t=1}^Q$ from the distributions specified in the protocol (where $\mathbf{r}^*$ includes terms corresponding to a secret key, public key error, and encryption errors):

$$\mathbf{r}^* \leftarrow \chi^{2+n/N}, \quad e^*_{ans_t} \leftarrow \chi_{ans}$$

(3) **Query Simulation:** For each $t \in \{1, \ldots, Q\}$, S_1 constructs the simulated OT response $\overline{\beta'_{t,sim}}$. First, S_1 computes the challenge vector $\gamma_t \in \mathcal{R}_q^{2+n/N}$ based on S_t and the blinding terms derived from rt. The components of γ_t are:

$$\gamma_t = \left(e_{q_t},\ u_{q_t},\ \sum_{i\in S_t\cap B_0} X^{-col(i)},\ \ldots,\ \sum_{i\in S_t\cap B_{n/N-1}} X^{-col(i)} \right)^\top$$

where B_k denotes the set of indices in the k-th database row, i.e., $B_k = \{kN, kN+1, \cdots, (k+1)N-1\}$, and $col(i)$ denotes the index of the column containing i, i.e., $col(i) = i \bmod N$. Note that the hint set S_t contains at most 1 index in each database row, therefore $\sum_{i\in S_t\cap B_k} X^{-col(i)}$ is either 0 or a monomial.

Then, S_1 computes the simulated response using the known output $y_t = f(x_t)$:

$$\overline{\beta'_{t,sim}} = \left\lfloor \frac{q}{p} \right\rfloor \cdot y_t + r_t[0] + e'_{q_t}[0] + [\langle \gamma_t, \mathbf{r}^* \rangle]_0 + e^*_{ans_t}$$

where $[\cdot]_0$ denotes the constant coefficient of the polynomial.

(4) **Output View:** S_1 outputs the view:

$$\text{VIEW}_{S_1} = (\mathbf{x}, \{y_t\}_{t=1}^Q, pk^*, rt, \{\tilde{\mathbf{T}}[k]^*\}_{k=0}^{n/N-1}, \{\overline{\beta'_{t,sim}}\}_{t=1}^Q)$$

Indistinguishability Analysis. We define a sequence of games to prove that the simulated view is computationally indistinguishable from the real view.

Game 0 (Real Protocol): This game corresponds to the real execution of the protocol in the $\mathcal{F}_{\text{OT}}$-hybrid model. The view is generated using the real protocol with a valid secret key sk and error vectors $\mathbf{e}_{enc}, e_p$, where $\mathbf{e}_{enc} = (e_{ct_0}, \cdots, e_{ct_{n/N-1}})$ consists of the noise polynomials corresponding to the ciphertexts $(\tilde{\mathbf{T}}[0], \cdots, \tilde{\mathbf{T}}[n/N - 1])$. The secret vector is $\mathbf{r} = (sk, e_p, \mathbf{e}_{enc})$. The public key and ciphertexts are structurally $\mathbf{target} = [\mathbf{I}|\mathbf{A}]\mathbf{r}$. The response is computed as:

$$\overline{\beta'_t} = \left\lfloor \frac{q}{p} \right\rfloor f(x_t) + r_t[0] + e'_{q_t}[0] + [\langle \gamma_t, \mathbf{r} \rangle]_0 + e_{ans_t}$$

Game 1 (Random Public Key and Database): In this game, we modify the Setup phase. The public key pk and the database ciphertexts $\{\tilde{\mathbf{T}}[k]\}$ are replaced by uniform random polynomials drawn from $\mathcal{U}(\mathcal{R}_q^2)$. However, the OT responses $\{\overline{\beta'_t}\}$ are still generated using the *original* secret $\mathbf{r}$ (which is now decoupled from the public parameters) and the real query noise e_{ans_t}. Specifically, the challenger samples $\mathbf{r} \leftarrow \chi^{2+n/N}$ internally and computes $\overline{\beta'_t}$ exactly as in Game 0, despite pk and $\tilde{\mathbf{T}}[k]$ being random.

Reduction to Adaptive Hint-MLWE. We claim that Game 0 is indistinguishable from Game 1 under the Adaptive Hint-MLWE assumption. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\text{VIEW}_{\text{Game 0}}$ from $\text{VIEW}_{\text{Game 1}}$ with non-negligible advantage. We construct an adversary $\mathcal{B}$ against the A-HintMLWE problem. Here, we set the A-HintMLWE error distribution ξ to be identical to the server's perturbation noise distribution χ_{ans}.

1. **Pre-computation and Challenge Vector Submission:** Since the protocol operates in the semi-honest model, the client $\mathcal{A}$ strictly follows the protocol. $\mathcal{B}$ first samples a random tape rt for $\mathcal{A}$. Utilizing rt and the client's input queries $\mathbf{x}$, $\mathcal{B}$ locally emulates the client's deterministic query generation process. This allows $\mathcal{B}$ to pre-calculate all corresponding challenge vectors $\gamma_1, \ldots, \gamma_Q$. $\mathcal{B}$ then submits these challenge vectors to the A-HintMLWE challenger.

2. **Setup:** The A-HintMLWE challenger returns the public matrix $\mathbf{A}_{\text{hint}} \in \mathcal{R}_q^{m \times d}$, the target vector $\mathbf{target}_{\text{hint}} \in \mathcal{R}_q^m$, and the sequence of hints $z_1, \ldots, z_Q$, where $m = 1 + n/N$. Note that the hint $z_t = \langle \gamma_t, \mathbf{r} \rangle + y_t$, where $y_t \sim \chi_{ans}^N$ (so that $y_t[0]$ matches the distribution of e_{ans}). $\mathcal{B}$ constructs the simulated server view as follows:

- **Public Key Assignment**: $\mathcal{B}$ sets the public key $pk = (pk_b, pk_a)$ by assigning the first sample from the A-HintMLWE instance, such that $pk_a = \mathbf{A}_{\text{hint}}[0]$ and $pk_b = \mathbf{target}_{\text{hint}}[0]$.
- **Encrypted Database Construction**: $\mathcal{B}$ constructs the encrypted database $\tilde{\mathbf{T}} = \{\tilde{T}[k]\}_{k=0}^{n/N-1}$. For each row k, the ciphertext $\tilde{T}[k] = (b_k, a_k)$ is formed using the $(k+1)$-th sample from the A-HintMLWE instance with a plaintext shift:

$$a_k = \mathbf{A}_{\text{hint}}[k+1], \quad b_k = \mathbf{target}_{\text{hint}}[k+1] + \lfloor q/p \rfloor \cdot \mathbf{T}[k] \pmod q$$

Then, $\mathcal{B}$ sends pk and the resulting encrypted database $\tilde{\mathbf{T}}$ to the adversary $\mathcal{A}$.

3. **Queries:** When $\mathcal{A}$ issues the t-th query, $\mathcal{B}$ leverages the random tape rt (used to determine $r_t[0]$) and the hint z_t obtained from the challenger to construct the OT response:

$$\overline{\beta'_t} = \left\lfloor \frac{q}{p} \right\rfloor f(x_t) + r_t[0] + e'_{q_t}[0] + [z_t]_0$$

Note that we only use the constant term of the hint. This is a valid reduction because providing the full polynomial z_t implies providing its constant term.

4. **Guess:** $\mathcal{B}$ outputs whatever bit $\mathcal{A}$ outputs.

If the A-HintMLWE environment is Real, $\mathcal{B}$ simulates Game 0 perfectly. If it is Random, $\mathcal{B}$ simulates Game 1 perfectly. Thus, $|\Pr[\mathcal{A}(\text{Game 0}) = 1] - \Pr[\mathcal{A}(\text{Game 1}) = 1]| \leq \text{Adv}^{\mathcal{B}}_{\text{A-HintMLWE}}(\lambda)$.

Game 1 ≡ Simulation: Observe that in Game 1, the public parameters $(pk, \tilde{\mathbf{T}}[k])$ are uniform random strings independent of $\mathbf{r}$. The responses $\overline{\beta'_t}$ are computed using a secret $\mathbf{r}$ and noise e_{ans_t} drawn from the exact same distributions as $\mathbf{r}^*$ and $e^*_{ans_t}$ in the Simulator S_1. Since $\mathbf{r}$ is not used anywhere else in Game 1 (it is statistically hidden from the public parameters), the joint distribution of $(\mathbf{r}, \{e_{ans_t}\})$ in Game 1 is identical to $(\mathbf{r}^*, \{e^*_{ans_t}\})$ in the Simulation. Therefore:

$$\{\text{VIEW}_{\text{Game 1}}\} \equiv \{\text{VIEW}_{S_1}\}$$

Combining the steps, we have:

$$\text{VIEW}_{\text{Real}} \equiv \text{VIEW}_{\text{Game 0}} \overset{c}{\approx} \text{VIEW}_{\text{Game 1}} \equiv \text{VIEW}_{S_1}$$

and

$$\text{Adv}^{\mathcal{A}}_{\Pi,S_1}(\lambda) \leq \text{Adv}^{\mathcal{B}}_{\text{A-HintMLWE}}(\lambda) + \text{negl}(\lambda)$$

which concludes the proof. □

B.4 Proof of Theorem 5.4

Theorem B.4 (Client Security(restated)). *Under the same assumptions as in Theorem 5.2, there exists a probabilistic polynomial-time (PPT) simulator S_2 in the $\mathcal{F}_{OT}$-hybrid model such that*

$$S_2(1^\lambda, 1^\kappa, f) \overset{c}{\equiv} \mathit{VIEW}_2^\Pi(1^\lambda, 1^\kappa, \mathbf{x}, f)$$

Specifically, for every PPT adversary $\mathcal{A}$ that distinguishes the simulated view from the real view with advantage $Adv^{\mathcal{A}}_{\Pi,S_2}$, the advantage is bounded by

$$Adv^{\mathcal{A}}_{\Pi,S_2} \leq Adv_{PRF}(\lambda) + Q \cdot Adv_{RLWE}(\lambda) + negl(\lambda)$$

where $Adv_{PRF}(\lambda)$ and $Adv_{RLWE}(\lambda)$ denote the maximum advantages of any PPT adversary against the PRF and the Decision-RLWE problem, respectively. Q is the total number of online queries, and is bounded by $poly(\lambda)$.

Proof. The view of the server in an execution of Π is defined as:

$$\text{VIEW}_2^\Pi(1^\lambda, \mathbf{x}, f) = (f, rt_2; (\text{view}_1, \ldots, \text{view}_Q))$$

where rt_2 is the server's random tape used to generate BFV secret key sk, public key (b_p, a_p) with noise e_p, ciphertexts $\tilde{\mathbf{T}}[0], \cdots, \tilde{\mathbf{T}}[n/N-1]$ and perturbation noise e_{ans_t}. $\text{view}_t = ((S_{t,0}, S_{t,1}), \tilde{\beta}''_t)$ denotes the client's t-th online query (2 query sets and a ciphertext for each query). Note that the server receives nothing from the $\mathcal{F}_{\text{OT}}$ functionality in each query.

We prove this claim by constructing a simulator S_2 and use a hybrid argument to prove the distinguishability.

We define the simulator S_2 as follows:

Simulator S_2: The simulator S_2, given the server's inputs $(1^\lambda, 1^\kappa, f, rt_2)$, generates a sequence of Q simulated online queries $(\text{view}_1^*, \ldots, \text{view}_Q^*)$ as follows:

(1) First, it uses the random tape rt_2 to generate BFV secret key sk, public key (b_p, a_p) with noise e_p, ciphertexts $\tilde{\mathbf{T}}[0], \cdots, \tilde{\mathbf{T}}[n/N-1]$ and perturbation noise e_{ans_t}, identical to the real protocol.
(2) For each $t \in \{1, \cdots, Q\}$:
   (a) It simulates the query sets received in the case of a query failure. It uniformly partitions the n/N database ($\mathbf{T}$) rows into two equal-sized subsets, $C_{t,0}$ and $C_{t,1}$. It then samples one random index from each row in $C_{t,0}$ to form $S_{t,0}$, and one from each row in $C_{t,1}$ to form $S_{t,1}$.
   (b) It samples a uniformly random 'ciphertext' $\tilde{\beta}_t''^* = (b_{q_t}^*, a_{q_t}^*) \leftarrow \mathcal{U}(\mathcal{R}_q^2)$.
   (c) It records $((S_{t,0}, S_{t,1}), \tilde{\beta}_t''^*)$ as the simulated t-th online query.

The simulator outputs the full view $(f, rt_2; (\text{view}_1^*, \ldots, \text{view}_Q^*))$.

Indistinguishability Analysis. We define a sequence of games to prove that the simulated view is computationally indistinguishable from the real view.

Game 0 (Real Protocol): This is the real execution of our PFE protocol for a sequence of Q client queries $\mathbf{x} = (x_1, \ldots, x_Q)$. The query sets and ciphertexts for each query are generated according to the protocol specification. We note that the client uses pseudo-random functions to generate the hint sets and the query sets.

Game 1 (Replacing PRFs with truly random functions): This experiment is identical to Game 0, except that the pseudorandom function PRF used for hint and query set generation is replaced with a truly random function.

Indistinguishability of Game 0 and Game 1: The distributions of the server's view in Game 0 and Game 1 are computationally indistinguishable. A standard argument shows that any adversary $\mathcal{A}$ distinguishing between these two experiments can be used to construct a distinguisher for the PRF. The advantage of $\mathcal{A}$ is bounded by $\text{Adv}_{\text{PRF}}(\lambda)$.

Game 2 (Replacing the queried ciphertexts with random ciphertexts):

This experiment is identical to Game 1, with one modification. We recall that in Game 1 (and in the Real Protocol), for the t-th query ($t \in \{1, \cdots, Q\}$), the client first randomly chooses a hint set S_t (with hint ID j_t) in H satisfying $x_t \in S_t$, whose corresponding encrypted parity is $\tilde{p}_{j_t}$. To analyze the security, we explicitly expand $\tilde{p}_{j_t}$ into its components (b_{S_t}, a_{S_t}). Then the client performs the ciphertext masking operation: it first generates a fresh ciphertext of 0, denoted as $(b_p u_{q_t} + e'_{q_t}, a_p u_{q_t} + e_{q_t})$, where (b_p, a_p) is the public key and $u_{q_t}, e_{q_t}, e'_{q_t} \leftarrow \chi_{err}$. Then, the client computes the masked ciphertext as follows:

$$\tilde{\beta}_t'' := (b_{q_t}, a_{q_t}) = (b_{S_t} + b_p u_{q_t} + e'_{q_t} + r_t, a_{S_t} + a_p u_{q_t} + e_{q_t})$$

where $r_t \leftarrow \mathcal{U}(\mathcal{R}_q)$ is a masking polynomial for the ciphertext.

In Game 2, on the other hand, for the t-th query, the client now samples a uniformly random 'ciphertext' $\tilde{\beta}_t''^* := (b_{q_t}^*, a_{q_t}^*) \leftarrow \mathcal{U}(\mathcal{R}_q^2)$.

Indistinguishability of Game 1 and Game 2:

We will show that

$$(f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q,t}, a_{q,t})\}_{t=1}^Q) \overset{c}{\approx} (f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q,t}^*, a_{q,t}^*)\}_{t=1}^Q)$$

The only difference lies in the query ciphertexts. We argue that the distribution of $\{\tilde{\beta}_t'' = (b_{q_t}, a_{q_t})\}_{t=1}^Q$ is computationally indistinguishable from a uniform distribution $\mathcal{U}(\mathcal{R}_q^{2Q})$, even conditioned on f, rt_2 and the query sets $\{(S_{t,0}, S_{t,1})\}_{t=1}^Q$.

First, for the b-component of the ciphertexts, since each r_t is sampled independently and uniformly at random from $\mathcal{R}_q$ and is independent of all other terms, the summation $\{b_{q_t}\}_{t=1}^Q = \{b_{S_t} + b_p u_{q_t} + e'_{q_t} + r_t\}_{t=1}^Q$ results in a perfectly uniform distribution over $\mathcal{R}_q^Q$ by the property of the one-time pad. Therefore,

$$(f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}, a_{q_t})\}_{t=1}^Q) \equiv (f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}^*, a_{q_t})\}_{t=1}^Q)$$

Second, for the a-component, we have $a_{q_t} = a_{S_t} + a_p u_{q_t} + e_{q_t}$. Let $\alpha_t = a_p u_{q_t} + e_{q_t}$ be the term contributing to the ciphertext masking. We first need to prove that $\{\alpha_t\}_{t=1}^Q$ is computationally indistinguishable from $\{u'_t\}_{t=1}^Q$ ($u'_t \leftarrow \mathcal{R}_q$) conditioned on any choice of f,rt,$\{(S_{t,0}, S_{t,1})\}_{t=1}^Q$ and $\{(b_{q_t}^*)\}_{t=1}^Q$. Note that u_{q_t} and e_{q_t} are freshly sampled (independent of f,rt,$\{(S_{t,0}, S_{t,1})\}_{t=1}^Q$ and $\{(b_{q_t}^*)\}_{t=1}^Q$), we have that given a_p,$\{\alpha_t\}_{t=1}^Q$ is independent of all other terms in the view. Therefore, it suffices to show that given a_p, $\{\alpha_t\}_{t=1}^Q$ and $\{u'_t\}_{t=1}^Q$ are computationally indistinguishable.

We prove this claim from a standard hybrid argument. Let H_0 be the distribution where all Q samples are uniform, and H_Q be the distribution where all Q samples are RLWE samples. We define a sequence of hybrid distributions H_k where the first k samples are RLWE samples and the remaining $Q-k$ are uniform. Since u_{q_t}, e_{q_t} are fresh for each sample, and that $a_p \leftarrow \mathcal{U}(\mathcal{R}_q)$, distinguishing H_k from H_{k+1} reduces strictly to the problem of distinguishing a single RLWE sample $(a_p, a_p u_{q_{k+1}} + e_{q_{k+1}})$ from uniform (a_p, u'_{k+1}). Under the RLWE assumption, any PPT adversary has a negligible advantage in distinguishing one sample. Therefore, by the triangle inequality, the advantage in distinguishing H_0 from H_Q is bounded by $Q \cdot \text{Adv}_{\text{RLWE}}(\lambda)$, which remains negligible.

Therefore,

$$(f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}^*, \alpha_t)\}_{t=1}^Q) \overset{c}{\approx} (f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}^*, u'_t)\}_{t=1}^Q)$$

for $u'_t \leftarrow \mathcal{U}(\mathcal{R}_q)$, which further implies

$$(f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}^*, a_{q_t})\}_{t=1}^Q) \overset{c}{\approx} (f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q_t}^*, u''_t)\}_{t=1}^Q)$$

where $u''_t = u'_t + a_{S_t}$ follows the same uniform distribution as $a_{q_t}^*$. (Note that u'_t is independent of a_{S_t})

Combining the discussions above, we have

$$(f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q,t}, a_{q,t})\}_{t=1}^Q) \overset{c}{\approx} (f, rt_2, \{(S_{t,0}, S_{t,1})\}_{t=1}^Q, \{(b_{q,t}^*, a_{q,t}^*)\}_{t=1}^Q)$$

where the advantage of $\mathcal{A}$ is bounded by $Q \cdot \text{Adv}_{\text{RLWE}}(\lambda)$.

Game 2≡ Simulation

The only difference lies in the process of generating query sets. We argue that the distribution of the sequence of query sets $\{(S_{t,0}, S_{t,1})\}_{t=1}^{Q}$ is statistically indistinguishable in both experiments given f and rt_2. The core of this argument relies on the privacy proof of the underlying hint mechanism, as detailed in the work of Ren et al.(RMS)[35].

Proof sketch:

- **Single Query Indistinguishability:** For any single query (e.g., the t-th query), the pair of sets $(S_{t,0}, S_{t,1})$ sent by the client in Game 2 is constructed from a "real" set and a "dummy" set. Because the hint selects a random half of the partitions and the dummy set is constructed from the other half, the union of the two sets covers all $\sqrt{n}$ partitions exactly once. Since the PRF is replaced by a truly random function (from Game 0) and the two sets are randomly permuted, the resulting pair is a uniformly random partition of the $\sqrt{n}$ chunks into two equal-sized halves. This is precisely the distribution generated by the simulator S_2 by definition. Therefore, for any single query, the views are statistically identical.
- **Extension to Multiple Queries:** To extend this argument to the full sequence of Q queries, we consider the hint replenishment process ('Refresh' step). As shown by Ren et al., the 'refresh' step is designed to ensure that the distribution of the hint sets in the primary hint H remains the same across queries. The consumed hint is replaced by an updated backup hint in a way that preserves the original statistical properties of the hint sets. Consequently, the argument of statistical indistinguishability for the query sets applies independently to each query $t \in \{1, \cdots, Q\}$.

Combining the steps, we have:

$$\mathrm{VIEW}_{\mathrm{Real}} \equiv \mathrm{VIEW}_{\mathrm{Game\ 0}} \overset{c}{\approx} \mathrm{VIEW}_{\mathrm{Game\ 1}} \overset{c}{\approx} \mathrm{VIEW}_{\mathrm{Game\ 2}} \equiv \mathrm{VIEW}_{S_2}$$

and

$$\mathrm{Adv}^{\mathcal{A}}_{\Pi,S_2} \leq \mathrm{Adv}_{\mathrm{PRF}}(\lambda) + Q \cdot \mathrm{Adv}_{\mathrm{RLWE}}(\lambda) + \mathrm{negl}(\lambda)$$

which concludes the proof.

C Open Science

In compliance with the ACM CCS Open Science policy, we provide a detailed description of the artifacts associated with this paper to ensure the transparency and reproducibility of our results.

C.1 Availability of Artifacts

We have made all artifacts necessary to evaluate the core contributions of this paper available for double-blind review. The artifacts include:

- **Source Code:** The complete C++ implementation of the PPFE framework, including the Oblivious Decryption mechanism and the Adaptive Hint-MLWE instantiation.
- **Baselines:** Implementations and configuration files for the comparative schemes (FABLE and SP-LUT) used in our evaluation.
- **Benchmarking Scripts:** Automated scripts to reproduce the latency, throughput, and communication cost experiments presented in Section 6.
- **Documentation:** A README file providing step-by-step instructions for building the project, installing dependencies, and running the experiments.

C.2 Access to Artifacts

The artifacts are hosted on an anonymous repository to maintain the integrity of the double-blind review process. Reviewers can access the repository via the following URL:

https://github.com/sysuWenchao/PPFE

This repository contains no personally identifiable information. Upon acceptance, the code will be migrated to a permanent, public GitHub repository, and a persistent DOI will be generated via Zenodo.

C.3 Completeness and Reproducibility

We confirm that the provided artifacts are sufficient to reproduce all experimental results reported in this paper. No proprietary datasets or restricted libraries are required. The experiments were conducted on standard x86-64 hardware, and the software dependencies are all open-source and publicly available.

□