Improved TensorPIR: Single-Server PIR with Lower Communication Cost

Yingchu Lv[1], Yanbin Pan[1], and Huaxiong Wang[2]

[1] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China
lvyingchu@amss.ac.cn
panyanbin@amss.ac.cn
[2] Nanyang Technological University, Singapore
hxwang@ntu.edu.sg

Abstract. Private Information Retrieval (PIR) is of growing importance in privacy-preserving data access, as it enables users to retrieve information from databases without revealing their query content, thereby aligning with modern data protection and regulatory standards. State-of-the-art schemes, such as **HintlessPIR** and **TensorPIR** proposed by Li *et al.* at CRYPTO 2024, leverage lattice-based cryptography for efficient and privacy-preserving data retrieval. HintlessPIR achieves a communication complexity of $O(N^{1/2})$, which remains suboptimal for large databases. To further reduce communication overhead, the same work introduces TensorPIR, lowering the asymptotic complexity to $O(N^{1/3})$. However, this improvement requires larger parameters and more CRT moduli, leading to a practical communication cost that is not significantly smaller than that of HintlessPIR.

In this work, we propose a new framework that rethinks the encryption strategy for the index, reducing both communication and computation costs through fewer CRT moduli. In experiments on 16 GB, 32 GB, 64 GB, and 128 GB databases, our total communication cost drops to as low as 45.5% of TensorPIR's. Theoretically, as N grows, our query and answer sizes are reduced to 36.9% and 22.2% of TensorPIR's, respectively. Compared with HintlessPIR, our scheme achieves lower theoretical communication complexity, leading to substantially smaller practical communication for large N. Moreover, our total online time is reduced to 28.9%–56.1% of HintlessPIR's.

1 Introduction

In a private information retrieval (PIR) protocol, the server holds a database containing N records, and the client wishes to retrieve a specific record without revealing which record is being accessed. Since this problem was introduced [1], PIR has become a key building block for privacy-preserving [2,3,4,5].

Modern PIR requires small communication overhead. Under modest cryptographic assumptions, the total communication required to fetch a database

record grows only polylogarithmically with the number N of records [6,7,8]. However, these PIR schemes are computationally expensive: the server must process every bit of the database to answer even a single client query [9]; otherwise, the scheme may leak information about which records the client is not interested in. Thus, a fundamental limitation on the throughput of PIR schemes—that is, the ratio between the database size and the server's time to answer a query—is the speed at which the server can read the database from memory.

One approach to improving efficiency is to consider multi-server PIR, where the database is replicated across multiple non-colluding servers and the client interacts with them to privately retrieve the desired record [10,11,12,13,6,8,14]. When the client can communicate with multiple database servers, there exist schemes with server-side throughput of up to 11.5 GB/s per core, or 93% of the memory bandwidth. However, these schemes rely on the strong assumption that the servers are non-colluding.

In the single-server PIR setting, an emerging paradigm that enables concrete reductions in the server's online computational cost (even achieving sublinear computation) is *preprocessing PIR*. In this model, the server and client preprocess the database to generate auxiliary information, which is stored and later used during query processing to reduce online communication and computation. In many constructions, the preprocessing is client-specific[15,10,16,17]—that is, it depends on private input from the client and must therefore be generated through interaction. On the client side, the preprocessing output depends on the database and is often referred to as a *hint*, as it provides partial information about the database content.

Related work

At USENIX Security '23, Henzinger *et al.* [18] proposed a scheme called *SimplePIR*, where the hint is independent of any client-specific information. It can therefore be computed solely by the server and reused across multiple clients. SimplePIR demonstrated that a database-dependent (but client- and query-independent) hint can significantly accelerate the server's answering process. Moreover, the initial cost of transmitting the database-dependent hint can be amortized over multiple queries made by the same client. Provided that each client issues a sufficiently large number of queries, the overall communication cost can be reduced. The same paper also introduces a second construction, *DoublePIR*, which theoretically reduces the hint size to be independent of the number of database records. However, the hint still heavily depends on the size of each individual record. For example, when records exceed 256 B in size, DoublePIR often requires hints that are several times larger than the entire database. Despite their advantages, both SimplePIR and DoublePIR share a major drawback: the client must download a hint that can be tens of megabytes in size for databases measured in gigabytes. If the client only makes a single query, this hint download dominates the total communication cost. This limitation poses challenges in several settings. For instance, database-dependent storage on the client side can be prohibitive, especially if the client intends to query multiple

databases. For storage-constrained devices, this may require evicting previously downloaded hints before issuing enough queries to amortize their cost. Additionally, when the database is updated, all existing hints must be recomputed to maintain correctness. For highly dynamic databases, this introduces significant communication overhead, as the updated hints must be redistributed to all clients. These communications are unrelated to client queries and therefore work against the amortization benefits. Such dynamically updating databases are common in domains like real-time data, e.g., databases of live stock prices.

HintlessPIR [19] proposed a solution that eliminates the hint. The authors observed that the hint (a matrix) in SimplePIR was only used to multiply with a secret vector held by the client, and pointed out that sending the hint solely for this operation is unnecessary. To address this, they replaced the hint with linear homomorphic encryption (RLWE encryption). In their scheme, instead of sending a hint in the offline phase, the server transmitted an encryption of the secret vector during the online phase. Upon receiving the ciphertext, the server computes an encryption of the product of the hint and the secret vector, which is then returned to the client. However, using linear homomorphic encryption increases computational costs, as most operations involve expensive polynomial multiplications. To alleviate this, they introduced the Number Theoretic Transform (NTT) and moved some operations in the linear homomorphic encryption scheme to the offline phase. The same paper also proposed another scheme called TensorPIR, which can be considered an iteration of HintlessPIR. Theoretically, TensorPIR achieves better communication complexity than HintlessPIR. However, the parameters in TensorPIR are significantly larger than those in HintlessPIR, resulting in higher actual communication costs in TensorPIR compared to HintlessPIR when the database is not sufficiently large.

Our Contributions

Our work builds upon TensorPIR and HintlessPIR [19] and proposes a novel PIR framework. In HintlessPIR, the client arranges the database into a matrix. To retrieve the element located in the v-th column, the client sends LWE encryptions of v under the secret key s, along with an RLWE encryption of s to the server. The server then performs a homomorphic computation to obtain an RLWE encryption of $DBAs$ and directly computes DBb, where b is the LWE encryption of v with the secret key s. The client then computes $DB(As-b) \approx DBv$.

TensorPIR can be seen as an iteration of HintlessPIR. In TensorPIR, the client arranges the database into multiple matrices. To retrieve the element located in the v_2-th column of the v_1-th matrix, the client sends LWE encryptions of v_1 and v_2, along with an RLWE encryption of the LWE secret key to the server. The server then performs a homomorphic computation to obtain an RLWE encryption of $DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)]$, where b_1, b_2 are LWE encryptions of v_1, v_2 with the same secret key s. The encryption is then returned to the client for decryption to recover the desired record.

Furthermore, both HintlessPIR and TensorPIR require the ring $\mathbb{Z}_Q$ where the LWE ciphertexts lie to be NTT-friendly. When Q is not NTT-friendly, it becomes

necessary to apply the Chinese Remainder Theorem (CRT) to reconstruct $DB \cdot (b - As) \bmod Q = DB \cdot [(b - As)] \bmod \prod_{i=1}^{k} Q_i$ in HintlessPIR and $DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)] \bmod Q = DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)] \bmod \prod_{i=1}^{k} Q_i$ in TensorPIR. This requires the selection of several Q_i's.

Shortcoming of HintlessPIR in Communication: In HintlessPIR, the size of the answer is proportional to the number of columns in the database, and the size of the query is proportional to the number of rows. This results in a communication complexity of at least $O(N^{1/2})$, causing the communication size to grow rapidly as the database size increases.

Shortcoming of TensorPIR in Communication: In TensorPIR, due to the change in the arrangement of the database, the theoretical communication complexity can be reduced to $O(N^{1/3})$. However, due to the introduction of homomorphic operations, the RLWE parameters in TensorPIR need to be larger. Additionally, in TensorPIR, more CRT moduli are required. In [19], the authors chose 6 CRT moduli Q_i, with an RLWE dimension of 2^{13} and a ciphertext modulus of 2^{144}. In contrast, HintlessPIR only requires 2 CRT moduli, with an RLWE dimension of 2^{12} and a ciphertext modulus of 2^{90}. As a result, although TensorPIR achieves better asymptotic communication complexity than HintlessPIR in theory, it does not show any communication advantage over HintlessPIR in practice.

We believe that the core ideas of HintlessPIR are twofold:

1. Using RLWE for homomorphic computation of matrix-vector multiplication.
2. Using LWE for matrix compression.

However, in TensorPIR, the first LWE encryption is unrelated to matrix compression. Based on this observation, we propose our scheme. In our scheme, the client sends only an RLWE encryption of v_1 and an LWE encryption of v_2 to the server. The server then directly computes an RLWE encryption of $DB \cdot [v_1 \otimes (b_2 - A_2 s)]$. This modification reduces the number of transmitted messages. In addition, our new framework reduces the number of CRT moduli from 6 to 2. We also provide a more rigorous bound on the RLWE ciphertext modulus q for both TensorPIR and our scheme. While TensorPIR offers only an empirical estimate of q, we present a formal proof establishing the correctness of our chosen q. Notably, even when setting the quality parameter γ to 2^{16}, our required modulus q is smaller than that used in TensorPIR.

All of these improvements contribute to a more efficient protocol. Compared with TensorPIR, our protocol achieves substantially lower communication cost in practice; in our experiments, the total communication cost is reduced to 62.8%, 65.0%, 55.9%, and 45.5% of TensorPIR's for 16 GB, 32 GB, 64 GB, and 128 GB databases, respectively. Theoretically, as N becomes sufficiently large, our query size can be reduced to 36.9% of TensorPIR's and our answer size can be reduced to 22.2% of TensorPIR's. Compared with HintlessPIR, our protocol achieves lower total communication cost on larger databases, reaching 84.9% of HintlessPIR's total communication at 64 GB and 68.5% at 128 GB. Additionally, our total online time can be reduced to 28.9%–56.1% of that of HintlessPIR since our query time is much smaller than that of HintlessPIR.

2 Notations and Preliminaries

2.1 Symbols

We regard database $DB = [DB_1, DB_2, ..., DB_{N^{1/3}}] \in \mathbb{Z}_p^{N^{1/3} \times N^{2/3}}$. LWE encryption is based on parameters $(p, Q, n_{\mathsf{lwe}}, \chi_\sigma)$ and RLWE encryption is based on parameters $(Q, q, n_{\mathsf{rlwe}}, \chi_\sigma^{n_{\mathsf{lwe}}})$, where the parameters denote the plaintext modulus, ciphertext modulus, dimension, and the error and secret key distributions. χ_σ is a centered binomial $\sum_{i=1}^{\sigma^2} X_i - X_i'$ where X_i and X_i' are uniform on $0, 1$. In this paper, we will regard $n = n_{\mathsf{lwe}} = n_{\mathsf{rlwe}}$ in the following discussion for simplicity. u_i is a vector which is 1 in the i -th coordinate and 0 in the others. $\mathrm{Enc}_v(s(x))$ means RLWE encryption for $s(x)$ with secret key v. Sometimes we omit the secret v and just write $\mathrm{Enc}(s)$. For $\mathrm{Enc}(s)=(a(x), b(x))$, we write $f(x)\mathrm{Enc}(s(x))=(f(x)a(x), f(x)b(x))$ which is an RLWE encryption for $f(x)s(x)$. For a matrix A of size $m \times n$, we define

$$\mathrm{diag}_j(A) = \left(a_{i \bmod m,\ (j+i) \bmod n}\right)_{i=0}^{n-1},$$

2.2 Private Information Retrieval (PIR)

First, we will give the definition of PIR.

Definition 1. *A Single-Server PIR scheme with preprocessing is a tuple of four algorithms: (S.setup, C.query, S.answer, C.recover).*

1. *S.setup(DB) $\longrightarrow$ (hint_c, hint_s): Given a $DB = (d_{i,j}) \in \mathbb{Z}_q^{n_1 \times n_2}$, output a client hint and a server hint.*
2. *C.query((i, j), hint_c)$\longrightarrow$ (query, state_c): Given an index (i, j) and the client hint, output a query and a secret client state.*
3. *S.answer(DB, query, hint_s)$\longrightarrow$ans: Given the database DB, a client query and a server hint, output an answer ans.*
4. *C.recover(state_c, hint_c, ans)$\longrightarrow$ d: Given a secret client state, a client hint and an answer, output a record $d \in DB$.*

- **Correctness**: Let $\delta \in [0, 1]$. A single-server PIR scheme with preprocessing is said to be $(1 - \delta)$-correct if, for any database $DB \in \mathbb{Z}_Q^{n_1 \times n_2}$, and for any index $(i, j) \in [n_1] \times [n_2]$, we have:

$$\Pr\left[d_{ij} = d \,\middle|\, \begin{array}{c} \mathrm{S.setup}(DB) \longrightarrow (\mathrm{hint_c}, \mathrm{hint_s}) \\ \mathrm{C.query}((i,j), \mathrm{hint_c}) \longrightarrow (\mathrm{query}, \mathrm{state_c}) \\ \mathrm{S.answer}(DB, \mathrm{query}, \mathrm{hint_s}) \longrightarrow \mathrm{ans} \\ \mathrm{C.recover}(\mathrm{state_c}, \mathrm{hint_c}, \mathrm{ans}) \longrightarrow d \end{array} \right] = 1 - \delta$$

- **Security**: A Single-Server PIR scheme with Preprocessing is said to be $(1-\epsilon)$ secure if for all (i_1, j_1) and $(i_2, j_2) \in [n_1] \times [n_2]$, we have:

$$\begin{aligned} &\Pr[\mathcal{A}(1^N, \text{query})=1 \mid \text{query}((i_1, j_1), \text{hint}_c) \longrightarrow (\text{query}, \text{state}_c)] \\ &- \Pr[\mathcal{A}(1^N, \text{query})=1 \mid \text{query}((i_2, j_2), \text{hint}_c) \longrightarrow (\text{query}, \text{state}_c)] \leq \epsilon. \end{aligned}$$

2.3 Number Theoretic Transform

In this section, we will give a brief description of Number Theoretic Transform (NTT). By using NTT, we can improve the computation complexity for polynomial multiplication from $O(n^2)$ to $O(n\log n)$.

Theorem 1. *[20] When n is a power of 2 and q is a prime such that $q \equiv 1 \pmod{2n}$, we have a ring isomorphism: $\mathbb{Z}_q[X]/(1+X^n) \cong \mathbb{Z}_q[X]/(X-z) \times \mathbb{Z}_q[X]/(X-z^3)\cdots\times\mathbb{Z}_q[X]/(X-z^{2n-1}) \cong \mathbb{Z}_q^n$ where $z^{2n} \equiv 1 \pmod{2n}$ but $z^n \not\equiv 1 \pmod{2n}$. The mapping NTT maps $v(X)$ to $(v(z), v(z^3), \cdots, v(z^{2n-1}))$. The polynomial multiplication corresponds to the component-wise product for vectors. We call this prime q an NTT-friendly prime.*

Notice $\mathbb{Z}_{2n}^*$ can be generated by -1 and 5, so we can change the order in the vector and get the theorem.

Definition 2. *Let*

$$a=\begin{pmatrix} a_0 & a_1 & \cdots & a_{n/2-2} & a_{n/2-1} \\ a_{n/2} & a_{n/2+1} & \cdots & a_{n-2} & a_{n-1} \end{pmatrix}.$$

We define:

$$per(a)=\begin{pmatrix} a_{n/2} & a_{n/2+1} & \cdots & a_{n-2} & a_{n-1} \\ a_0 & a_1 & \cdots & a_{n/2-2} & a_{n/2-1} \end{pmatrix},$$

$$rot(a)=\begin{pmatrix} a_1 & a_2 & \cdots & a_{n/2-1} & a_0 \\ a_{n/2+1} & a_{n/2+2} & \cdots & a_{n-1} & a_{n/2} \end{pmatrix}.$$

Theorem 2. *There exists a ring isomorphism between $\mathbb{Z}_q[X]/(1+X^n)$ and $\mathbb{Z}_q^n$ with the mapping NTT from $\mathbb{Z}_q[X]/(1+X^n)$ to $\mathbb{Z}_q^n$. For any $v(x) \in \mathbb{Z}_q[X]/(1+X^n)$, we have NTT($v(x^5)$)=rot(NTT($v(x)$)), NTT($v(x^{-1})$)=per(NTT($v(x)$)).*

Proof. $\mathbb{Z}_{2n}^*$ can be generated by $\langle -1, 5\rangle$. We put $v(z^{5^i})$ at the ith coordinate and $v(z^{-5^i})$ at the $(i+n/2)$th coordinate.

2.4 Sub-Gaussian Analysis

When using (R)LWE-based encryption, the ciphertext modulus must be sufficiently large to ensure correct decryption—specifically, it must exceed the infinity norm of the error term. In our protocol, we do some operations on the encryption that increase the error. We need to compute the bound for the new error to select a parameter. A simple way to compute the bound is using the inequality: $\|a(x)b(x)\|_\infty \le n\|a(x)\|_\infty\|b(x)\|_\infty$ for $a(x), b(x)$ in $\mathbb{Z}_q/(1+x^n)$. But this upper bound is too strict. The coordinates of the polynomial come from Gaussian distribution. For $\mathcal{X} - \mathcal{N}(0,\sigma^2)$, we have $Pr[|X| > t] < 2e^{-t^2/2\sigma^2}$. It seems that we can use this property to give a relaxed bound for the error. In this part, we will introduce Sub-Gaussian analysis to get this relaxed bound for the error.

We first give the definition of the ψ norm. It gives a bound for the tail for a distribution.

Definition 3. *Suppose X is a random variable. We define*

$$\|X\|_{\psi_2} = inf\left\{\sigma\middle|Pr[|X| > t] < 2e^{-t^2/2\sigma^2}, \forall.t\right\},$$
$$\|X\|_{\psi_1} = inf\left\{\sigma\middle|Pr[|X| > t] < 2e^{-t/2\sigma}, \forall.t\right\},$$
$$\|X\|_{\psi_\infty} = inf\left\{\sigma\middle|Pr[|X| > \sigma] = 0\right\}.$$

For a random vector $\boldsymbol{x}$, we define $\|\boldsymbol{x}\|_{\psi_j} = max_i|x_i|_{\psi_j}$.

We choose the secret key and error from χ_σ. It can be considered as a Gaussian distribution from $\mathcal{N}(0, \frac{\sigma^2}{2})$, so we have $\|\chi_\sigma\|_{\psi_2} < \sigma$.

Lemma 1. *[19] Let $X_1, X_2, \cdots, X_n$ be n random variables. Then*

$$\|X_1 + X_2 +, \cdots, +X_n\|_{\psi_2} \leq \sum_{i=1}^{n} |X_i|_{\psi_2}.$$

If $X_1, X_2, \cdots, X_n$ are n independent random variables. Then

$$\|X_1 + X_2 + \cdots + X_n\|_{\psi_2} \leq \sqrt{\sum_{i=1}^{n} |X_i|_{\psi_2}^2}.$$

The following lemma gives the ψ norm inequality when we do multiply. It is the main lemma used in computing the bound for the error.

Lemma 2. *[19] Suppose $f, g \in \mathbb{Z}[X]/(1 + X^n)$ and the coefficients of f, g are random vectors.*

- *If $\|f\|_{\psi_2} \leq \infty$ and $\|g\|_{\psi_\infty} \leq \infty$, we have*
$$\|fg\|_{\psi_2} \leq n\|f\|_{\psi_2}\|g\|_{\psi_\infty}.$$
- *If $\|f\|_{\psi_\infty} \leq \infty$ and $\|g\|_{\psi_\infty} \leq \infty$, we have*
$$\|fg\|_{\psi_\infty} \leq n\|f\|_{\psi_\infty}\|g\|_{\psi_\infty}.$$
- *If all the coefficients of f and g are independent, $\|f\|_{\psi_2} \leq \infty$ and $\|g\|_{\psi_\infty} \leq \infty$, we have*
$$\|fg\|_{\psi_2} \leq \sqrt{n}\|f\|_{\psi_2}\|g\|_{\psi_\infty}.$$

The following lemma gives the relationship between the infinity norm and ψ norm. It is used in selecting the parameter.

Lemma 3. *[19] Let $\boldsymbol{u}$ be an n-dimensional random variable of bounded ψ_2-norm. Then for any δ*

$$\Pr[\|\boldsymbol{u}\|_\infty > \sqrt{\ln(1 + n\delta^{-1})}\|\boldsymbol{u}\|_{\psi_2}] \leq \delta.$$

In our subsequent discussion, we will assume that all components of the error are mutually independent.

3 (R)LWE

(R)LWE is the main mathematical problem used to construct our protocol. In this section, we will first give the definition of RLWE encryption. Then, we will discuss two other technologies related to RLWE that we use in our protocol.

3.1 (R)LWE Encryption

Regev introduced a secret-key encryption scheme in [21] based on the LWE assumption with parameters (n, Q, χ). The encryption for $m \in \mathbb{Z}_p$ with secret key s is given by $(a, b = a^T s + e + \frac{Q}{p}m)$, where a, s are uniformly chosen from $\mathbb{Z}_q^n$, and e is sampled from χ. For a person who possesses the secret key s, they can compute $b - s^T a = \frac{Q}{p}m + e$. As long as $||e||_{\psi_\infty} < \frac{Q}{2p}$, the message m can be recovered. RLWE encryption is a variant of LWE, and we provide its formal definition below.

Definition 4 (RLWE encryption). *[22] Let $Q, q \in \mathbb{N}$ and n be a power of 2. RLWE encryption is defined by the following three algorithms:*

- ***KeyGen**: Sample $v(x) \leftarrow \chi^n$.*
- ***Enc**$_v(m)$: Sample $a(x) \leftarrow \mathbb{Z}_q[x]/(1+x^n)$ and $e(x) \leftarrow \chi^n$. Let $ct = [a(x), b(x) = a(x)v(x) + e(x) + \frac{q}{Q}m]$.*
- ***Dec**$_v(ct)$: $m = \left\lfloor \frac{b(x) - a(x)v(x)}{\frac{q}{Q}} \right\rceil$.*

If $|e(x)|_{\psi_\infty} \leq \frac{q}{2Q}$, the message m can be decrypted correctly, and the security is based on the LWE assumption.

3.2 Key switching

Key-switching is a technique used to switch a secret key from v to v', with the error increasing [23]. In this section, we will describe RLWE key-switching and provide a norm bound for the new error.

Definition 5. *A $\mathbb{Z}_Q$ gadget g of size l and quality γ is a vector in $\mathbb{Z}_Q^l$ and a mapping g^{-1} from $\mathbb{Z}_q$ to $\mathbb{Z}_Q^l$ such that for any $x \in \mathbb{Z}_q$, we have $\langle g^{-1}(x), g\rangle = x$ and $|g^{-1}(x)|\infty \leq \gamma$.*

Definition 6. *For a polynomial $a(x) = \sum_{j=0}^{n-1} a_j x^j \in \mathbb{Z}_q[X]$, we define that $g_i^{-1}(a(x)) = \sum_{j=0}^{n-1} u_i^T g^{-1}(a_j)x^j \in \mathbb{Z}_q[x]$ where u_i is a unit vector with 1 in the i-th position and 0 elsewhere. It is obvious that $\sum_{i=0}^{l-1} g_i^{-1}[a(x)]g_i = a(x)$.*

Definition 7. *Suppose g is a $\mathbb{Z}_Q$ gadget of size l and quality γ. A key-switch key (KSK) from $v(x)$ to $v'(x)$ with g is a collection of RLWE encryptions ksk_i, where $ksk_i = [a_i(x), b_i(x) = a_i(x)v'(x) + e_{ksk_i}(x) + g_i v(x)]$ for $i \in [l]$.*

Theorem 3. *Suppose $[a(x), b(x)]$ is an RLWE encryption for m with secret key $v(x)$. Then $[A(x), B(x)] = [0, b(x)] - \sum_{i=0}^{l-1} g_i^{-1}[a(x)]ksk_i$ is an RLWE encryption with secret $v'(x)$ and error $e_{new}(x) = e(x) + \sum_{i=0}^{l-1} g_i^{-1}[a(x)]e_{ksk_i}(x)$. If $|e(x)|_{\psi_2} < \sigma$, $|e(x)|_{\psi_\infty} < B$, $|e_i(x)|_{\psi_2} < \sigma_1$, and $|e_i(x)|_{\psi_\infty} < B_1$, then $|e_{new}(x)|_{\psi_2} \le \sqrt{nl}\gamma\sigma_1$ and $|e_{new}(x)|_{\psi_\infty} \le ln\gamma B_1$.*

Proof. Notice that $\sum_{i=0}^{l-1} g_i^{-1}[a(x)]g_i = a(x)$. Thus,

$$\begin{aligned} B(x) &= b(x) - \sum_{i=0}^{l-1} g_i^{-1}[a(x)]\left(a_i(x)v'(x) + e_{ksk_i}(x) + g_i v(x)\right) \\ &= e_{\text{new}}(x) + A(x)v'(x) + \Delta m. \end{aligned}$$

For the ψ_2 norm, we have

$$\begin{aligned} \|e_{\text{new}}(x)\|_{\psi_2} &\le \sqrt{\|e(x)\|_{\psi_2}^2 + \sum_{i=0}^{l-1} \|g_i^{-1}[a(x)]e_{ksk_i}(x)\|_{\psi_2}^2} \\ &\le \sqrt{\sigma^2 + ln\gamma^2\sigma_1^2} \\ &\approx \sqrt{nl}\gamma\sigma_1. \end{aligned}$$

For the ψ_∞ norm, by Lemma 2, we have

$$\begin{aligned} \|e_{\text{new}}(x)\|_{\psi_\infty} &\le \|e(x)\|_{\psi_\infty} + \left\| \sum_{i=0}^{l-1} g_i^{-1}[a(x)]e_{ksk_i}(x) \right\|_{\psi_\infty} \\ &\le B + ln\gamma B_1 \\ &\approx ln\gamma B_1. \end{aligned}$$

This completes the proof.

Remark 1. In the following, we use σ, B to denote the ψ_2, ψ_∞ norms of errors from all initial RLWE and LWE encryptions, and use σ_1, B_1 to denote those of errors from all KSKs.

3.3 Key switching with preprocessing

Suppose $\text{Enc}(m) = [a(x), b(x)]$ is an RLWE encryption. We call $a(x) = \alpha(\text{Enc}(m))$ the random part, and $b(x) = \beta(\text{Enc}(m))$ the secret part. The random part of an RLWE encryption can be generated in a preprocessing phase, as it is independent of the message m. Thus, some computations can be performed in advance.

$\text{Preproc}_{\text{ks}}\big(\text{NTT}(a_i)_{i\in[l]}, \text{NTT}(a)\big)$**:** Compute and output

$$g = \left\{ \text{NTT}\big(g_i^{-1}[a]\big)_{i\in[l]},\ \sum_{i=0}^{l-1} \text{NTT}\big(g_i^{-1}[a]\big)\text{NTT}\big(a_i\big) \right\}.$$

Apply$_{ks}$**(NTT**$((b_i)_{i\in[l]})$**, NTT**(b)**,** g**)**: Compute and output $\big[0, \mathrm{NTT}(b)\big] - \sum_{i=0}^{l-1} \mathrm{NTT}\big(g_i^{-1}[a]\big)\big[\mathrm{NTT}(a_i), \mathrm{NTT}(b_i)\big]$.

Computation cost: In **Preproc**$_{ks}$, we perform nl multiplications, l NTTs, 1 iNTT, and n decompositions. In **Apply**$_{ks}$, we perform ln multiplications.

3.4 Expand Algorithm

In this section, we introduce the Expand algorithm [24], which expands one RLWE encryption into n RLWE encryptions at the cost of increased error. We first describe a sub-algorithm required by Expand, and then present the full Expand algorithm treating the sub-algorithm as a black-box.

Theorem 4. *Suppose ct is an encryption of $m(x)$ under secret $v(x)$, and ksk is a key-switching key from $v(x^k)$ to $v(x)$. Then ct' is an encryption of $m(x^k)$ under the secret $v(x)$.*

Input: $ct = [a(x), b(x)]$, KSK, k.
Output: ct'.
Set $ct = [a(x^k), b(x^k)]$;
Compute $ct' = [0, b(x)] - \sum_{i=0}^{l-1} g_i^{-1}[a(x)] \cdot ksk_i$;
return ct'.

Algorithm 1: Sub

Theorem 5. *Suppose the input of the Expand algorithm is an RLWE encryption of a polynomial $f(x) = \sum_{i=0}^{n-1} f_i x^i$ under secret key $v(x)$, and a KSK of size l and quality γ whose errors have ψ_2-norm bounded by σ_1 and ψ_∞-norm bounded by B_1. Then the Expand algorithm outputs n RLWE encryptions of nf_i under the same secret $v(x)$. The ψ_2-norm of the error in each ciphertext ct_j is bounded by $l^{1/2}n^{3/2}\gamma\sigma_1$, and the ψ_∞-norm is bounded by $ln^2\gamma B_1$.*

Proof. We mainly focus on bounding the error here. The correctness of the algorithm is proved in Theorem 8. In the Expand algorithm, we perform m iterations, where $2^m = n$. After the i-th iteration, the error changes from e_{i-1} to

$$e_i = e_{i-1}(x) + e_{i-1}(x^k) + \sum_{j=0}^{l-1} g_j^{-1}[a(x)] e_{ksk_j}(x).$$

Then we have

$$|e_i|_{\psi_2}^2 \le 4|e_{i-1}|_{\psi_2}^2 + ln\gamma^2 |e_{ksk}|_{\psi_2}^2,$$
$$|e_i|_{\psi_\infty} \le 2|e_{i-1}|_{\psi_\infty} + ln\gamma |e_{ksk}|_{\psi_\infty}.$$

Thus, after m iterations, we obtain

$$|e_m|_{\psi_2}^2 \leq n^2|e|_{\psi_2}^2 + \frac{1}{3}(n^2-1)ln\gamma^2|e_{ksk}|_{\psi_2}^2 \leq \frac{1}{3}ln^3\gamma^2\sigma_1^2,$$

and

$$|e_m|_{\psi_\infty} \leq n|e|_{\psi_\infty} + (n-1)ln\gamma|e_{ksk}|_{\psi_\infty} \leq B_1ln^2\gamma.$$

Input: ct_0.
Output: ct_j.
for $i \in [\log n]$ **do**
  $k = n/2^i + 1$;
  for $b \in [2^i]$ **do**
    $ct_{2b} = ct_b + Sub(ct_b, k)$;
    $ct_{2b+1} = (ct_b - Sub(ct_b, k))X^{-k}$;
  end
end
return ct_j.

Algorithm 2: Expand

3.5 Expand algorithm with preprocessing

Key-switching technology is used in the Expand algorithm, enabling some computations to be performed during the preprocessing phase. In this part, we present an optimized version of the Expand algorithm with preprocessing, consisting of two sub-algorithms: $\mathbf{Preproc}_{ex}$, executed offline, and $\mathbf{Apply}_{ex}$, executed online. We denote by $ct_{i,b} = (A(x)_{i,b}, B(x)_{i,b})$ the b-th ciphertext in the i-th iteration, and by $(a_{i,s}, b_{i,s})$ the key-switching keys from $v(x^{n/2^i+1})$ to $v(x)$, where $i \in [\log n]$, $s \in [l]$, and $b \in [2^i]$.

Theorem 6. *The $\mathbf{Preproc}_{ex}$ algorithm performs $O(ln^2 \log n)$ operations in $\mathbb{Z}_q$ and outputs a string of length $(n^2l - nl)\log Q + n^2 \log q$ bits.*

Proof. The output of $\mathbf{Preproc}_{ex}$ includes the random part of the ciphertext in the $\log n$-th round, as well as the decomposition of the random part of the ciphertext in the first $\log n - 1$ rounds. Thus, the output size is given by

$$\sum_{i=0}^{\log n} 2^i ln \log Q + n^2 \log q.$$

The computational cost is primarily due to iNTT operations. In each round, the algorithm performs $2^i l$ iNTT operations.

Theorem 7. *The* $\boldsymbol{Apply}_{ex}$ *algorithm outputs a string of length* $n^2 \log q$ *bits and performs* $O(n^2 l)$ *operations in* $\mathbb{Z}_q$.

Proof. The output of Apply_{ex} includes the secret part of n RLWE ciphertext. All the computation in this algorithm is vector multiplication.

Theorem 8. *Suppose the input of* $\boldsymbol{Preproc}_{ex}$ *consists of the random part of the KSK from* $v(x^{n/2^i+1})$ *to* $v(x)$ *and the random part of the ciphertext* $ct = Enc_{v(x)}\left(\sum_{i=0}^{n-1} n^{-1} a_i x^i\right)$. *The output of* $\boldsymbol{Preproc}_{ex}$ *is denoted as* g. *The input of* $\boldsymbol{Apply}_{ex}$ *consists of* g, *the pairs* $(u_s, v_s)_{s\in[\log n]}$ *satisfying* $u_s v_s = \frac{n}{2^i} + i \pmod{2n}$, *the secret part of* ct, *and the secret parts of all KSKs from* $v(x^{n/2^i+1})$ *to* $v(x)$. *Define* $b = \sum_{i=0}^{\log n-1} 2^i b_i$, *then we have* $(A_{\log n,b}(x), B_{\log n,b}(x)) = Enc_{v(x)}(a_{b'})$, *where* $b' = \sum_{i=0}^{\log n-1} 2^i b_{\log n-1-i}$.

Proof. In the first round, ct_0 encrypts a polynomial consisting of the even-degree terms of the original polynomial, while ct_1 encrypts a polynomial composed of its odd-degree terms. Treating x^2 as the new variable, each subsequent round applies the expansion algorithm to these encrypted polynomials. The theorem follows by mathematical induction.

Input: $\text{NTT}(A_{0,0}(x))$, $\text{NTT}(a_{i,s}(x))_{i\in[\log n],s\in[l]}$.
Output: g.
for $i \in [\log n]$ **do**
  $k = n/2^i + 1$;
  for $b \in [2^i]$ **do**
    $\text{NTT}(A_{i+1,2b}(x)) = \text{NTT}(A_{i,b}(x)) - \sum_{s=0}^{l-1} \text{NTT}(g_s^{-1}[A_{i,b}(x^k)])\text{NTT}(a_{i,s}(x))$;
    $\text{NTT}(A_{i+1,2b+1}(x)) =$
      $\left[\text{NTT}(A_{i,b}(x)) + \sum_{s=0}^{l-1} \text{NTT}(g_s^{-1}[A_{i,b}(x^k)])\text{NTT}(a_{i,s}(x))\right]\text{NTT}(x^{-k})$;
  end
end
return $g = \begin{bmatrix} \text{NTT}(g_s^{-1}(A_{i,b}(x^k)))_{i\in[\log n],s\in[l],b\in[2^i],k=n/2^i+1}, \\ \text{NTT}(A_{\log n,b}(x))_{b\in[n]} \end{bmatrix}$.

Algorithm 3: Preproc$_{ex}$

4 Our construction

HintlessPIR and TensorPIR. TensorPIR [19] can be considered an iteration of HintlessPIR. The core idea behind HintlessPIR is to compute

$$(A, \text{Enc}(s)) \to \text{Enc}(As),$$

where $A \in \mathbb{Z}_Q^{m\times n}$ (we assume $n = m$ here; if $m \neq n$, A is divided into several square matrices and computed one by one), and $\text{Enc}(s(x)) \in \mathbb{Z}_q^n[x]/(1+x^n)$. By combining

$$As = \sum_{i=0}^{n-1} \text{diag}_i(A) \cdot \text{rot}^i(s),$$

Input: $\mathrm{NTT}(b_{i,s}(x))_{i\in[\log n],s\in[l]}$, $\mathrm{NTT}(B_{0,0}(x))$, g.
Output: $\mathrm{NTT}(B_{\log n,b}(x))_{b\in[n]}$.
for $i \in [\log n]$ **do**
  $k = n/2^i + 1$;
  for $b \in [2^i]$ **do**
    $\mathrm{NTT}(B_{i+1,2b}(x)) =$
      $\mathrm{NTT}(B_{i,b}(x)) + \mathrm{NTT}(B_{i,b}(x^k)) - \sum_{s=0}^{l-1} \mathrm{NTT}(g_s^{-1}[A_{i,b}(x^k)])\mathrm{NTT}(a_{i,s}(x))$;
    $\mathrm{NTT}(B_{i+1,2b+1}(x)) = \big[\mathrm{NTT}(B_{i,b}(x)) - \mathrm{NTT}(B_{i,b}(x^k)) +$
      $\sum_{s=0}^{l-1} \mathrm{NTT}(g_s^{-1}[A_{i,b}(x^k)])\mathrm{NTT}(b_{i,s}(x))\big]\mathrm{NTT}(x^{-k})$;
  end
end
return $\mathrm{NTT}(B_{\log n,b}(x))_{b\in[n]}$.

Algorithm 4: Apply$_{ex}$

with the iNTT mapping, we obtain

$$\mathrm{Enc}(\mathrm{iNTT}(As)) = \sum_{i=0}^{n-1} \mathrm{iNTT}(\mathrm{diag}_i(A)) \cdot \mathrm{Enc}(s(x^{5^i})),$$

where $\mathrm{iNTT}(s) = s(x)$. By applying the key-switching key (KSK) from $s(x^5)$ to $s(x)$ iteratively n times, along with $\mathrm{Enc}(s(x))$, we can derive $\mathrm{Enc}(\mathrm{iNTT}(As))$, with an error bound in the ψ_2 norm of at most $l^{1/2}n^2Q\gamma\sigma$.
In TensorPIR, we consider

$$DB = [DB_1, \ldots, DB_{N^{1/3}}] \in \mathbb{Z}_p^{N^{1/3}\times N^{2/3}},$$

where the record the client wants to retrieve is in the v_2-th column of DB_{v_1}. Initially, the client sends b_1, b_2, A_1, A_2, $\mathrm{Enc}(s)$, and $\mathrm{Enc}(A_1 s)$ to the server. The server then computes

$$\mathrm{Enc}(DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)])$$

locally, where $b_i - A_i s \approx u_{v_i} \in [0,1]^n$. Notice that

$$DB = \sum_{i=1}^{N^{1/3}} u_i^T \cdot DB_i,$$

so the expression $DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)]$ expands as:

$$\begin{aligned} &DB \cdot [(b_1 - A_1 s) \otimes (b_2 - A_2 s)] \\ &= \sum_{i=1}^{N^{1/3}} [u_i^T (b_1 - A_1 s)] \cdot [DB_i (b_2 - A_2 s)]. \end{aligned}$$

This simplifies to:

$$\begin{aligned} &\sum_{i=1}^{N^{1/3}} u_i^T b_1 \cdot DB_i b_2 - \sum_{i=1}^{N^{1/3}} u_i^T A_1 s \cdot DB_i b_2 \\ &- \sum_{i=1}^{N^{1/3}} u_i^T b_1 \cdot DB_i A_2 s + \sum_{i=1}^{N^{1/3}} u_i^T A_1 s \cdot DB_i A_2 s. \end{aligned}$$

The server needs to homomorphically compute

$$\mathrm{Enc}(u_i^T A_1 s \cdot DB_i b_2), \mathrm{Enc}(u_i^T b_1 \cdot DB_i A_2 s),$$
$$\mathrm{Enc}(u_i^T A_1 s \cdot DB_i A_2 s).$$

In the following homomorphic computations, we focus on scalar multiplication. For a scalar a and a vector s, we have

$$\mathrm{iNTT}(a \cdot s) = a \cdot \mathrm{iNTT}(s).$$

To simplify notation, we omit the explicit use of iNTT in our writing. Therefore, we use $\mathrm{Enc}(m)$ to represent $\mathrm{Enc}(\mathrm{iNTT}(m))$.

4.1 Motivation

In HintlessPIR, they use LWE encryption to reduce computing $\mathrm{Enc}(DBu_v)$ to computing $\mathrm{Enc}(DBAs)$ and DBb, which reduces the computation of $O(N)$ operations on $\mathbb{Z}_q$ to $O(\sqrt{N}n)$ operations on $\mathbb{Z}_q$ and N operations on $\mathbb{Z}_Q$. However, in TensorPIR, the LWE encryption for the first index is not necessary, we can directly write $DB_{v_1} e_{v_2}$ as:

$$\sum_{i=1}^{N^{1/3}} u_i^T u_{v_1} \cdot DB_i(b - As).$$

The client only needs to send $\mathrm{Enc}(s)$ and $\mathrm{Enc}(u_{v_1})$ to the server to complete the computation, thereby reducing the overall computational cost. Moreover, since

$$\sum_{i=1}^{N^{1/3}} u_i u_{v_1} \cdot DB_i b \quad \text{and} \quad \sum_{i=1}^{N^{1/3}} u_i u_{v_1} \cdot DB_i As$$

are bounded by Q and $nQ||s||_\infty$, respectively, we only require $\prod Q_i > nQB$ for CRT.

4.2 Optimized TensorPIR

The protocol consists of four algorithms:

- **Setup**: The client and server generate all the random parts of the encryption, and the server executes the **Preproc** algorithm for key generation, HintlessPIR, and extension. ($\mathbf{Preproc}_{hintless}$ means the setup phase in HintlessPIR and $\mathbf{apply}_{hintless}$ means the answering phase in HintlessPIR).
- **C.query**: The client generates an LWE encryption of u_{v_2} with secret key s, as well as an RLWE encryption of u_{v_1} and $\mathrm{iNTT}(s)$.
- **S.answer**: The server applies the corresponding algorithms for key generation, HintlessPIR, and extension, and uses the outputs to compute:

$$\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_i As),$$
$$\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_i b).$$

– **C.recover**: The client decrypts the received ciphertexts to recover the desired result.

Remark: In the following, when we refer to the multiplication of a vector of dimension $\sqrt[3]{N}$ with a vector of dimension n, we mean that the vector of dimension $\sqrt[3]{N}$ is first divided into $\sqrt[3]{N}/n$ vectors of dimension n, and then the multiplication is performed.

Input: $1^\lambda, n$.
Output: $seed, g_1, g_2$.
$\mathrm{NTT}(a_{0,k}) = H(seed|0|k|3)$;
$\mathrm{NTT}(a_1) = H(seed|0|0|1)$;
$\mathrm{NTT}(a_2) = H(seed|0|0|2)$;
$A = H(seed|0|0|0)$;
g_1=**Preproc**$_{Gen}(\mathrm{NTT}(a_{0,k})_{k\in[l]}, \mathrm{NTT}(a'_{0,k})_{k\in[l]}))$;
g_2=**Preproc**$_{hintless}(g_1,\ \mathrm{NTT}(a_1))$;
g_3=**Preproc**$_{ex}(g_2,\ \mathrm{NTT}(a_2))$;
return $seed, g_1, g_2, g_3$.

Algorithm 5: Setup

Input: *seed*.
Output: *query*.
$b = As + e + \frac{Q}{p}u_{v_2}$;
$b_1 = a_1 s' + e_1 + \frac{q}{Q}\mathrm{iNTT}(s)$;
$b_2 = a_2 s' + e_2 + \frac{q}{Q}u_{v_1}$;
$b_{0,k} = a_{0,k}s'(x^5) + e_{0,k}(x) + g[k]s'(x),\ k \in [l]$;
$b'_{0,k} = a'_{0,k}s'(x^{-1}) + e'_{0,k}(x) + g[k]s'(x),\ k \in [l]$;
return query $= [(\mathrm{NTT}(b_{0,k}), \mathrm{NTT}(b'_{0,k}))_{k\in[l]},$
$b,\ \mathrm{NTT}(b_1),\ \mathrm{NTT}(b_2)]$.

Algorithm 6: C.query

Theorem 9. *For the protocol above, we have:*
-Server's Preprocessing: $O((n + nl + \sqrt[3]{N})nl \log n)$ *operations in* $\mathbb{Z}_q$, nN *operations in* $\mathbb{Z}_Q$.
-Server's responding time: $O(nl\sqrt[3]{N} + n^2l + n^2l^2 + n\sqrt[3]{N^2} + \sqrt[3]{N}))$ *operations in* $\mathbb{Z}_q$, N *operations in* $\mathbb{Z}_Q$.
-Client upload: $\sqrt[3]{N}$ *in* $\mathbb{Z}_Q$ *and* $n + \sqrt[3]{N} + 2nl \log n$ *in* $\mathbb{Z}_q$.
-Client download: $5\sqrt[3]{N}$ *in* $\mathbb{Z}_q$.

Input: $seed, query, g_1, g_2, g_3$
Output: ct_1, ct_2
1 $\mathrm{NTT}(b_{j,k}) \leftarrow \mathbf{apply_{gen}}\big(\mathrm{NTT}(b_{0,k}), \mathrm{NTT}(b'_{0,k}), g_1\big)$;
2 $\mathrm{NTT}(\mathrm{Enc}(DB_iAs)) \leftarrow \mathbf{apply_{hintless}}\big(DB_ib,\ \mathrm{NTT}(b_1),\ \mathrm{NTT}(b_{0,k}),\ g_2\big)$;
3 $\mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1}))_{i\in[n]} \leftarrow \mathbf{apply_{ex}}\big(\mathrm{NTT}(b_{j,k}),\ \mathrm{NTT}(b'_{j,k}),\ \mathrm{NTT}(b_2),\ g_3\big)$;
4 $\mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_ib)) \leftarrow DB_ib \cdot \mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1}))$;
5 $\mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_iAs)) \leftarrow \mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1})) \cdot \mathrm{NTT}(\mathrm{Enc}(DB_iAs))$;
6 **return** $ct_1 = \sum_i \mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_ib)),\quad ct_2 = \sum_i \mathrm{NTT}(\mathrm{Enc}(u_i^T u_{v_1} \cdot DB_iAs))$.

Algorithm 7: S.answer

Input: ct_1, ct_2.
Output: m.
1 m=NTT($\lfloor \frac{\mathrm{Dec}(iNTT(ct_2)) - \mathrm{Dec}(iNTT(ct_1))}{\frac{q}{Q}} \rceil$);
2 **return** m.

Algorithm 8: C.recover

4.3 Correctness and Security Analysis of Our PIR

Theorem 10 (Correctness of Our PIR). *Our PIR is a correct scheme with probability at least* $1-\delta$ *if:*

$$q > 2\max_i \left(N^{1/6} Q_i e^i_{homo} \sqrt{\ln(1 + 4(N^{1/3}+n)\delta^{-1})} \right)$$
$$Q > 2N^{1/6} p^2 \sigma \sqrt{\ln(1 + 4(N^{1/3}+n)\delta^{-1})}$$
$$\prod_{i=1}^{k} Q_i > nQ$$

Proof. The correctness of our PIR is guaranteed by:

- The correct decryption of RLWE,
- The correct decryption of RLWE and LWE encryption,
- The plaintext computation never wraps around $\prod_{i=1}^{k} Q_i$.

For LWE encryption, we have:

$$DB[u_{v_1} \otimes (b - As)] = DB[u_{v_1} \otimes \Delta u_{v_2} + u_{v_1} \otimes e].$$

The error in the plaintext is $DB(u_{v_1} \otimes e)$, and it is easy to compute that:

$$\|DB(u_{v_1} \otimes e)\|_{\psi_2} \leq N^{1/6} p\sigma.$$

For RLWE encryption, the error increases quickly in the homomorphic multiplication operation. Just as the analysis in TensorPIR, the error in the homomorphic multiplication operation can be divided into four parts:

Error	Error bound
$DB_i As \cdot e_i$	$\|DB_i As \cdot e_i\|_{\psi_2} \leq l^{1/2} n^2 Q\gamma\sigma$
$u_i^T u_{v_1} \cdot e_i'$	$\|u_i^T u_{v_1} \cdot e_i'\|_{\psi_2} < l^{1/2} n^2 Q\gamma\sigma$
$t_i' \cdot e_i$	$\|t_i' \cdot e_i\|_{\psi_2} < l^{1/2} n^3 Q\gamma\sigma^3$
$t_i \cdot e_i'$	$\|t_i \cdot e_i'\|_{\psi_2} < l^{1/2} n^{3.5} Q^2\gamma\sigma^3$

The error $t_i \cdot e_i'$ accounts for the largest proportion of all errors. So, the error e in the homomorphic multiplication operation satisfies:

$$\|e_{home}^i\|_{\psi_2} \leq l^{1/2} n^{3.5} Q^2 \gamma \sigma^3.$$

For the CRT moduli Q_i, we have:

$$DB[u_{v_1} \otimes (b - As)] = \sum_{i=1}^{\sqrt[3]{N}} u_i u_{v_1} \cdot DB_i(b - As).$$

Notice that v_1 is a unit vector and $s \in [-1, 0, 1]^n$. It is easy to compute that the infinity norm of $DB[u_{v_1} \otimes (b - As)]$ can be bounded by Qn. So the first and second restrictions can be derived from Lemma 3. This ensures that the client can retrieve $DB[u_{v_1} \otimes u_{v_2}]$ correctly. The third restriction guarantees that:

$$DB[u_{v_1} \otimes (b - As)] \mod \prod_{i=1}^{k} Q_i = DB[u_{v_1} \otimes (b - As)].$$

Lemma 4 (Security of Our PIR). *Let $n_1, n_2, Q, n, q, N \in \mathbb{N}$ and let $\sigma > 0$. Assume that the Learning With Errors (LWE) problem $\mathsf{LWE}_{n_1,m,Q,\sigma}$ is hard, and that the Ring Learning With Errors (RLWE) problem $\mathsf{RLWE}_{n_2,q,\sigma}$ is also hard. Moreover, assume that RLWE is circularly secure. Then our PIR scheme is a secure Private Information Retrieval (PIR) scheme with preprocessing for an m-dimensional database.*

Proof. When querying our PIR on an index (v_1, v_2), the adversary observes:

- An RLWE encryption of u_{v_1},
- An RLWE encryption of the LWE secret key s,
- An LWE encryption of u_{v_2},
- Some RLWE key switch keys.

Under the decisional LWE and RLWE assumptions, and assuming circular security, the encryption of the secret key and the rotation keys are indistinguishable from random. Thus, a query on index (v_1, v_2) is computationally indistinguishable from a query on index (v_1', v_2'), ensuring the security of our PIR.

4.4 Implementation

We benchmarked our system using large-scale databases ranging from 8GB to 128GB. All databases consist of records with 8-bit entries. This design choice is based on the observation that in all three schemes—**HintlessPIR**, **TensorPIR**, and **our PIR**—the client ultimately retrieves a **column of records** from the database. To support large records, each record is decomposed into multiple 8-bit segments, which are then organized as columns in a newly constructed matrix.

We conducted all experiments on a machine equipped with an Intel Xeon w5-3425 processor running at 3.50 GHz and 256 GB of RAM. For fair comparison, we also benchmarked HintlessPIR under the same hardware and software environment.

Our PIR scheme can be seen as a combination of the expansion algorithm and HintlessPIR. Both can be performed with preprocessing. However, in real-world experiments, we executed the entire expansion algorithm online. Despite this, the time taken by the expansion algorithm accounts for less than 1% of the total response time. Additionally, we allowed the server to send all key-switching keys used in expand algorithm in the offline phase to reduce the response time.

In our protocol, the database has a size of $d_u \times d_v \times d_w$, which can be regarded as d_u databases, each of size $d_v \times d_w$. For HintlessPIR, we choose the LWE ciphertext modulus $Q = 2^{32}$, which allows for direct arithmetic computation in hardware. The LWE plaintext modulus is set to $p = 2^8$. If the database record size exceeds p, we partition the database into multiple databases with 8-bit records. This can be considered as performing PIR multiple times, but some operations are reused, improving efficiency. We set the LWE dimension to $n_1 = 1408$, the error standard deviation to $\sigma = 6.4$, and sample the LWE secret key from a uniform ternary distribution. For RLWE, we set the quality of the KSK $\gamma = 2^{16}$, the ciphertext modulus q is set to 138 bits, satisfying the condition:

$$\|e_{\text{homo}}\|_{\psi_2} \leq 2Q^2 \cdot n^{2.5} \cdot \sigma^3 \cdot n_{\text{rot}} \cdot \gamma.$$

This requires the RLWE dimension n_2 to be at least 2^{13}. The error standard deviation is set to 3.2 for both the RLWE secret key and the error terms. Both LWE and RLWE parameters achieve 128-bit security with up to 2^{30} samples [25].

The ℓ_∞-norm of $DB[v_1 \otimes (b - As)]$ is bounded by 2^{45}. Therefore, we select two NTT-friendly primes Q_i of 23 bits each to recover $DB[v_1 \otimes (b - As)]$ using the Chinese Remainder Theorem (CRT).

For the expansion algorithm, in the real implementation, we allow the client to send all $\log d_u$ key-switching keys (KSKs) in the offline phase and perform all expansion operations in the online phase.

Our scheme implementation is based on the Residue Number System (RNS) variant of the Brakerski/Fan-Vercauteren (BFV) fully homomorphic encryption (FHE) scheme, which supports both linear and multiplicative homomorphic operations. To reduce response time, we do not use the linearization operation in multiplicative homomorphic computations. Additionally, we apply the optimizations from HintlessPIR.

Our RLWE parameters provide 8192 slots, enabling the packing of four copies of $\mathbf{s}$ in the query ciphertexts if the block size is set to 1024. For each p_j, this packing strategy reduces the number of rotations to 255 while increasing the communication cost.

4.5 Communication Cost Comparison

In our scheme, the query consists of one LWE ciphertext b, multiple key-switching keys (KSKs), and two RLWE ciphertexts. The response consists of two RLWE ciphertexts. We provide a detailed analysis of the communication cost, taking into account blocking (block size = 1024) and the Chinese Remainder Theorem (CRT) in real experiments, as shown in Table 1. We regard the database in HintlessPIR with size $d_v \times d_w$).

Comparison with TensorPIR. In the TensorPIR scheme, the query includes two LWE ciphertexts, b_1 and b_2, and two RLWE encryptions. The response consists of a message in $\mathbb{Z}_Q$ and three ciphertexts. In TensorPIR, the parameters are set to $k = 6$ and $q = 144$. In the theoretical analysis, we assume $d_u = d_v = d_w = N^{1/3}$, all the $Q_i = B$, and Q, q in TensorPIR and our scheme are the same. As $N \to \infty$, our query size is approximately

$$\frac{3(\log_B n + \log_B Q)\log_2 q + 3\log_2 Q}{(3\log_B n + 9\log_B Q + \log_B N)\log_2 q + 6\log_2 Q}$$

of the TensorPIR query size. Meanwhile, our response size is about

$$\frac{78(\log_B n + \log_B Q)\log_2 q}{3\log_2 Q + 42(3\log_B n + 9\log_B Q + \log_B N)\log_2 q}$$

of TensorPIR's response size. When we set $N = 2^{50}, q = 2^{144}, Q = 2^{32}, B = 2^{23}$, our query size is approximately 36.9% of TensorPIR's, and our response size is approximately 22.2% of TensorPIR's.

Comparison with HintlessPIR. From Table 1, we observe that the query size is strongly correlated with d_w, while the answer size is primarily determined by d_v in both our PIR and HintlessPIR schemes. Moreover, this growth relationship appears to be approximately linear. In our PIR, we can adjust the value of d_u to reduce both d_v and d_w, which explains why the communication cost of our PIR is lower than that of HintlessPIR on larger databases.

We summarize the communication cost for different database sizes in Table 2. The communication cost is influenced by the layout of the database. In HintlessPIR, the database is arranged to minimize communication overhead. In contrast, our scheme does not adopt a layout optimized exclusively for communication, which would lead to a significant increase in answer time. Instead, the database is organized to balance both communication cost and answer time.

	Query Size	**Answer Size**
HintlessPIR	$(k+l)n\log_2(q) + d_w\log_2(Q)$	$d_v\log_2(Q) + kd_v\log_2(q)$
TensorPIR	$k(n + d_u)\log_2(q) + (d_u + d_w)\log_2(Q) + ln\log d_u\log_2(q)$	$d_v\log_2(Q) + 7kd_v\log_2(q)$
Ours	$k(n+d_u)\log_2(q)+d_w\log_2(Q)+ ln\log d_u\log_2(q)$	$5kd_v\log_2(q)$

Table 1: Comparison of communication costs in theory

Database		**128 GB**	**64 GB**	**32 GB**	**16 GB**
Hintless	Query Size (KB)	8750.77	8750.77	4556.46	4556.46
	Answer Size (KB)	12058.62	6029.31	6029.31	3014.66
	Total Size (MB)	20.81	14.78	10.59	7.57
Tensor	Query Size (KB)	11829.50	11829.50	11829.50	10780.93
	Answer Size (KB)	19480.58	10625.02	6195.97	6195.97
	Total Size (MB)	31.31	22.45	18.03	16.98
Ours	Query Size (KB)	10293.25	10293.25	10293.25	9244.67
	Answer Size (KB)	3956.74	2260.99	1413.12	1413.12
	Total Size (MB)	14.25	12.55	11.71	10.66

Table 2: Comparison of communication costs

4.6 Computation Cost Comparison

In the preprocessing phase, the most computationally expensive operation is computing DB_iA for $i \in [N^{1/3}]$, which requires performing nN operations over $\mathbb{Z}_Q$. Another source of computational overhead arises from computing the random part of the RLWE ciphertext; however, this overhead is negligible. Therefore, for databases of the same scale, the preprocessing time cost of our scheme is approximately the same as that of HintlessPIR.

In the online phase, the computational cost of computing DB_ib for $i \in [N^{1/3}]$ requires N operations over $\mathbb{Z}_Q$. The time cost for this part in our PIR is identical to HintlessPIR. Additionally, when computing all ciphertexts of the form $\mathrm{Enc}(\mathrm{rot}^i(s))$, where s is the LWE encryption secret key, although the RLWE dimension in our scheme is twice that of HintlessPIR, the number of required rotations is halved. As a result, the computational overhead for this step is also nearly the same in both schemes. The additional computational overhead in our scheme mainly stems from computing $\mathrm{Enc}(DB_iAs)$, executing the expansion algorithm, and performing homomorphic operations. These costs are closely related to the choice of the parameter d_v. Therefore, we set $d_v = 6$ in

our implementation to mitigate the impact of this overhead on the overall runtime. Moreover, as the database size increases, the relative proportion of this additional cost becomes increasingly smaller compared to the total execution time.

We also report the query time in Table 3. The communication cost is primarily influenced by the parameter d_w. Since our protocol adopts a smaller d_w, it results in a lower query time compared to HintlessPIR.

Note that since the original paper [19] does not provide the implementation of TensorPIR, and TensorPIR incurs even greater overhead than our scheme in the online phase, we did not run TensorPIR to measure its execution time.

Data Size	**HintlessPIR (s)**			**Our PIR (s)**		
	Preprocessing	**Query**	**Answer**	**Preprocessing**	**Query**	**Answer**
64GB	12236.3	229.4	9.1	11146.4	116.3	13.58
32GB	6135.4	117.6	4.8	5454.9	59.6	9.1
16GB	3193.4	120.2	2.5	2758.4	28.1	7.3

Table 3: Comparison of computation costs for HintlessPIR and Our PIR under varying data sizes.

5 Conclusion

In this paper, we present a new encryption framework built upon TensorPIR, which effectively reduces communication cost in practice and is theoretically more efficient in terms of computation cost.

Scheme	**Off. Comm.**	**Off. Comp.**	**On. Comm.**	**On. Comp.**
SimplePIR	$n\sqrt{N}$	nN	$\sqrt{N}$	N
HintlessPIR	$O(1)$	$nN + \tilde{O}(n\sqrt{N})$	$O(\sqrt{N}+n)$	$N + O(n\sqrt{N})$
TensorPIR	$O(1)$	$nN + \tilde{O}(n^2 + nN^{1/3})$	$O(N^{1/3}+n)$	$N + O(nN^{2/3})$
Our	$O(1)$	$nN + \tilde{O}(n^2 + nN^{1/3})$	$O(N^{1/3}+n)$	$N + O(nN^{2/3})$

Table 4: Comparison of PIR Schemes

References

1. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, p. 965–981, Nov. 1998. [Online]. Available: https://doi.org/10.1145/293347.293350

2. S. Angel and S. Setty, "Unobservable communication over fully untrusted infrastructure," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 551–569. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/angel
3. M. Backes, A. Kate, M. Maffei, and K. Pecina, "Obliviad: Provably secure and practical online behavioral advertising," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 257–271.
4. T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish, "Scalable and private media consumption with popcorn," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 91–107. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/gupta-trinabh
5. D. Kogan and H. Corrigan-Gibbs, "Private blocklist lookups with checklist," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 875–892. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/kogan
6. C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 402–414.
7. S. Angel, H. Chen, K. Laine, and S. Setty, "Pir with compressed queries and amortized query processing," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 962–979.
8. C. Gentry and Z. Ramzan, "Single-database private information retrieval with constant communication rate," in *Automata, Languages and Programming*, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 803–815.
9. A. Beimel, Y. Ishai, and T. Malkin, "Reducing the servers' computation in private information retrieval: Pir with preprocessing," *J. Cryptol.*, vol. 17, no. 2, p. 125–151, Mar. 2004. [Online]. Available: https://doi.org/10.1007/s00145-004-0134-y
10. A. Lazzaretti and C. Papamanthou, "Treepir: Sublinear-time and polylog-bandwidth private information retrieval from ddh," in *Advances in Cryptology – CRYPTO 2023*, H. Handschuh and A. Lysyanskaya, Eds. Cham: Springer Nature Switzerland, 2023, pp. 284–314.
11. R. Henry, "Polynomial batch codes for efficient it-pir," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, pp. 202 – 218, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:9564677
12. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995, pp. 41–50.
13. E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing: Improvements and extensions," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1292–1303. [Online]. Available: https://doi.org/10.1145/2976749.2978429
14. R. Ostrovsky and W. E. Skeith, "A survey of single-database private information retrieval: Techniques and applications," in *Public Key Cryptography – PKC 2007*, T. Okamoto and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 393–411.

15. H. Corrigan-Gibbs and D. Kogan, "Private information retrieval with sublinear online time," in *Advances in Cryptology – EUROCRYPT 2020*, A. Canteaut and Y. Ishai, Eds. Cham: Springer International Publishing, 2020, pp. 44–75.
16. M. Zhou, A. Park, W. Zheng, and E. Shi, "Piano: Extremely simple, single-server pir with sublinear server computation," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 4296–4314.
17. S. Patel, G. Persiano, and K. Yeo, "Private stateful information retrieval," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1002–1019. [Online]. Available: https://doi.org/10.1145/3243734.3243821
18. A. Henzinger, M. M. Hong, H. Corrigan-Gibbs, S. Meiklejohn, and V. Vaikuntanathan, "One server for the price of two: Simple and fast Single-Server private information retrieval," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 3889–3905. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger
19. B. Li, D. Micciancio, M. Raykova, and M. Schultz-Wu, "Hintless single-server private information retrieval," in *Advances in Cryptology – CRYPTO 2024*, L. Reyzin and D. Stebila, Eds. Cham: Springer Nature Switzerland, 2024, pp. 183–217.
20. D. Lei, D. He, C. Peng, M. Luo, Z. Liu, and X. Huang, "Faster implementation of ideal lattice-based cryptography using avx512," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 5, Sep. 2023. [Online]. Available: https://doi.org/10.1145/3609223
21. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 84–93. [Online]. Available: https://doi.org/10.1145/1060590.1060603
22. V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology – EUROCRYPT 2010*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23.
23. N. Genise, D. Micciancio, and Y. Polyakov, "Building an efficient lattice gadget toolkit: Subgaussian sampling and more," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 655–684.
24. H. Chen, I. Chillotti, and L. Ren, "Onion ring oram: Efficient constant bandwidth oblivious ram from (leveled) tfhe," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 345–360. [Online]. Available: https://doi.org/10.1145/3319535.3354226
25. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015. [Online]. Available: https://doi.org/10.1515/jmc-2015-0016