YsPIR: HE-Based Single-Server Private Information Retrieval with Low Communication Cost and High Throughput

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

[1] The Academy of Mathematics and Systems Science (AMSS), Chinese Academy of Sciences, China
379934963@qq.com
panyanbin@amss.ac.cn
[2] Nanyang Technological University, Singapore
hxwang@ntu.edu.sg

Abstract. We introduce **YsPIR**, a single-server private information retrieval (PIR) protocol that outperforms the state-of-the-art **KsPIR** (Luo, Liu, and Wang, CCS 2024) in terms of server response time and offline communication. We propose a novel First-dimension folding method to reduce the online time and reduce the public key used in the offline phase. Our approach offers three significant advantages: first, it allows the most resource-intensive computations to be preprocessed in the offline phase, thereby optimizing online response time; second, the offline communication is independent of the database and remains small; third, it maintains a relatively low communication cost under high throughput. We conduct comprehensive experiments to evaluate the concrete performance of YsPIR. The results show that YsPIR achieves approximately 1.64× higher online throughput and reduces offline communication by roughly 3.09× compared to KsPIR.

Keywords: Private Information Retrieval, RLWE

1 Introduction

In a *private information retrieval (PIR)* protocol, a server stores a database containing N records, and the client seeks to retrieve a specific record without disclosing which one is being accessed. Since the introduction of this problem [8], PIR has evolved into a crucial component for various privacy-preserving applications [4,13,16].

Modern PIR requires small communication. Under modest cryptographic assumptions, the total communication required to fetch a database record grows only polylogarithmically with the number N of records [6,3,11,18]. However, these PIR schemes are computationally expensive: the server must touch every bit of the database to answer even a single client query, as otherwise the scheme may leak information about which records the client is not interested in. One approach to improve efficiency is to consider multi-server PIR [14,7,5,6,11,23].

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 [6,11,23]. However, these schemes rely on the strong assumption that the servers do not collude. In the single-server PIR setting, an emerging paradigm that enables concrete reductions in the server's online computational cost 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 [9,27,24]—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.

1.1 Related Work

Early homomorphic-encryption-based PIR schemes [1,3,2] model the database as a d-dimensional hypercube with side lengths N_i. The user sends N_i RLWE ciphertexts per dimension, where only one encrypts 1 and the rest encrypt 0. The server homomorphically evaluates

$$\sum_{i_d=0}^{N_d-1} \mathrm{GSW}_d[i_d] \odot \cdots \sum_{i_1=0}^{N_1-1} \mathrm{GSW}_1[i_1] \odot \sum_{i_0=0}^{N_0-1} \mathrm{RLWE}_0[i_0] \cdot DB_{i_d i_{d-1} \cdots i_1 i_0},$$

and returns a single RLWE ciphertext. To control noise, early schemes refresh ciphertexts after each multiplication layer, introducing an expansion factor l and making the response size grow as l^d. Thus, they usually restrict d to 2 or 3.

Later works [22,20] introduced GSW encryption, enabling homomorphic multiplication with only linear noise growth and achieving $O(\log N)$ communication. Query packing and unpacking further reduce query communication from $O(\log N)$ to $O(\log N / n_{\mathrm{RLWE}})$. Subsequent improvements [20] used matrix-form RLWE ciphertexts to improve practical communication. However, schemes in this paradigm still require plaintext–ciphertext multiplications between database records and RLWE ciphertexts during First-dimension folding, limiting online efficiency. KsPIR [18] simplifies this folding step and improves throughput by 4× over [20].

Another line of work, represented by SimplePIR [15], improves throughput via preprocessing. The user sends a single LWE query query $= (A, b)$, and the server responds with answer $= (DBA, DBb)$. Since DBA can be precomputed offline, online computation only requires N multiplications modulo the ciphertext modulus. This yields high throughput, but the hint is large and database-dependent; for example, a 1GB database may require a 121MB hint, making SimplePIR less suitable for dynamic databases. HintlessPIR [17] and YPIR [21] reduce this dependency using LWE-to-RLWE techniques, at the cost of additional computation. Nevertheless, these schemes retain $O(\sqrt{N})$ communication, which becomes a bottleneck for large databases.

1.2 Our Contribution

Our scheme builds on OnionPIR. Its main online bottleneck is the First-dimension folding stage, which requires $O(N)$ polynomial multiplications over $\mathbb{Z}_q$. We replace this step with an LWE-based folding method, reducing it to N multiplications over a smaller ring $\mathbb{Z}_Q$, thereby improving throughput. In the offline phase, we use key-switching keys to support homomorphic matrix–ciphertext computation and reduce offline communication. Our contributions are summarized as follows:

- **Low communication.** Our scheme achieves $O(\log N)$ communication complexity. For databases up to 16 TB with 1 KB records, its total online communication is only 460 KB.
- **High online efficiency.** Our online computation remains $O(N)$. Among PIR schemes with $O(\log N)$ communication, our throughput is 1.64× that of KsPIR, 7.05× that of Spiral, and 13.74× that of OnionPIR. It reaches 24.62% of SimplePIR's throughput while providing much lower offline and online communication.
- **Low offline cost.** Our offline communication is 3.09×, 7.38×, 2.14×, and 207.69× smaller than KsPIR, Spiral, OnionPIR, and SimplePIR, respectively. It is also database-independent, avoiding repeated offline transmissions for dynamic databases.

2 Notations and Preliminaries

2.1 Symbols

In this paper, our LWE encryption is based on parameters $(p, Q, n_{\mathrm{LWE}}, \chi_\sigma)$, and RLWE encryption is based on parameters $(Q, q, n_{\mathrm{RLWE}}, \chi_\sigma^{n_{\mathrm{RLWE}}})$, where these parameters denote the plaintext modulus, ciphertext modulus, dimension, and the error and secret key distributions, respectively. The error distribution χ_σ is defined as a centered binomial distribution: $\chi_\sigma = \sum_{i=1}^{2\sigma^2}(X_i - X_i')$, where X_i and X_i' are sampled uniformly from $\{0,1\}$. We define $R = \mathbb{Z}[x]/(1+x^{n_{\mathrm{RLWE}}})$ and $R_q = \mathbb{Z}_q[x]/(1+x^{n_{\mathrm{RLWE}}})$. For a matrix A of size $m \times n$, we define $\mathrm{diag}_j(A) = \left(a_{(i-j) \bmod m,\ i}\right)_{i=0}^{n-1}$, which is a vector of length n. For a vector s, $\mathrm{rot}^i(s)$ denotes the vector obtained by left-rotating s by i positions. Let B denote the basis of the gadget vector used in GSW encryption and key-switching key. Here, "$\odot$" denotes the external product, and "$\cdot$" denotes the matrix-vector product. v_i is a unit vector with 1 at the i-th position and 0 elsewhere.

2.2 PIR Definition

We now formally define a single-server PIR scheme with preprocessing.

Definition 1. *A single-server PIR scheme with preprocessing consists of a tuple of four algorithms:*

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

– **Correctness:** Let $\delta \in [0,1]$. The scheme is $(1-\delta)$-correct if, for any $DB \in \mathbb{Z}_p^{n_1 \times n_2}$ and any $(i,j) \in [n_1] \times [n_2]$, we have:

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

– **Security:** The scheme is $(1-\varepsilon)$-secure if for all $(i_1,j_1), (i_2,j_2) \in [n_1] \times [n_2]$:

$$\begin{aligned} &\Pr\left[\mathcal{A}(1^N, \mathrm{query}) = 1 \mid \mathrm{query}((i_1,j_1), \mathrm{hint}_c) \rightarrow (\mathrm{query}, \mathrm{state}_c)\right] \\ &\quad - \Pr\left[\mathcal{A}(1^N, \mathrm{query}) = 1 \mid \mathrm{query}((i_2,j_2), \mathrm{hint}_c) \rightarrow (\mathrm{query}, \mathrm{state}_c)\right] \leq \varepsilon \end{aligned}$$

2.3 (R)LWE Encryption

Regev introduced a secret-key encryption scheme based on the LWE assumption [25].

Definition 2 (LWE Encryption [25]). *Let* $n_{LWE}, p, Q \in \mathbb{N}$ *and* χ *be an error distribution over* $\mathbb{Z}_Q$*. The LWE encryption scheme is defined as follows:*

– ***KeyGen****: Sample secret key* $s \leftarrow \mathbb{Z}_Q^{n_{LWE}}$*.*
– ***Enc***$_s(m)$*: Sample* $a \leftarrow \mathbb{Z}_Q^{n_{LWE}}$ *and* $e \leftarrow \chi$*. For* $m \in \mathbb{Z}_p$*, output ciphertext* $(a, b = s^T a + e + \frac{Q}{p}m)$*.*
– ***Dec***$_s(a,b)$*: Compute* $m = \left\lfloor \frac{b - s^T a}{Q/p} \right\rceil$*.*

Decryption succeeds if $\|e\|_\infty < \frac{Q}{2p}$*.*

Definition 3 (RLWE Encryption [19]). *Let* n_{RLWE} *be a power of* 2*,* $Q, q \in \mathbb{N}$*, and* χ *be an error distribution over* $\mathbb{Z}_q$*. The RLWE encryption scheme is defined as follows:*

– ***KeyGen****: Sample* $s(x) \leftarrow \chi^{n_{RLWE}}$*.*
– ***Enc***$_v(m)$*: Sample* $a(x) \leftarrow R_q$*,* $e(x) \leftarrow \chi^{n_{RLWE}}$*. For* $m \in R_Q$*, output ciphertext* $ct = [a(x), b(x) = a(x)s(x) + e(x) + \frac{q}{Q}m(x)]$*.*
– ***Dec***$_v(ct)$*: Output* $m = \left\lfloor \frac{b(x) - a(x)s(x)}{q/Q} \right\rceil$*.*

Decryption succeeds if $\|e(x)\|_\infty \leq \frac{q}{2Q}$*.*

2.4 GSW Encryption

The GSW encryption scheme was introduced in [12]. It is a fully homomorphic encryption (FHE) scheme with additive error growth during multiplication.

Definition 4 (Gadget). *Let B be the gadget base, and let $l = \left\lceil \frac{\log q}{\log B} \right\rceil$ denote the decomposition length. Define the gadget vector:*

$$g = \left(1, B, B^2, \ldots, B^{l-1}\right)^\top \in \mathbb{Z}^l$$

and the gadget matrix as follows:

$$G = \begin{pmatrix} g & 0 \\ 0 & g \end{pmatrix} \in \mathbb{Z}^{2l \times 2}.$$

Definition 5 (Gadget decomposition). *Let g be a gadget vector with base B. The gadget decomposition $g^{-1}(a)$ of $a \in \mathbb{Z}_q$ under the basis g is defined as follows:*

$$g^{-1}(a) = (a_0, \ldots, a_{l-1}) \quad \text{such that} \quad \sum_{i=0}^{l-1} a_i B^i = a,\ a_i < B.$$

For a polynomial $a(x) = \sum_{j=0}^{n-1} a_j x^j \in R_q$, we define

$$g_i^{-1}(a(x)) = \sum_{j=0}^{n-1} v_i^\top g^{-1}(a_j) x^j \in \mathbb{Z}_q[x],$$

where v_i is the unit vector with 1 in the i-th position and 0 elsewhere. It holds that:

$$\sum_{i=0}^{l-1} g_i^{-1}(a(x))\, g_i = a(x).$$

Definition 6 (GSW Encryption). *A GSW encryption of a plaintext polynomial $m \in R_Q$ with the secret key $s \in R_q$ is*

$$C = Z + m \cdot G,$$

where $Z \in R_q^{2l \times 2}$ is a matrix whose rows each consist of an RLWE encryption of 0 under the secret key s.

Remark: *A GSW encryption can be regarded as $2l$ RLWE ciphertexts without scaling the message, where the first l rows encrypt $(ms, Bms, \ldots, B^{l-1}ms)$, and the last l rows encrypt $(m, Bm, \ldots, B^{l-1}m)$, respectively.*

Definition 7 (External Product). *Let $GSW(m) = Z + m \cdot G$ be a GSW encryption of $m \in R_Q$, and $RLWE(x) = (a(x), b(x))$ be an RLWE encryption under the same secret key s. The external product is defined as:*

$$GSW(m) \odot RLWE(x) = \left(g^{-1}(a(x)), g^{-1}(b(x))\right) \cdot GSW(m).$$

Lemma 1. *Let $GSW(m_1) = Z + m_1 G$ and $RLWE(m_2) = (a(x), b(x))$ be under the same secret key s. Then*

$$GSW(m_1) \odot RLWE(m_2)$$

is an RLWE encryption of $m_1 \cdot m_2$ with error:

$$e(x) + \left(g^{-1}(a(x)),\ g^{-1}(b(x))\right) \cdot E,$$

where $E \in R_q^{2l}$ contains the RLWE encryption errors from Z, and $e(x)$ denotes the error in $RLWE(m_2)$.

2.5 Sub-Gaussian Analysis

When using (R)LWE encryption, the ciphertext modulus must be large enough to accommodate noise growth. A naïve upper bound uses:

$$|a(x)b(x)|_\infty \leq n|a(x)|_\infty |b(x)|_\infty,$$

but this is overly pessimistic. Since coefficients are drawn from a Gaussian distribution, we can leverage sub-Gaussian tail bounds. If $X \sim \mathcal{N}(0, \sigma^2)$, then:

$$\Pr[|X| > t] < 2e^{-t^2/2\sigma^2}.$$

Definition 8. *Let X be a random variable. Define:*

$$\|X\|_{\psi_2} = \inf\left\{\sigma \mid \Pr[|X| > t] < 2e^{-t^2/2\sigma^2},\ \forall t\right\},$$

$$\|X\|_{\psi_1} = \inf\left\{\sigma \mid \Pr[|X| > t] < 2e^{-t/2\sigma},\ \forall t\right\},$$

$$\|X\|_{\psi_\infty} = \inf\left\{\sigma \mid \Pr[|X| > \sigma] = 0\right\}.$$

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

It follows that for $X \sim \mathcal{N}(0, \sigma^2)$, we have:

$$\|X\|_{\psi_2} < \sigma.$$

Lemma 2 ([26]). *Let $X_1, \ldots, X_n$ be random variables. Then:*

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

and if X_i are independent:

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

Lemma 3 ([26]). *Let $f, g \in \mathbb{Z}[x]/(1+x^n)$ with random coefficients:*

- *If $\|f\|_{\psi_2} < \infty$ and $\|g\|_{\psi_\infty} < \infty$, we have $\|fg\|_{\psi_2} \leq n\|f\|_{\psi_2}\|g\|_{\psi_\infty}$.*
- *If both have bounded ψ_∞, we have $\|fg\|_{\psi_\infty} \leq n\|f\|_{\psi_\infty}\|g\|_{\psi_\infty}$.*
- *If coefficients are independent, we have $\|fg\|_{\psi_2} \leq \sqrt{n}\|f\|_{\psi_2}\|g\|_{\psi_\infty}$.*

Lemma 4 ([26]). *Let u be a random n-dimensional vector with bounded ψ_2-norm. Then for any $\delta > 0$:*

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

2.6 Key switching

Key-switching [10] is a technique used to switch a secret key from v to v', with the error increasing. In this section, we describe RLWE key-switching and provide a bound on the ψ_2-norm.

Definition 9. *Let g be a gadget with base B and decomposition length ℓ. A g-based key-switching key (from key $v(x)$ to $v'(x)$) is a collection of ℓ RLWE encryptions ksk where $ksk_i = Enc_{v'(x)}(g^i v(x))$ for $i \in [\ell]$.*

Theorem 1. *Suppose $[a(x), b(x)]$ is an RLWE encryption for m with secret key $v(x)$ and g is a gadget vector with base B. 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 $|e(x)|_{\psi_2} < \sigma$, $|e_i(x)|_{\psi_2} < \sigma_1$, then $|e_{new}(x)|_{\psi_2} \leq \sqrt{nl}B\sigma_1$.*

3 Fast Matrix–Vector Multiplication with Preprocessing

Rotation of ciphertext. For a vector $\mathbf{v} \in R_p$, when n_{RLWE} is a power of two and $p \equiv 1 \pmod{2n_{\text{RLWE}}}$, there exists an isomorphism between R_p and $\mathbb{Z}_p^n$ that maps $m(x)$ to $\mathbf{m}$, such that $m(x^5)$ corresponds to $\text{rot}^1(\mathbf{m})$. Therefore, for an encryption of a vector $\mathbf{v}$ (where $\mathbf{v}$ is first mapped to $v(x)$ via the inverse NTT before encryption) represented as $(a(x), b(x))$, we have $(a', b') = KSK_{s(x^5)\to s(x)}(a(x^5), b(x^5))$, which is an encryption of $\text{rot}^1(\mathbf{v})$.

BSGS Algorithm: Fast homomorphic matrix-vector multiplication (Algorithm 1). Given an RLWE ciphertext encrypting a vector $\mathbf{v} \in \mathbb{Z}_p^m$ and a plaintext matrix $M \in \mathbb{Z}_p^{n\times m}$ where $p \equiv 1 \pmod{2n_{\text{RLWE}}}$, the BSGS algorithm [18,17] can be used to homomorphically compute an RLWE ciphertext encrypting $M\mathbf{v}$. The core idea of this algorithm is based on the equation:

$$M\mathbf{v} = \sum_{i=0}^{n-1} \text{rot}^i(\text{diag}_i(M)) \cdot \text{rot}^i(\mathbf{v}).$$

The computational cost of this procedure arises from n ciphertext–plaintext multiplications and n ciphertext rotations. With the use of the Number Theoretic

Transform (NTT), the rotations can be interpreted as a permutation and some element-wise product between vectors. Furthermore, parts of the computation, such as the NTT and key-switching operations, can be preprocessed during the offline phase. However, despite this preprocessing, the algorithm still requires $O(nm)$ arithmetic operations over $\mathbb{Z}_q$.

To further improve efficiency, we propose Algorithm 2, a faster homomorphic matrix–vector multiplication algorithm with preprocessing (F-BSGS) by examining the structure of the baby-step algorithm and LWE encryption. Specifically, given a matrix M, an LWE ciphertext encrypting $\mathbf{v}$, and an auxiliary RLWE ciphertext encrypting the LWE secret key, we generate an RLWE ciphertext encrypting $M\mathbf{v}$. The core idea is to compress M using the LWE ciphertext (A, b) that encrypts $\mathbf{v}$. Observing that $M\mathbf{v}$ can be expressed as $M(b - As)$, we only need to apply the BSGS algorithm to the smaller matrix MA, which requires $O(nn_{\mathrm{LWE}})$ operations over $\mathbb{Z}_q$. In addition, we perform mn operations over $\mathbb{Z}_Q$ to compute Mb. When $n_{\mathrm{LWE}} \ll m$, the cost of the BSGS algorithm on the smaller matrix MA becomes negligible.

Algorithm 1 BSGS Algorithm for Matrix-Vector Multiplication

Require: An $n \times m$ plaintext matrix $\mathbf{M}$ with $n < m$; an RLWE ciphertext ct encrypting vector $\boldsymbol{v}$.
Ensure: An RLWE ciphertext ct' encrypting the product $\mathbf{M}\boldsymbol{v}$.
1: Initialize $ct' \leftarrow 0$.
2: **for** $i := 0$ to $n - 1$ **do**
3: $\quad ct_i \leftarrow \mathrm{rot}^i(ct)$.
4: $\quad ct' \leftarrow ct' + \mathrm{rot}^i(\mathrm{diag}_i(\mathbf{M})) \cdot ct_i$.
5: **end for**
6: **return** ct'.

Theorem 2. *Let $M \in \mathbb{Z}_p^{n\times m}$ and $v \in \mathbb{Z}_p^m$. Suppose all RLWE ciphertext errors are sampled from $\chi_{\sigma_{RLWE}}^{n_{RLWE}}$, all LWE ciphertext errors are sampled from $\chi_{\sigma_{LWE}}$, the gadget in the key-switching key uses base B, and the LWE ciphertext modulus Q is NTT-friendly. We can, using the outputs of Algorithm 2, with probability at least $1 - \delta$, correctly decrypt $Mv \in \mathbb{Z}_p$ with the RLWE secret key, as long as the following conditions hold:*

$$\frac{p}{q}\left(\frac{Q}{p} + \frac{q}{Q} + 1\right)p^2 \;\le\; \frac{1}{8},$$

$$\frac{p}{q}(3mpQ) \le 1/8$$

$$\left(\frac{p}{Q}\right)\sqrt{m}\,p\,\sigma_{LWE} \;<\; \frac{1}{8}\frac{1}{\sqrt{\ln\!\left(1 + 2n\delta^{-1}\right)}},$$

$$\frac{p}{q}\left(nn_{RLWE}\sqrt{l}\,QB\,\sigma_{RLWE}\right) \;<\; \frac{1}{8}\frac{1}{\sqrt{\ln\left(1+2n_{RLWE}\delta^{-1}\right)}}.$$

Proof. The proof for Theorem 2 is provided in Appendix A.

Algorithm 2 Faster Homomorphic Matrix-Vector Multiplication Algorithm (F-BSGS)

Require: An $n \times m$ plaintext matrix $\mathbf{M}$; an LWE ciphertext (A, b) encrypting v; a key-switching key from $S(x^5)$ to $S(x)$, where $S(x)$ is the RLWE secret key; an RLWE ciphertext ct encrypting the LWE secret key s.
Ensure: An RLWE ciphertext ct' encrypting $\mathbf{M}v \in \mathbb{Z}_p^n$.

Offline Phase: F-BSGS.Setup**.**
1: Compute and store hint $\leftarrow \mathbf{M}A$.

Online Phase: F-BSGS.Evaluation**.**
2: Compute $ct' \leftarrow \mathbf{M}b - \mathrm{BSGS}(\mathrm{hint}, ct)$.
3: **return** ct'.

4 YsPIR

In this section, we present the YsPIR protocol. We first introduce some algorithms used in our protocol.
QueryPack. Given a index $(u_d, u_{d-1}, \cdots, u_1) \in [N_d] \times [N_{d-1}] \times \cdots \times [N_1]$, this algorithm outputs an RLWE ciphertext encrypting $(s_d, s_{d-1}, \ldots, s_1, 0, \ldots, 0)$, where each $s_i = v_{u_i} \otimes [1, B, \ldots, B^{l-1}]$ has length $N_i l$.
QueryUnPack. Given an RLWE ciphertext output by QueryPack, KSKs, and a GSW encryption of the RLWE secret key, this algorithm outputs

$$\{\mathrm{GSW}_i[j] \mid i = 1, 2, .., d;\ j = 1, 2, ..., N_i\}$$

satisfying

$$\mathrm{GSW}_i[j] = \begin{cases} \mathrm{GSW}(1), & j = u_i, \\ \mathrm{GSW}(0), & j \neq u_i. \end{cases}$$

4.1 Motivation and Construction

We construct our PIR scheme by replacing the ciphertext–plaintext multiplication in OnionPIR with our F-BSGS algorithm. In our construction, we set the record size to be Lp bits. We represent the database as a $(d+1)$-dimensional hypercube, where the i-th dimension has length N_i. Each record in the database is uniquely identified by $d+1$ indices $(i_d, i_{d-1}, \ldots, i_0)$, denoted as $DB_{i_d i_{d-1} \cdots i_0}$. We group N_0 records together so that

$$DB_{i_d i_{d-1} \cdots i_1} = [DB_{i_d i_{d-1} \cdots i_1 0}, \ldots, DB_{i_d i_{d-1} \cdots i_1 (N_0-1)}],$$

which forms a matrix of size $L \times N_0$. The client aims to retrieve the record at index $(u_d, u_{d-1}, \ldots, u_1, u_0)$. This can be computed as:

$$\sum_{j_d=0}^{N_d-1} \mathrm{GSW}_d[j_d] \odot \cdots \odot \sum_{j_1=0}^{N_1-1} \mathrm{GSW}_1[j_1] \odot \mathrm{RLWE}[DB_{i_d, i_{d-1}, \ldots, i_1} \cdot v_{u_0}].$$

We compute $\mathrm{RLWE}[DB_{i_d, i_{d-1}, \ldots, i_1} \cdot v_{u_0}]$ using our F-BSGS algorithm. The subsequent external product steps are performed as in the original OnionPIR.

A detailed description of the protocol is given in Algorithm 3.

4.2 Correctness of YsPIR

Theorem 3. *YsPIR is a correct scheme with probability at least $1 - \delta$ if the following conditions hold:*

$$Q > 2p^2 \sqrt{N_0} \sigma \sqrt{\ln\left(1 + \frac{3L}{\delta}\right)},$$

$$q > 2 \left\| e_d \right\|_{\psi_2} \sqrt{\ln\left(1 + \frac{3L}{\delta}\right)} max_{i=1}^{k} Q_i,$$

$$\prod_{i=1}^{k} Q_i > \sqrt{n_{LWE}} Q \sigma \sqrt{\ln\left(1 + \frac{3L}{\delta}\right)}.$$

Proof. The proof of this theorem can be found in Appendix B.

4.3 Security of YsPIR

Theorem 4 (Security of YsPIR). *Assume that $LWE_{n_{LWE}, Q, L, \sigma}$ is hard, and $RLWE_{n_{RLWE}, q, (2+\log n_{RLWE})l, \sigma}$ is also hard. Furthermore, assume that the RLWE problem is circularly secure. Then the YsPIR scheme is a secure private information retrieval (PIR) protocol with preprocessing for a database of size N, in which each record consists of Lp bits, and p denotes the plaintext modulus used in the LWE encryption.*

Proof. The proof can be found in Appendix C.

4.4 Optimizations

Reducing the External Product by Half. In the answering phase, the server needs to perform external products a total of

$$\frac{N}{N_0} + \frac{N}{N_0 N_1} + \cdots + \frac{N}{N_0 N_1 \cdots N_{d-1}}$$

Algorithm 3 Our Protocol

Notation:

- p, Q, n_{LWE}, s: LWE parameters (plaintext modulus, ciphertext modulus, dimension, and secret key).
- Q, q, n_{RLWE}, S: RLWE parameters (plaintext modulus, ciphertext modulus, dimension, and secret key).
- DB: A $(d+1)$-dimensional hypercube, where the i-th dimension has length N_i.
- L: The record size, in p-bit units.
- $DB_{i_d i_{d-1} \ldots i_1}$: An $L \times N_0$ matrix with p-bit elements.
- $(u_d, u_{d-1}, \ldots, u_1, u_0)$: The index of the desired record.

Setup:

Client and server use a seed to generate all random parts of the LWE and RLWE encryptions.
The client computes GSW(S), RLWE(s), and KSK, and sends them to the server.
The server computes $(\text{hint}_s)_{i_d i_{d-1} \ldots i_1} \leftarrow \text{F-BSGS.Setup}(DB_{i_d i_{d-1} \ldots i_1}, A)$.

Query:

The client computes an LWE ciphertext (A, b) encrypting e_{u_0}.
The client conducts QueryPack$(u_d, \ldots, u_1)$, yielding query.
The client sends b and query to the server.

Answer:

The server conducts F-BSGS.Evaluation(RLWE(s), $(\text{hint}_s)_{i_d i_{d-1} \ldots i_1}$), yielding $\text{ct}_{i_d i_{d-1} \ldots i_1}$.
The server conducts QueryUnPack(query, GSW(S), KSK), yielding $\text{GSW}_j = [\text{GSW}_j[0], \ldots, \text{GSW}_j[N_j - 1]]$.
The server computes:
Result $\leftarrow 0$
for $i_d = 0$ **to** $N_d - 1$ **do**
    for $i_{d-1} = 0$ **to** $N_{d-1} - 1$ **do**
    $\vdots$
        for $i_2 = 0$ **to** $N_2 - 1$ **do**
            for $i_1 = 0$ **to** $N_1 - 1$ **do**
                Result += $\text{GSW}_d[i_d] \cdots \text{GSW}_2[i_2]\text{GSW}_1[i_1] \cdot \text{ct}_{i_d i_{d-1} \ldots i_1}$.
            end for
        end for
    end for
end for

Recover:

The client performs two rounds of decryption.

times. When we set $N_1 = N_2 = \cdots = N_d = 2$, this total cost approximates to $\frac{2N}{N_0}$. In this case, the d-level nested summation can be viewed as a series of d binary selections (i.e., "choose 1 out of 2" at each level). In the standard method, for each level j, both components of the GSW ciphertext $\mathrm{GSW}_j = (\mathrm{GSW}_j[0], \mathrm{GSW}_j[1])$ are separately subjected to an external product with RLWE ciphertexts of size $N/(2N_0N_1...N_{j-1})$. The results of these two products are then summed, resulting in $N/(N_0N_1 \cdots N_{j-1})$ external product operations in the j-th layer. However, if we reformulate the computation as:

$$\mathrm{GSW}(u_j) \cdot (\mathrm{RLWE}_1 - \mathrm{RLWE}_0) + \mathrm{RLWE}_0,$$

we only need to perform one external product for each $(\mathrm{RLWE}_0, \mathrm{RLWE}_1)$, reducing the number of external product operations to $N/(2N_0N_1...N_{j-1})$ in the j-th layer. This optimization effectively halves the number of external product computations required at each level. We only need to perform $\frac{N}{N_0}$ external products.

Use Seed to Reduce the Ciphertext Size. Since the random components of LWE and RLWE encryptions are independent of the encrypted message, they can be deterministically generated from seeds. Using this approach, the size of the query can be reduced by half.

Use Modulus Switching to Reduce the Answer Size. To reduce communication overhead, we apply modulus switching to the server's answer ciphertexts. Specifically, after performing the homomorphic evaluation under a large modulus q_1, the ciphertexts are transformed from $R_{q_1}^2$ to $R_{q_2} \times R_{q_3}$, where $q_2, q_3 < q_1$. This transformation can significantly reduce the size of the server's response.

In the actual implementation, we adopt all these improvements which make our scheme more efficient.

5 Implementation and Evaluation

5.1 Parameter Selection

We choose the scheme parameters to tolerate a correctness error of at most 2^{-80} with the database size up to 16 TB. We also choose the lattice parameters to ensure that each of the underlying RLWE/LWE assumptions which we require for security provides 128 bits of classical security.

Lattice parameters. In this work, we consider two lattice-based encryptions.

- **LWE.** We set the LWE dimension $n_{\mathrm{LWE}} = 1400$. The ciphertext modulus is $Q = 2^{32}$, and the plaintext modulus is $p = 2^8$. The standard deviation σ of the error distribution is 3.2.
- **RLWE.** We set the RLWE dimension $n_{\mathrm{RLWE}} = 2^{12}$. The ciphertext modulus is $q = q_1q_2$, where $q_i \equiv 1 \mod 2n_{\mathrm{RLWE}}$ and $q_i \approx 2^{48}$. The plaintext moduli are $Q_1 \approx Q_2 \approx 2^{22}$, where $Q_i \equiv 1 \mod (2n_{\mathrm{RLWE}})$. The standard deviation σ of the error distribution is 3.2.

Database configuration. We represent the database as $N_1 N_2 \cdots N_d$ matrix of size $L \times N_0$ where $N_1 = N_2 = \cdots = N_d = 2, N_0 = 2^{16}$ and $L = 2^{10}$.
Gadget parameters. In this work, we select two gadget parameters for KSKs and GSW encryption, choosing the same gadget base $B = 2^{16}$ for both. The decomposition length $l = 6$.
Moduli parameters. In the answering phase, we use modulus switching to reduce the response size. We switch $(a, b) \in R^2_{q,n_{\mathrm{RLWE}}}$ to $(a', b') \in R_{q_1,n_{\mathrm{RLWE}}} \times R_{q_2,n_{\mathrm{RLWE}}}$ where we set $q_1 \approx 2^{27}$ and $q_2 \approx 2^{33}$.

5.2 Experimental Evaluation

Database	Metric	$O(\log N)$ communication				$O(\sqrt{N})$ communication			
		YsPIR	KsPIR	Spiral	OnionPIR	SimplePIR	Hintless	YPIR	YPIR+SP
4 GB (1 B)	Offline communication	2.34 MB	7.00 MB	17.28 MB	5.00 MB	250.38 MB	–	–	–
	Query Size	400.00 KB	140.00 KB	14.00 KB	64.00 KB	227.00 KB	2465.79 KB	1259.00 KB	4046.00 KB
	Answer Size	60.00 KB	26.00 KB	21.00 KB	128.00 KB	250.00 KB	1474.56 KB	12.00 KB	36.00 KB
	Total Size	460.00 KB	166.00 KB	35.00 KB	192.00 KB	477.00 KB	3940.35 KB	1271.00 KB	4082.00 KB
	Answer Time	1.87 s	2.50 s	9.08 s	22.29 s	0.43 s	0.74 s	0.43 s	0.83 s
	Throughput	2.14 GB/s	1.60 GB/s	0.44 GB/s	0.18 GB/s	9.32 GB/s	5.41 GB/s	9.32 GB/s	4.30 GB/s
8 GB (512 B)	Offline communication	2.34 MB	7.11 MB	17.28 MB	5.00 MB	343.31 MB	–	–	–
	Query Size	400.00 KB	140.00 KB	14.00 KB	64.00 KB	341.00 KB	2465.79 KB	–	7631.00 KB
	Answer Size	60.00 KB	26.00 KB	21.00 KB	128.00 KB	343.00 KB	2949.12 KB	–	36.00 KB
	Total Size	460.00 KB	166.00 KB	35.00 KB	192.00 KB	684.00 KB	5414.91 KB	–	7667.00 KB
	Answer Time	3.42 s	5.01 s	17.40 s	42.28 s	0.74 s	1.47 s	–	1.58 s
	Throughput	2.34 GB/s	1.60 GB/s	0.46 GB/s	0.19 GB/s	10.80 GB/s	5.44 GB/s	–	5.06 GB/s
16 GB (1 KB)	Offline communication	2.34 MB	7.22 MB	17.28 MB	5.00 MB	486.00 MB	–	–	–
	Query Size	400.00 KB	140.00 KB	14.00 KB	64.00 KB	481.00 KB	4562.94 KB	–	14797.00 KB
	Answer Size	60.00 KB	26.00 KB	21.00 KB	128.00 KB	486.00 KB	2949.12 KB	–	36.00 KB
	Total Size	460.00 KB	166.00 KB	35.00 KB	192.00 KB	967.00 KB	7512.06 KB	–	14833.00 KB
	Answer Time	6.12 s	10.06 s	42.91 s	83.02 s	1.51 s	2.54 s	–	3.22 s
	Throughput	2.61 GB/s	1.59 GB/s	0.37 GB/s	0.19 GB/s	10.60 GB/s	6.29 GB/s	–	4.97 GB/s

Table 1: **Comparison of PIR Schemes.** In our experiment for HintlessPIR, we set the database as a matrix with entry size 1 B and dimensions $2^{13} \times 2^{19}$, $2^{14} \times 2^{19}$, and $2^{14} \times 2^{20}$ when the database sizes are 4, 8, and 16 GB, respectively; we regard each column of the database as a record. YPIR is designed only for small records of less than 1 B. Here, we present the results of YPIR+SP for large records. The results of KsPIR shown here correspond to the third construction in the paper.

We conducted all experiments on a machine equipped with an Intel Xeon W5-3425 processor running at 3.50 GHz, 256 GB of RAM, and running Ubuntu 22.04.5 LTS. We used Clang 16 as our C++ compiler. We use a single-threaded execution environment for all measurements. All measured running times were averaged over at least 5 trials and have a standard deviation of at most 5% of the average value.
Comparison schemes. We compare our scheme with OnionPIR, Spiral and KsPIR which have communication cost $O(\log N)$ and SimplePIR, HintlessPIR and YPIR which have communication cost $O(\sqrt{N})$. We set the record size from 1 B to 1 KB and the database size from 4 GB to 16 GB.

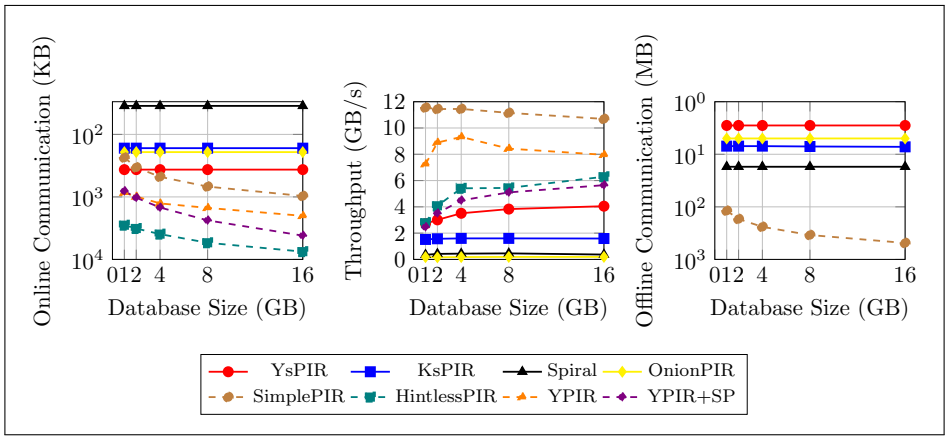


Fig. 1: Comparison of PIR schemes

Server throughput. In Table 1, we report the different computation and communication costs for retrieving a record from databases of varying sizes. The throughput of YsPIR is 75.38% slower than SimplePIR, 58.51% slower than HintlessPIR and 47.48% slower than YPIR+SP when the database is 16 GB. This is because we need to perform external products and execute the unpacking algorithm. The throughput of YsPIR is 1.64× faster than KsPIR, 7.05× faster than Spiral when the database is 16 GB. This is because we spend less time in First-dimension folding. In Table 1, the throughput of YsPIR increases with the database size. This is because the computation cost of the unpacking algorithm is $O(d)$. As the database size increases, the costs of First-dimension folding and the external product will dominate.

Communication. Compared with HE-based PIR schemes that have $O(\sqrt{N})$ communication complexity, the online communication of YsPIR is significantly smaller. Specifically, when the database size is 16 GB, YsPIR's total online communication is 2.10× smaller than SimplePIR, 16.33× smaller than HintlessPIR, and 32.25× smaller than YPIR+SP. Moreover, this advantage increases as the database size increases, due to the logarithmic dependence of YsPIR's communication cost on the database size. However, compared to PIR schemes with $O(\log N)$ communication complexity, YsPIR does not achieve the smallest communication overhead. The total online communication size of YsPIR is approximately 2.77× that of KsPIR, 13.14× that of Spiral, and 2.40× that of OnionPIR. This is primarily due to three factors: (1) we choose larger parameters to ensure the correctness of decryption; (2) the use of CRT in the recovery phase requires more RLWE ciphertexts to reconstruct LWE ciphertexts; and (3) an extra LWE ciphertext is included in the query phase. These factors collectively result in a slightly increased total communication size compared to other $O(\log N)$ PIR schemes.

Nonetheless, in terms of offline communication, YsPIR maintains a significant advantage. Its offline communication cost is only 2.34 MB, which is lower than other $O(\log N)$ PIR schemes such as KsPIR (7.22 MB), Spiral (17.28 MB), and OnionPIR (5.00 MB). Moreover, our offline communication cost is independent of the database size, which means that for dynamic databases, multiple offline communications are not required.

6 Conclusion

We have presented **YsPIR**, a novel HE-based PIR framework. Our scheme introduces a tunable trade-off between communication cost and server-side computation (answer time), enabling flexible deployment depending on application requirements.

References

1. Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies*, avril 2016:155–174, April 2016.
2. Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–Computation trade-offs in PIR. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1811–1828. USENIX Association, August 2021.
3. Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 962–979, 2018.
4. Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *2012 IEEE Symposium on Security and Privacy*, pages 257–271, 2012.
5. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1292–1303, New York, NY, USA, 2016. Association for Computing Machinery.
6. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 402–414, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
7. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50, 1995.
8. Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, November 1998.
9. Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 44–75, Cham, 2020. Springer International Publishing.

10. Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 655–684, Cham, 2019. Springer International Publishing.
11. Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, pages 803–815, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
12. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
13. Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with popcorn. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 91–107, Santa Clara, CA, March 2016. USENIX Association.
14. Ryan Henry. Polynomial batch codes for efficient it-pir. *Proceedings on Privacy Enhancing Technologies*, 2016:202 – 218, 2016.
15. Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast Single-Server private information retrieval. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3889–3905, Anaheim, CA, August 2023. USENIX Association.
16. Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with checklist. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 875–892. USENIX Association, August 2021.
17. Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz-Wu. Hintless single-server private information retrieval. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 183–217, Cham, 2024. Springer Nature Switzerland.
18. Ming Luo, Feng-Hao Liu, and Han Wang. Faster fhe-based single-server private information retrieval. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1405–1419, New York, NY, USA, 2024. Association for Computing Machinery.
19. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
20. Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server pir via fhe composition. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 930–947, 2022.
21. Samir Jordan Menon and David J. Wu. Ypir: high-throughput single-server pir with silent preprocessing. In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, USA, 2024. USENIX Association.
22. Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server pir. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2292–2306, New York, NY, USA. Association for Computing Machinery.
23. Rafail Ostrovsky and William E. Skeith. A survey of single-database private information retrieval: Techniques and applications. In Tatsuaki Okamoto and Xiaoyun

Wang, editors, *Public Key Cryptography – PKC 2007*, pages 393–411, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
24. Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1002–1019, New York, NY, USA, 2018. Association for Computing Machinery.
25. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.
26. Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.
27. Mingxun Zhou, Andrew Park, Wenting Zheng, and Elaine Shi. Piano: Extremely Simple, Single-Server PIR with Sublinear Server Computation . In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 4296–4314, Los Alamitos, CA, USA, May 2024. IEEE Computer Society.

A Proof of Theorem 2

Proof. Let the LWE ciphertext be (A, b), where

$$b = As + e_{\mathrm{LWE}} + \alpha Q + \frac{Q}{p}v.$$

Decrypting the ciphertext ct output by Algorithm 2 with the RLWE secret $S(X)$, we get

$$\begin{aligned}\mathrm{ct}[1] - \mathrm{ct}[0]S(X) =& \left\lceil \frac{q}{Q} \right\rfloor \left\lceil \frac{Q}{p} \right\rfloor Mv + \left\lceil \frac{q}{Q} \right\rfloor (Me_{\mathrm{LWE}} + MQ\alpha) + e_{\mathrm{BSGS}} \\ =& \frac{q}{p}Mv + e_1 + e_2 + e_3 + e_{\mathrm{BSGS}},\end{aligned}$$

where

$$e_1 = \left(\frac{Q}{p}\delta_{q/Q} + \frac{q}{Q}\delta_{Q/p} + \delta_{q/Q}\delta_{Q/p}\right) Mv, \quad e_2 = \frac{q}{Q}Me_{\mathrm{LWE}},$$

and

$$e_3 = \delta_{Q/p}(Me_{\mathrm{LWE}} + MQ\alpha).$$

The error terms satisfy

$$\|e_1\|_\infty \le \left(\frac{Q}{p} + \frac{q}{Q} + 1\right) p^2, \quad \|e_2\|_{\psi_2} \le \frac{q}{Q}\sqrt{m}\, p\, \sigma_{\mathrm{LWE}},$$

$$\|e_3\|_\infty \le 3mpQ, \quad \|e_{\mathrm{BSGS}}\|_{\psi_2} \le nn_{\mathrm{RLWE}}\sqrt{l}\, QB\, \sigma_{\mathrm{RLWE}}.$$

The theorem follows from Lemma 4.

Alternatively, Mv can be recovered by two rounds of decryption. First, using the RLWE secret key $S(X)$, we obtain $M(b - As)$. Then, rounding by Q/p recovers Mv. Thus, only the errors Me_{LWE} and e_{BSGS} need to be bounded, requiring

$$\frac{p}{Q}\sqrt{m}\, p\, \sigma_{\mathrm{LWE}} < \frac{1}{4\sqrt{\ln(1 + 2n\delta^{-1})}},$$

and

$$\frac{Q}{q}\left(nn_{\mathrm{RLWE}}\sqrt{l}\, QB\, \sigma_{\mathrm{RLWE}}\right) < \frac{1}{4\sqrt{\ln(1 + 2n_{\mathrm{RLWE}}\delta^{-1})}}.$$

Hence, a larger q is required.

B Proof of Theorem 3

F-BSGS with Non-NTT-Friendly Moduli. F-BSGS requires an NTT-friendly plaintext modulus Q. For non-NTT-friendly Q, we instead compute

$$DB \cdot (b - A \cdot s) \bmod Q_i$$

over NTT-friendly primes Q_i, and recover

$$DB \cdot (b - A \cdot s) \bmod \prod_{i=1}^{k} Q_i$$

via the Chinese Remainder Theorem. If

$$\|DB \cdot (b - A \cdot s)\|_\infty < \prod_{i=1}^{k} Q_i,$$

this also determines the value modulo Q. Since

$$\|DB \cdot A \cdot s\|_{\psi_2} \le \sqrt{n_{\mathrm{LWE}}}\, Q\, \sigma,$$

and $DB \cdot b$ can be reduced modulo Q after computation, it suffices to choose $\prod_i Q_i$ large enough to cover this bound.

Requirement of RLWE Parameters. Let e_{expand} be the expansion error and let e be the error in GSW(s), with $\|e\|_{\psi_2} < \sigma$. Each GSW ciphertext generated by OnionPIR.QueryPack has error

$$\|e_{\mathrm{query}}\|_{\psi_2} \le l^{1/2} n_{\mathrm{RLWE}}^{3/2} B\sigma + \sqrt{2l} B\sigma \approx l^{1/2} n_{\mathrm{RLWE}}^{3/2} B\sigma.$$

Let $\|e'\|_{\psi_2} < Ql^{1/2} B\sigma n_{\mathrm{RLWE}} L$ be the error in Enc($DBAs$), and define

$$K = \sqrt{2l} B \left(l^{1/2} n_{\mathrm{RLWE}}^{3/2} B\sigma\right).$$

Then the error after the first summation satisfies

$$\|e_1\|_{\psi_2} = \sqrt{N_1}\left(\|e'\|_{\psi_2} + K\right),$$

and for $j \geq 2$,

$$\|e_j\|_{\psi_2} = \sqrt{N_j}\left(\|e_{j-1}\|_{\psi_2} + K\right).$$

Thus, after the d-th summation,

$$\|e_d\|_{\psi_2} = \left(\prod_{j=1}^{d-1}\sqrt{N_j}\right)\|e_1\|_{\psi_2} + K\sum_{j=1}^{d-1}\left(\prod_{k=1}^{j}\sqrt{N_k}\right).$$

Requirement of LWE Parameters. Suppose the error in LWE is denoted by e''. Then the error in (DBb, DBA) becomes DBe''. From Lemma 2, we have

$$\|DBe''\|_{\psi_2} < p\sqrt{N_0}\|e''\|_{\psi_2}.$$

The inequality can be derived from Lemma 4.

C Proof of Theorem 4

Definition 10 (LWE Problem). *Let $\sigma > 0$, $n_{LWE}, Q, n_{sample} \in \mathbb{N}$, and let the secret $\mathbf{s} \in \mathbb{Z}_Q^{n_{LWE}}$ be sampled either uniformly or from $\chi_\sigma^{n_{LWE}}$. An LWE sample is*

$$(\mathbf{a}_i, b_i) = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \bmod Q), \quad \mathbf{a}_i \leftarrow \mathbb{Z}_Q^{n_{LWE}},\ e_i \leftarrow \chi_\sigma.$$

The decision-LWE problem $\mathrm{LWE}_{n_{LWE},Q,n_{sample},\sigma}$ *is to distinguish n_{sample} such samples from n_{sample} uniform samples $(\mathbf{a}_i, r_i) \in \mathbb{Z}_Q^{n_{LWE}} \times \mathbb{Z}_Q$.*

Definition 11 (RLWE Problem). *Let $\sigma > 0$, $n_{RLWE}, q, n_{sample} \in \mathbb{N}$, and let the secret $s \in R_q$ be sampled either uniformly or from χ_σ. An RLWE sample is*

$$(a_i, b_i) = (a_i, a_i \cdot s + e_i), \quad a_i \leftarrow R_q,\ e_i \leftarrow \chi_\sigma.$$

The decision-RLWE problem $\mathrm{RLWE}_{n_{RLWE},q,n_{sample},\sigma}$ *is to distinguish n_{sample} such samples from n_{sample} uniform samples $(a_i, r_i) \in R_q \times R_q$.*

Proof. When client queries an index $(u_d, \ldots u_1, u_0)$, the adversary observes:

- $\log n_{RLWE}$ key-switching keys under RLWE secret key S,
- An LWE encryption of u_0,
- An RLWE encryption of the LWE secret key s,
- An RLWE encryption of $u_d, u_{d-1}, \ldots, u_1$,
- A GSW encryption of the RLWE secret key S, which can be regarded as $2l$ RLWE encryptions.

Under the decisional LWE and RLWE assumptions, and assuming circular security, the RLWE encryption of the LWE secret key, RLWE encryption of $u_d, u_{d-1}, \ldots, u_1$, and GSW encryption of the RLWE secret key are indistinguishable from random. Thus, a query on index $(u_d, u_{d-1}, \ldots, u_1, u_0)$ is computationally indistinguishable from a query on index $(u'_d, u'_{d-1}, \ldots, u'_1, u'_0)$, ensuring the security of our PIR.