# NewtonPIR: Communication Efficient Single-Server PIR

Pengfei  $Lu^{1,2}(\boxtimes)$  and Hongyuan  $Qu^{1,2}$ 

<sup>1</sup> School of Cyber Science and Technology, Shandong University, Qingdao 266237, Shandong, China

PengfeiLu@mail.sdu.edu.cn

<sup>2</sup> Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao 266237, Shandong, China ghy1ghy@mail.sdu.edu.cn

Abstract. Private information retrieval (PIR) is a key component of many privacy-preserving systems. Although numerous PIR protocols have been proposed, designing a PIR scheme with communication overhead independent of the database size N and computational cost practical for real-world applications remains a challenge. In this paper, we propose the NewtonPIR protocol, a communication efficient single-server PIR scheme. NewtonPIR can directly generate query values for the entire index without splitting the index and sending multiple query ciphertexts. Specifically, NewtonPIR achieves communication overhead that is  $7.5 \times$  better than the state-of-the-art PIR protocol and  $35.9 \sim 75 \times$  better than the other protocols. In experiments, when the database size and entry size increase, the communication overhead of NewtonPIR remains stable. By utilizing the single-ciphertext fully homomorphic encryption (FHE) scheme and the simple Newton interpolation polynomial, along with precomputing coefficients in the offline phase, we reduce the computational overhead of NewtonPIR from hours in previous schemes to seconds. To the best of our knowledge, NewtonPIR is the first protocol to achieve communication cost independent of N along with computational overhead comparable to ring learning with errors (RLWE)-based PIR schemes. Additionally, we extend and introduce a private set intersection (PSI) protocol that balances computational and communication overhead more effectively.

**Keywords:** Private Information Retrieval · Homomorphic Encryption · Private Set Intersection.

# 1 Introduction

Protecting personal privacy has become a major focus for cloud-based applications. A recent global survey [27] shows that 85% of adults want more measures to protect their online identities, highlighting concerns about privacy leakage. Private information retrieval (PIR) protocols [13] enable clients to access server database entries without revealing the queried indices or keywords, as shown



Fig. 1: Simple process description of PIR protocol. The server begins by executing the setup phase ①. Next, the client generates and sends a query ②. Upon receiving the query, the server generates and returns a response ③. Finally, the client extracts the response to retrieve the requested database entry  $\mathbf{DB}[i]$  ④. Throughout this process, the server remains unaware of the query index i.

in Fig. 1. PIR is a fundamental component for various privacy-preserving applications, including anonymous communication [6], privacy-preserving media streaming [25], private contact discovery [30], privacy-friendly advertising [24], private blocklist lookups [32] and private navigation [43], among others.

PIR is primarily divided into two categories: single-server PIR [4,5,17,18,19] and multi-server PIR [13,14]. Multi-server schemes, referred to as information-theoretic PIR (or IT-PIR), are generally more efficient and offer guarantees of information-theoretic security. However, IT-PIR schemes require a stronger trust assumption, specifically that multiple servers do not collude. This need for coordination across multiple parties makes them challenging to implement in real-world scenarios. Single-server PIR, also known as computational PIR (or C-PIR), guarantees computational security. However, although powerful, C-PIR is prohibitively costly, and regrettably, this cost is intrinsic: C-PIR schemes require the server to process all elements in the database to respond to a single query [13]. Ultimately, if the server could exclude an element when responding to a query, it would infer that the excluded element holds no interest to the client. The central goal of this paper is to significantly enhance the efficiency of single-server PIR schemes and make them viable for practical adoption.

In single-server PIR, there are two main performance metrics: computation cost and communication overhead. There are often trade-offs between the two measures. Since Aguilar-Melchor *et al.* [36] proposed the first practical single-server PIR protocol, numerous PIR schemes [5,37,38,44,45] have been developed to further reduce computational and communication costs. These single-server PIR schemes can be broadly categorized into two groups: those based on the Learning with Errors (LWE) or Ring Learning with Errors (RLWE) problems [35], and those with simpler constructions.

The advantage of PIR schemes based on LWE or RLWE [5,37,38] lies in their practical computational cost, making them suitable for real-world applications. To achieve sublinear communication overhead, these schemes first treat the database as a multi-dimensional cube, then split the query index according to the dimensionality, and represent the split indices with different unit vectors. Finally, the client encrypts the query vectors and sends multiple ciphertexts to the server. However, it is worth noting that the communication overhead of these PIR protocols still depends on the database size N. As N increases, the communication efficiency decreases. Therefore, designing a PIR protocol whose communication overhead is independent of the database size N remains a challenge to be addressed.

The second category of simple PIR schemes [44,45] generates query values directly from the index, aiming to design communication overhead that is independent of N. These schemes use interpolation polynomials to compute the response values. However, we observe that the complex construction of polynomial bases in these schemes leads to response generation times reaching the scale of hours, significantly diminishing their practicality. Therefore, the natural question garners greater attention.

Can we design a communication efficient single-server PIR scheme with the communication overhead independent of N and the corresponding computational cost comparable to RLWE-based schemes besides?

#### 1.1 Our Contributions

We answer the question above in a positive affirmation by constructing the scheme using a combination of single-ciphertext FHE encryption and simple polynomial interpolation techniques. Below, we conclude our main contributions with high-level technique overviews.

- We propose the NewtonPIR scheme, a communication efficient single-server PIR protocol. Our NewtonPIR scheme leverages single-ciphertext FHE encryption to directly generate query values based on the index, making the communication overhead independent of the database size N. We conduct experiments with different database and item sizes, comparing NewtonPIR with the latest PIR schemes. Specifically, NewtonPIR's communication overhead is  $7.5 \times$  better than the state-of-the-art PIR scheme [37] and  $35.9 \sim 75 \times$ better than other schemes [3,4,5,38]. In the experiments, as the database size and item size increase, the communication overhead of our NewtonPIR scheme remains stable.
- We divide the NewtonPIR scheme into offline and online phases, using the Newton interpolation polynomial with the simple basis construction. In the offline phase, NewtonPIR preprocesses the database and precomputes the interpolation coefficients. This optimization shifts the majority of computational overhead from the online phase to the offline process, significantly enhancing the practicality of the scheme. Experiments show that, compared to the hour-level time consumption of previous schemes, NewtonPIR reduces the computational cost of generating responses by several orders of magnitude. To the best of our knowledge, NewtonPIR is the first protocol that achieves communication cost independent of N while bringing computational overhead close to that of RLWE-based schemes.

- 4 Pengfei  $Lu^{(\boxtimes)}$  and Hongyuan Qu
  - We also combine the single-ciphertext FHE scheme with private set intersection (PSI) to extend and construct a PSI protocol. The grouping optimization proposed in [11] reduces computational complexity but increases communication overhead. To address this issue, we use noiseless single-ciphertext FHE encryption to achieve a balance between computational and communication costs.

### 1.2 Organization

The remainder of the paper is as follows: In section 2, we introduce related work. Section 3 describes the definitions of the single-ciphertext FHE scheme, PIR, and PSI protocols. Sections 4 and 5 present our NewtonPIR and PSI schemes respectively. Theoretical analysis and experimental evaluation are provided in section 6. Conclusions are in the section 7.

# 2 Related Work

### 2.1 Early Single-Server PIR Schemes

Numerous initial designs for single-server PIR systems [9,23] are based on the Kushilevitz-Ostrovsky framework [33], which utilizes homomorphic encryption. However, Sion and Carbunar [40] point out that when network bandwidth is limited to a few hundred Kbps, these methods often perform worse than downloading the entire database. The server's inefficiency is largely due to the requirement of performing at least N large integer modular multiplications or exponentiations. The computational overhead of these operations can often outweigh the simplicity of transmitting the data directly to the client. Recent advancements in lattice cryptography have introduced post-quantum and parallel-friendly PIR techniques, available in multiple variants.

### 2.2 Recent Single-Server PIR Schemes

Current practical single-server PIR schemes utilize lattice-based cryptographic techniques, with a particular emphasis on somewhat homomorphic encryption (SHE) algorithms. Aguilar-Melchor *et al.* [36] introduce the first practical method, known as XPIR. While XPIR significantly reduces the computational cost compared to earlier systems, its communication overhead remains prohibitively high. Subsequent PIR approaches [4,5] have been developed to address these limitations. SealPIR [5], for instance, employs query compression techniques to mitigate the issue of large request sizes, successfully reducing the request size to 64 KB. However, this improvement comes with a slight increase in computational complexity. Despite this advancement, the response size in SealPIR remains similar to that of XPIR, resulting in a total communication overhead approximately 2,360× the size of the desired retrieval item.

MulPIR [4] improves upon SealPIR by optimizing the response size, reducing the communication overhead to approximately  $932 \times$  that of a similar scenario. The key innovation in MulPIR is the direct multiplication of ciphertexts in the second and higher dimensions, rather than employing the chunking method used by SealPIR. However, this technique leads to increased noise expansion, necessitating the use of less efficient RLWE parameters, which in turn raises computational cost. The OnionPIR scheme [38] offers a holistic approach by employing private batch sum retrieval, effectively reducing the computational load and thereby lowering the online cost associated with performing PIR. Meanwhile, Spiral [37] introduces modulus switching and matrix RLWE ciphertext, which dramatically reduces communication overhead by approximately  $126 \times$  in similar scenarios.

SimplePIR [26] and FrodoPIR [19] are distinct PIR solutions that share a common insight: in LWE-based ciphertext, the majority of the server's computations are independent of the client's query and can therefore be performed during the offline phase. However, this advantage comes with the trade-off that the client must download a significant amount of data during the offline phase. In PIR schemes with preprocessing [8] or doubly-efficient PIR [10], the server can preprocess the database, enabling it to respond to online queries in sub-linear time. Boyle *et al.* [8] and Canetti *et al.* [10] illustrate the construction of doubly-efficient PIR schemes using virtual black box obfuscation, although these constructions rely on very strong cryptographic assumptions.

The PIR schemes discussed above are all based on the LWE or RLWE problems. Besides these, there are also simpler constructions of PIR. Yi *et al.* [45] combine the DGHV encryption [42] over integers with PIR, where the communication complexity is related to the ciphertext length and the database size. In another scheme, Xu *et al.* [44] use the Lagrange interpolation polynomial to construct PIR. However, the computation and construction of the interpolation polynomial are complex, significantly impacting the practicality of the scheme. These types of PIR schemes have simpler constructions, but there are many aspects of computation and communication overhead that require further optimization.

### **3** Preliminaries

**Notations.** In this paper, we denote scalars in plain (e.g. x) and vectors in bold (e.g. **DB**). For an N-element database **DB**, let **DB** $[i]_{i \in [N]}$  denote its *i*-th element. We use [N] to represent the set  $\{1, \dots, N\}$ . All logarithms in this paper have base 2.

The ring of polynomials over the integers, which consists of symbolic polynomials with integer coefficients, is denoted by  $\mathbb{Z}[x]$ . Given a polynomial f(x), the ring  $\mathbb{Z}[x]/\langle f(x) \rangle$  is the ring of all polynomials modulo f(x). The ring of polynomials with coefficients in  $\mathbb{Z}_n$  is denoted by  $\mathbb{Z}_n[x]$  and  $\mathbb{Z}_n[x]/\langle f(x) \rangle$  is defined analogously. And let  $\mathbb{Z}_n[x, y]/\langle f(x), g(y) \rangle$  denote bivariate truncated polynomial ring [44].

#### 3.1 Single-Ciphertext FHE Scheme

FHE is an encryption scheme that enables arbitrary computations on encrypted data [21,22]. In practical applications, FHE is often implemented in a leveled manner, meaning that operations can only be performed a finite number of times. Exceeding this limit results in failure of ciphertext decryption.

Single-ciphertext FHE differs from general FHE in two key aspects. First, single-ciphertext FHE performs homomorphic computations on a single ciphertext rather than on multiple ciphertexts. Second, homomorphic computations on a single ciphertext generate no noises. Therefore, the proposed single-ciphertext FHE is a specialized form of FHE that supports unlimited univariate computations on a single ciphertext without requiring access to the secret key.

A single-ciphertext FHE scheme consists of four probabilistic polynomial time (PPT) algorithms: key generation, encryption, decryption, and homomorphic evaluation. A specific scheme [44] is described as follows:

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{KeyGen}(\lambda)$ : The algorithm takes the security parameter  $\lambda$  as an input and randomly generates two  $\lambda/2$ -bit large primes p and q such that gcd(p-1,3) = 1 and gcd(q-1,3) = 1. It then computes n = pq,  $\varphi(n) = (p-1)(q-1)$  and the inverse d of 3 modulo  $\varphi(n)$ , namely,  $3d = 1 \pmod{\varphi(n)}$ . The modulus n is defined as the public key pk = n, the evaluation key is set as evk = n, and the integer d is assigned as the secret key, i.e., sk = d.
- $\mathbf{c} \leftarrow \mathsf{Enc}(m,\mathsf{pk})$ : The algorithm takes a plaintext  $m \in \mathbb{Z}_n$  as input and randomly selects  $a, b \in \mathbb{Z}_n$ , then computes  $u = a^3 \pmod{n}$  and  $v = b^3 \pmod{n}$ . It also chooses 9 random integers  $a_{ij} \in \mathbb{Z}_n$  for  $i, j \in \{0, 1, 2\}$ to construct the polynomial  $f(x, y) = \sum_{i=0}^2 \sum_{j=0}^2 a_{ij} x^i y^j$ . The polynomial F(x, y) is defined as  $F(x, y) = f(x, y) - f(a, b) \pmod{n}$ , and the ciphertext is calculated as  $c(x, y) = F(x, y) + m \pmod{n}$ . The resulting ciphertext is  $\mathbf{c} = \mathsf{Enc}(m, n) = (u, v, c(x, y))$ .
- $-m \leftarrow \mathsf{Dec}(\mathbf{c}, \mathsf{sk})$ : The algorithm takes a ciphertext  $\mathbf{c}$  and the private key sk = d as input, computes  $a = u^d \pmod{n}$  and  $b = v^d \pmod{n}$  and substitutes a, b into c(x, y) to recover the plaintext m.
- Eval $(h, \mathbf{c}, \mathsf{evk})$ : The algorithm takes the ciphertext  $\mathbf{c}$  and a univariate polynomial  $H(x) = \sum_{i=0}^{l} h_i x^i$  as input, and outputs the evaluation result. The specific evaluation process is detailed in Algorithm 1, where the addition and multiplication operations are performed over the truncated polynomial ring  $\mathbb{Z}_n[x, y] / \langle x^3 u, y^3 v \rangle$ .

**Correctness.** The correctness of decryption requires that the plaintext m can be correctly decrypted from the ciphertext, i.e., Dec(c(x, y), d) = c(a, b) = $F(a, b) + m = m \pmod{n}$ . Similarly, for homomorphic evaluation to be correct, the ciphertext  $c_H(x, y)$  can be correctly decrypted to plaintext H(m), i.e.,  $\text{Dec}(c_H(x, y), d) = c_H(a, b) = H(c(a, b)) = H(m) \pmod{n}$ .

**Security**. Below, we provide the definition of the 3rd RSA problem. The oneway security of the above single-ciphertext FHE scheme can be reduced to the hardness of this problem [44].

Algorithm 1: Evaluation algorithm

Input: The univariate polynomial  $H(x) = \sum_{i=0}^{l} h_i x^i$ , the ciphertext  $\mathbf{c} = (u, v, c(x, y))$  and the evaluation key  $\mathbf{evk} = n$ Output: The evaluation result  $\mathbf{c}_H$ 1  $c_H(x, y) = H(c(x, y)) = 0$ 2 for i = 0 : l do 3  $| c_H(x, y) = c_H(x, y)c(x, y) + h_{l-i} \pmod{n, x^3 - u, y^3 - v}$ 4 end 5 return  $\mathbf{c}_H = (u, v, c_H(x, y))$ 

**Definition 1 (3rd RSA problem)** The e-th RSA problem can be defined as follows: given the RSA public key n = pq and e, along with a ciphertext c', to determine the corresponding plaintext m' that satisfies the equation  $c' = m'^{e}$  (mod n). The 3rd RSA problem refers to this scenario when e is set to 3.

**Theorem 2 (see [44], Theorem 1).** The one-way security of the proposed single-ciphertext FHE scheme is polynomially equivalent to the 3rd RSA problem.

#### 3.2 Newton's interpolation

#### 3.3 Private Information Retrieval (PIR)

A PIR protocol deals with a scenario where a server holds a database  $\mathbf{DB} = \{d_1, d_2, \dots, d_N\}$  consisting of N elements, and a client possesses an input index *i*. The objective of the protocol is to facilitate the client in learning  $\mathbf{DB}[i]$  while ensuring that the server learns nothing about *i*.

We work in a model where an initial setup algorithm outputs an internal server state. A PIR scheme with server-independent preprocessing [7] consists of the following four algorithms:

- $dbp \leftarrow \mathsf{PIR.Setup}(1^{\lambda}, \mathbf{DB})$ . On input the security parameter  $\lambda$  and a database **DB**, the setup algorithm outputs a set of database parameters dbp;
- $-(st, \mathbf{Q}) \leftarrow \mathsf{PIR}.\mathsf{Query}(i)$ . On input an index *i*, the query algorithm outputs a secret state *st* and a corresponding query  $\mathbf{Q}$ ;
- $R \leftarrow \mathsf{PIR.Response}(\mathbf{dbp}, \mathbf{Q})$ . On input the database parameters  $\mathbf{dbp}$ , the query  $\mathbf{Q}$ , the answer algorithm outputs a response R;
- **DB** $[i] \leftarrow$  **PIR.Extract**(st, R). On input the client secret state st and the response R, the extract algorithm outputs a database record **DB**[i].

The algorithms should satisfy the following properties:

**Correctness.** We say that a PIR scheme has correctness error  $\delta$  if, on database size N, for all databases  $\mathbf{DB} = \{d_1, d_2, \dots, d_N\}$ , and for all indices  $i \in [N]$ , the probability defined below is at least  $1 - \delta$ :

$$Pr\left( \begin{aligned} \mathbf{DB}[i] = \mathbf{DB}^{'}[i] : & \underbrace{(st, \mathbf{Q}) \leftarrow \mathsf{PIR}.\mathsf{Setup}(1^{\lambda}, \mathbf{DB})}_{R \leftarrow \mathsf{PIR}.\mathsf{Query}(i)} \\ \mathbf{DB}^{'}[i] : & \underbrace{(st, \mathbf{Q}) \leftarrow \mathsf{PIR}.\mathsf{Response}(\mathbf{dbp}, \mathbf{Q})}_{\mathbf{DB}^{'}[i] \leftarrow \mathsf{PIR}.\mathsf{Extract}(st, R)} \end{aligned} \right). \tag{1}$$

For the PIR scheme to be non-trivial, the total client-to-server communication should be significantly smaller than the bit length of the database. In other words, it must satisfy the condition  $|\mathbf{Q}| + |R| \ll |DB|$ .

**Security.** The query generated by the client should not leak any information about the desired database record. That is, we say a PIR scheme is  $(T, \epsilon)$ -secure if, for all adversaries  $\mathcal{A}$  running in time at most T, with a database size of N, and for any  $i, j \in [n]$ ,

$$\begin{aligned} |Pr(\mathcal{A}(1^{N},\mathbf{Q}_{i}) = 1:(st,\mathbf{Q}_{i}) \leftarrow \mathsf{PIR}.\mathsf{Query}(i)) \\ - Pr(\mathcal{A}(1^{N},\mathbf{Q}_{i}) = 1:(st,\mathbf{Q}_{i}) \leftarrow \mathsf{PIR}.\mathsf{Query}(j))| \leq \epsilon. \end{aligned}$$
(2)

**Batch PIR**. An extension of the standard PIR protocol is known as Batch PIR or Multi-query PIR [29], where the client may want to retrieve a batch of entries from the database at once. Currently, the Batch PIR schemes [6,34,39] mainly use the batch framework from [5], which is composed of batch codes [29] and cuckoo hashing. The cuckoo hashing is described as follows:

**Cuckoo hashing**. Given  $n_b$  balls, b buckets, and  $\omega$  independent hash functions  $h_0, h_1, \dots, h_{\omega-1}$  that assign each ball to a random bucket, the algorithm determines  $\omega$  candidate buckets for each ball by applying the  $\omega$  hash functions. For each ball g, it is placed in any empty candidate bucket. If none of the  $\omega$  candidate buckets are empty, one is selected at random, and the ball currently in that bucket  $(g_{old})$  is removed. Then, g is placed in the bucket, and  $(g_{old})$  is re-inserted. If re-inserting  $(g_{old})$  causes another ball to be removed, this process continues recursively for a maximum number of iterations.

#### 3.4 Private Set Intersection (PSI)

Private set intersection (PSI) allows a sender and a receiver to compute the intersection of their private sets X and Y. The receiver learns only the intersection  $X \cap Y$ , while the sender learns nothing. Most PSI protocols are balanced, assuming that the set sizes are similar and that the communication cost depends on the larger set. In contrast, unbalanced PSI protocols [11,12,16] focus on scenarios where the receiver's set is much smaller, with the goal of achieving communication complexity dependent on the size of the receiver's set. Unbalanced PSI can be viewed as Batch PIR with stronger security guarantees: it protects both the receiver's queries and the sender's database.

Existing PSI protocols [11,12,16] operate by having the receiver encrypt the elements in Y using FHE and send the ciphertexts to the sender. For each

encrypted  $y_i$ , the sender performs a homomorphic evaluation of a polynomial that interpolates the entries and then returns the result to the receiver.

**Oblivious pseudorandom function**. However, simply executing this protocol would expose information about the sender's entries that do not belong to the intersection. To address this, an oblivious pseudorandom function (OPRF) [20] is employed to obscure the elements. In this approach, both the sender and the receiver independently apply the OPRF to the elements in their respective sets X and Y. The sender then uses the OPRF outputs to mask the corresponding entries. The receiver retrieves these masked entries and utilizes its own OPRF values to unmask the entries, thereby obtaining the intersection.

**Optimizations**. To effectively reduce the computational cost of polynomial interpolation, the above scheme implements two optimizations: first, encrypting additional powers of the receiver's set elements, and second, partitioning the sender's set into multiple subsets. The second optimization increases the return communication from the sender to the receiver due to the additional subsets, which can be further optimized.

### 4 Our Communication Efficient PIR Protocol

#### 4.1 Detailed Description

We use the single-ciphertext FHE scheme described in Section 2.1 and the Newton interpolation polynomials to construct our communication efficient singleserver PIR protocol. Below, we provide a full description of our NewtonPIR protocol, which involves four phases: setup, query, response and extract. We completely present it in the Fig. 2.

We describe our protocol by separating it into offline and online phases. During the offline phase, the server executes the setup algorithm, preprocessing the database and precomputing interpolation coefficients as database parameters. The PIR query and response generation, as well as the extraction process, are carried out during the online phase. The client uses single-ciphertext FHE encryption to generate the query value and the secret state, where the secret state is kept by the client, and the query value is sent to the server. Upon receiving the query, the server utilizes the precomputed database parameters and Newton interpolation polynomial to generate the response value and sends it back. Using the secret state, the client decrypts the response to obtain the desired database entry.

### Offline preprocessing.

- PIR.Setup $(1^{\lambda}, \mathbf{DB})$ : On input the security parameter  $\lambda$  and a database  $\mathbf{DB} = \{d_1, d_2, \cdots, d_N\}$ , the setup algorithm proceeds as follows:
  - 1. The server precomputes N divided differences according to the Algorithm 2.

Algorithm 2: Computing Newton interpolation coefficients

Input: A database  $\mathbf{DB} = \{d_1, d_2, \cdots, d_N\}$ Output: Newton interpolation coefficients **R** 1 for i = 1 : N do 2 |  $R_i = d_i$ 3 end 4 for j = 1 : N - 1 do 5 | for i = j : N - 1 do 6 |  $R_{i+1} = (R_{i+1} - R_i)/(j)$ 7 | end 8 end 9 return  $\mathbf{R} = \{R_1, R_2, \cdots, R_N\}$ 

2. Output the database parameters dbp = R.

#### Online processing.

- PIR.Query(i): On input an index  $i \in [N]$ , the query algorithm does the following:
  - 1. The client randomly generates two  $\lambda/2$ -bit prime numbers p and q, ensuring that gcd(p-1,3) = 1 and gcd(q-1,3) = 1, and then computes n = pq and the inverse d of 3 modulo  $\varphi(n)$ . Set st = d.
  - 2. Randomly choose  $a_{ij} \in \mathbb{Z}_n$  for i, j = 0, 1, 2 and construct  $f(x, y) = \sum_{i=0}^2 \sum_{j=0}^2 a_{ij} x^i y^j$ . Randomly choose  $a, b \in \mathbb{Z}_n$  and compute  $u = a^3 \pmod{n}, v = b^3 \pmod{n}$ .
  - 3. Set  $F(x, y) = f(x, y) f(a, b) \pmod{n}$ , compute  $c(x, y) = F(x, y) + i \pmod{n}$  and set  $\mathbf{Q} = (n, u, v, c(x, y))$ .

4. Output  $(st, \mathbf{Q})$ .

- PIR.Response(dbp, Q): On input the database parameters dbp =  $\{R_1, R_2, \dots, R_N\}$  and the query  $\mathbf{Q} = (n, u, v, c(x, y))$ , the server response algorithm proceeds as follows:

1. The server computes

$$N(x,y) = R_1 + \sum_{l=2}^{N} R_l \prod_{j=1}^{l-1} (c(x,y) - j) \pmod{n, x^3 - u, y^3 - v}.$$

2. Output the response R = N(x, y).

- PIR.Extract(st, R): On input the client secret state st = d and the response R = N(x, y), the extraction algorithm first computes N(a, b) with  $a = u^d \pmod{n}$ ,  $b = v^d \pmod{n}$  and outputs the result.

Fig. 2: The NewtonPIR protocol.

#### 4.2 Correctness and Security Analysis

**Correctness.** In the response generation phase, the response R = N(x, y) is computed as an evaluation of the encrypted index *i*. Upon correctly decrypting the response R, the client will obtain that

$$N(a,b) = R_1 + \sum_{l=2}^{N} R_l \prod_{j=1}^{l-1} (c(a,b) - j)$$
  
=  $R_1 + R_2(c(a,b) - 1) + \dots + R_N \prod_{j=1}^{N-1} (c(a,b) - j)$  (3)  
=  $R_1 + R_2(i-1) + \dots + R_N \prod_{j=1}^{N-1} (i-j) \pmod{n}.$ 

By the properties of Newton interpolation, we can obtain  $N(a, b) = d_i = \mathbf{DB}[i] \pmod{n}$  since  $c(a, b) = i \pmod{n}$ . Therefore, the probability expression 1 is satisfied:

$$Pr\left(\mathbf{DB}[i] = \mathbf{DB}'[i]\right) \ge 1 - \delta.$$
(4)

Consequently, the correctness of our NewtonPIR protocol holds as expected. Security. We know that the query index is encrypted using the single-ciphertext FHE scheme, and only the querying user can obtain the index. According to Theorem 2, the security of the single-ciphertext FHE scheme can be reduced to the 3rd RSA problem, meaning that without knowledge of the private key, no one can retrieve the query index. Therefore, in the single-server NewtonPIR protocol, the query value does not leak any information about the index, thereby ensuring privacy protection for the query index. Equation 2 is satisfied:

$$|Pr(\mathcal{A}(1^N, \mathbf{Q}_i) = 1) - Pr(\mathcal{A}(1^N, \mathbf{Q}_j) = 1)| \le \epsilon.$$
(5)

Consequently, the security of our NewtonPIR protocol holds as expected.

#### 4.3 Optimizations and Extensions

A variant of the single-ciphertext FHE scheme. In our NewtonPIR protocol, the client can encrypt the index using the single ciphertext FHE scheme and then send the ciphertext to the server. For efficiency reasons, we can instead directly use a symmetric encryption scheme to encrypt the index. The parameters a and b link the partial ciphertext c(x, y) with its corresponding plaintext. Specifically, the parameters a and b call a polynomial that is used to encrypt the query index. Meanwhile, the polynomial ciphertext c(x, y) can be decrypted using the parameters a and b directly, bypassing the auxiliary information uand v. The server then outputs a function of the *i*-th data **DB**[*i*] related to

the polynomial ciphertext. At this point, the user uses the parameters a and b to decrypt the function and accurately obtain the  $\mathbf{DB}[i]$  corresponding to the index i. Throughout this process, the server provides some computational and storage space but cannot access information about the index i. Therefore, our NewtonPIR protocol still meets the security requirements.

By using this optimization, the need to compute modular inverses during the query generation process is eliminated, and the time-consuming exponentiation operations during the extraction process are no longer required, greatly improving computational efficiency. We will apply this optimization in the experiments later.

**Preventing the semi-malicious server**. An important security enhancement for PIR is the need to prevent the semi-malicious server from forging and tampering with data [15,28,31]. To prevent a semi-malicious server from forging entries in the database as a response, we can add a verifiable process. Here is an ideal approach: we use a hash function h to process the data because it is one-way. We require the server to replace  $d_j$  with  $d_j || h(d_j)$  to compute the coefficients and send the correct result to the client in the response R. Undoubtedly, the length of  $d_j || h(d_j)$  is less than n. The client can verify whether the server has forged the entry. After decrypting the response R to obtain  $d_j$ , the client can compute the hash value  $h'(d_j)$ . If it matches the value  $h(d_j)$  sent by the server, then the entry  $d_j$  corresponds exactly to index i, with no errors. If not, the server is semi-malicious. We omit the details here.

Supporting databases with larger entries. We propose an extension to handle this situation efficiently with large database elements in practical scenarios. To handle entries larger than  $\lambda$  bits, the server divides each entry into  $\lambda$ -bit segments, effectively creating multiple sub-databases ( $\mathbf{DB}_1, \mathbf{DB}_2, \dots, \mathbf{DB}_k$ ), where  $\mathbf{DB}_j$  corresponds to the *j*-th segment of all entries. The server applies the same client query to all sub-databases. By performing multiple online queries and combining the returned results, the client can retrieve any record of its choice. Batching client queries (Batch PIP) In many applications [6, 34, 20], a client

**Batching client queries (Batch PIR)**. In many applications [6,34,39], a client wishes to retrieve k records from a PIR server. If the client runs the NewtonPIR protocol k times on a database of size N, the total time for the server would be approximately kN. We can apply the Batch PIR framework from [5], allowing the client to retrieve k records at a cost much lower than kN for the server. Specifically, the server uses batch codes [29] to add each entry to all candidate buckets, and the client uses cuckoo hashing to assign each index to one of the candidate buckets. Therefore, if a specific index i is assigned to bucket j on the client side, bucket j on the server side is guaranteed to contain the i-th entry of the database. The client and server perform a NewtonPIR instance for each bucket to retrieve all the desired entries.

## 5 Description of the PSI Protocol

We combine the single-ciphertext FHE encryption scheme with the framework from [11] for the first time to construct the PSI protocol. The brief description is as follows:

- **Perform hashing**. The receiver constructs a cuckoo hash table for its set Y. Specifically, the receiver uses three hash functions  $h_1, h_2, h_3$  and a vector  $\{M_R[0], \dots, M_R[m]\}$  of O(|Y|) bins, where m is the size of the cuckoo hash table. For each  $y \in Y$ , the receiver places y into bin  $M_R[h_i(y)]$  for some i, ensuring that no bin contains more than one item. For all  $x \in X$  and all  $i \in \{1, 2, 3\}$ , the sender places x into bin  $M_S[h_i(x)]$ . Note that when |X| is larger than m, each bin on the sender's side will likely contain O(|X|/m) items. The intersection  $X \cap Y$  is then maintained as the union of the intersections across all corresponding bins. In other words,

$$X \cap Y = \bigcup_{j} M_R[j] \cap M_S[j] = \bigcup_{j} \{y_j\} \cap M_S[j],$$

where  $y_j$  is the unique item in bin  $M_R[j]$ .

- **Encrypt** Y. The receiver encrypts y with the single-ciphertext FHE scheme and sends the ciphertext c to the sender.
- **Intersect**. The sender randomizes the encoding using a uniformly sampled element r and computes

$$\hat{z} = r \prod_{x \in M_S[j]} (c - x).$$

The sender then returns  $\hat{z}$  to the receiver.

- Decrypt and get result. When  $y \in M_S[j]$ , one term in the product becomes zero, so  $\hat{z}$  will be an encryption of zero. The receiver concludes that if z = 0, then  $y \in X$ . In the case where  $y \notin M_S[j]$ , the product will be a nonzero value. Therefore, z = 0 if and only if  $y \in X$ ; otherwise, z is uniformly distributed and independent of the set X.

To prevent the leakage of information about elements in the sender's set that are not in the intersection, we use the OPRF introduced in Section 3.3 to mask the elements. The following describes the optimizations applied in our protocol. **Windowing and partitioning**. For  $M \approx |X|/m$ , the multiplication depth for directly computing z is  $O(\log M)$ . Using the windowing technique, this can be reduced to  $O(\log(\log M))$ . It is observed that z can be expressed as a polynomial  $g(y) = g_M y^M + \cdots + g_1 y + g_0$ , where the coefficients  $g_i$  are determined by r and  $M_S[j]$ . The receiver sends encryptions of  $y, y^{2^1}, \cdots, y^{2^{\log M}}$ , and the sender can use these terms to compute all necessary powers of y with a multiplication depth of  $O(\log(\log M))$ .

Using the partitioning technique, the sender's bins are divided into t subsets. The sender can then process these subsets independently, further reducing the multiplication depth to  $O(\log(\log(M/t)))$ . As mentioned in Section 3.3, the drawback

of this technique is that for each y, the response ciphertexts  $\hat{z}_1, \hat{z}_2, \dots, \hat{z}_t$  must be sent back to the receiver, increasing the response size by a factor of t. To reduce the communication overhead, our protocol uses the single-ciphertext FHE encryption.

# 6 Performance Evaluation

The performance of our NewtonPIR is contingent upon the selection of RSA parameters. To identify the optimal choices, we conduct tests with varying parameters. We set the length of the module n to 2048 bits. In NewtonPIR, polynomial multiplications are executed using the number-theoretic transformation (NTT). To further optimize the computation, we utilize NFLlib [1], an efficient library known for its arithmetic optimizations and AVX2 specialization tailored for polynomial arithmetic operations.

#### 6.1 Theoretical Analysis

**Communication cost**. In the NewtonPIR protocol, we use the single-ciphertext FHE scheme to encrypt the index. On one hand, during the query generation process, the index value can be directly encrypted without first splitting it into multiple sub-indices according to the database dimensions, representing them with multiple unit vectors, and then encrypting each of those unit vectors. On the other hand, the generated query is sent to the server as a single ciphertext, eliminating the need to send multiple ciphertexts based on the database dimensions. These two aspects reduce both the dimensionality and the number of ciphertexts sent, effectively lowering communication cost.

Below, we compare the communication complexity of NewtonPIR with different PIR schemes. Among them, the Spiral scheme is currently the state-of-the-art PIR scheme in terms of communication cost. The underlying encryption systems of the compared schemes are based on the Learning with Errors (LWE) or Ring Learning with Errors (RLWE) problems [35]. Additionally, we included a PIR scheme based on the DGHV encryption scheme [42] as a comparison group. The specific comparison is shown in Table 1, where the second column indicates the underlying hard problem, and the third column indicates whether the scheme is batched.

As shown in Table 1, the communication complexity of the FastPIR scheme is linear with respect to the database size N, while other schemes based on the LWE or RLWE problems exhibit sublinear complexity with N. In contrast, the communication complexity of our NewtonPIR scheme is independent of N.

**Computational cost**. We analyze the computational complexity of our NewtonPIR scheme from both the client and server perspectives. On the client side, the encryption optimization introduced in Section 4.3 effectively reduces the number of modular multiplications required during query generation. Specifically, computing the polynomial F(x, y) involves 18 modular multiplications and some modular additions, as each monomial  $a_{ij}x^iy^j$  in F(x, y) requires i + j

	Hardness	Communication	n Batch
SealPIR [5]	RLWE	$O(\sqrt{N})$	X
MulPIR [4]	RLWE	$O(\sqrt{N})$	×
FastPIR [3]	RLWE	O(N)	×
OnionPIR [38]	RLWE	$O(\log N)$	1
Spiral [37]	RLWE	$O(\log N)$	×
SimplePIR [26]	LWE	$O(\sqrt{N})$	×
Yi et al. [45]	Approximate-GCD	$O(\log N)$	×
NewtonPIR	RSA	O(1)	1

Table 1: Comparison of communication complexity between NewtonPIR and existing schemes

modular multiplications. The time cost of modular additions is negligible compared to that of modular multiplications. Therefore, the query generation phase involves a total of 18 modular multiplications and some negligible modular additions.

The server primarily uses the Newton interpolation polynomial to generate the response value. In contrast to Scheme [44], which uses the Lagrange interpolation with the computational complexity linear in  $N^2$ , Newton interpolation allows for simpler construction of the basis, leading to linear complexity with respect to N. Additionally, with offline precomputation of interpolation coefficients and the capability to process N bivariate polynomials in parallel, the computational complexity for the server is significantly reduced.

#### 6.2 Experimental Evaluations

**Experimental setup**. We conducted our experiments on a server equipped with a 13th Gen Intel(R) Core(TM) i7-13700K processor, 16 cores, 64 GB of RAM, enabled with AVX, running Ubuntu 22.04. In contrast with previous work, our execution is single-threaded. We ran each experiment 10 times and reported averages from those 10 trials. Standard deviations are less than 10% of the reported means. NewtonPIR is implemented using C++ with the NTL Library and the crypto++ library [41]. For details of our experiments, we refer to https://github.com/pflu/NewtonPIR.

We also provide an overview of the server's monetary expenses, comprising the combined costs of CPU usage for server computations and the expenses associated with server-side network traffic. These calculations are based on established rates sourced from Amazon EC2 Instance prices [2], which currently stand at one cent per CPU-hour and nine cents per GB of Internet traffic at the time of writing.

**Baselines.** [5] provides a publicly available implementation of SealPIR, but it lacks certain functionalities. We can only evaluate the computation and communication costs using the data points presented in their paper [5]. For FastPIR,

Table 2: Performance of NewtonPIR scheme for different database sizes

N	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$2^{17}$	$2^{18}$
Setup	$1.84~\mathrm{s}$	$7.41~{\rm s}$	$15.72~\mathrm{s}$	$57.55~\mathrm{s}$	$126.16~\mathrm{s}$	$238.48~\mathrm{s}$	$323.48~\mathrm{s}$
Query Generation	$3.6 \ \mu s$	$3.8 \ \mu s$	$4.7 \ \mu s$	$5.9 \ \mu s$	$6.3 \ \mu s$	$6.4 \ \mu s$	$6.7 \ \mu s$
Response Generation	$1.03~{\rm s}$	$1.96~{\rm s}$	$2.46~{\rm s}$	$5.35~{\rm s}$	$9.67~{\rm s}$	$13.29~\mathrm{s}$	$19.12~\mathrm{s}$
Extraction Process	$1.56~\mathrm{ms}$	$2.12~\mathrm{ms}$	$3.56~\mathrm{ms}$	$3.66~\mathrm{ms}$	$4.53~\mathrm{ms}$	$4.75~\mathrm{ms}$	$4.59~\mathrm{ms}$
Server Cost (US cents)	0.00033	0.00059	0.00073	0.00153	0.00273	0.00374	0.00536

Table 3: Performance of Xu et al. scheme for different database sizes

N	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$2^{17}$
Response Generation	$9.87~\mathrm{h}$	$19.74~\mathrm{h}$	$39.48~\mathrm{h}$	$78.97~\mathrm{h}$	157.94 h	$315.88~\mathrm{h}$
Server Cost (US cents)	9.87004	19.74005	39.48005	78.97005	157.94005	315.88004

OnionPIR, and Spiral schemes, we use publicly available source codes [3,38,37] and integrate them into our testing framework.

**Computational cost**. For the number of database entries ranging from  $2^{12}$  to  $2^{18}$ , Table 2 shows the offline precomputation time, the computational cost of the three processes during the online phase, and the server cost for the NewtonPIR scheme. In Table 3, we separately list the response generation time and server costs for Xu *et al.* scheme [44] to facilitate comparison between the two schemes. In both tables, the size of the database entries used in the experiments is 1024 bits.

The Spiral scheme [37], based on the RLWE problem, takes an average of 2.46 seconds to generate response values when the database size is  $2^{18}$ . As shown in Table 2, our NewtonPIR scheme has a higher response generation time compared to the Spiral scheme. However, the computational cost for query generation and extraction in NewtonPIR are in the microsecond and millisecond range, respectively, which is significantly better than RLWE-based schemes like SealPIR, OnionPIR, and Spiral. Next, we compare NewtonPIR with non-RLWE-based schemes. The response time in Table 3 reaches an hourly level, greatly impacting the practicality of the scheme. With the precomputation in the offline phase, NewtonPIR reduces the computational cost of response generation by several orders of magnitude. In summary, the computational cost of our NewtonPIR scheme is close to that of RLWE-based schemes, significantly enhancing its practicality.

**Communication cost**. Table 4 compares the communication cost of Newton-PIR with recent PIR schemes. The "Rate" represents the ratio of the retrieved record size to the response size and a higher ratio indicates a smaller response. The experimental database contains  $2^{20}$  entries, with each entry being 256 bytes in size. Green represents better performance.

As shown in Table 4, our NewtonPIR scheme has the highest rate, with communication cost  $7.5 \times$  better than the state-of-the-art Spiral scheme. Compared to

Table 4: Comparison of communication overhead between NewtonPIR and existing schemes

	SealPIR[5]	MulPIR[4]	FastPIR[3]	OnionPIR[38]	Spiral[37]	NewtonPIR
Query Size	$67.7~\mathrm{KB}$	$123.5~\mathrm{KB}$	$33.5 \mathrm{MB}$	$64.2~\mathrm{KB}$	$15.9~\mathrm{KB}$	3.1 KB
Response Size	$329.8~\mathrm{KB}$	$120.8~\mathrm{KB}$	$66.6~\mathrm{KB}$	126.1  KB	$23.4~\mathrm{KB}$	$2.2~\mathrm{KB}$
Rate	0.00076	0.00207	0.00375	0.00198	0.01068	0.11364



Fig. 3: Communication comparison of NewtonPIR and existing schemes across different database sizes

Fig. 4: Communication comparison of NewtonPIR and existing schemes for varying entry size

other PIR schemes, NewtonPIR achieves a reduction in communication overhead by  $35.9 \sim 75 \times$ .

Figures 3 and 4 respectively compare the communication overhead of NewtonPIR with SealPIR, OnionPIR, and Spiral schemes under different database sizes and entry sizes. In Figure 3, the number of database entries ranges from  $2^{16}$  to  $2^{24}$ , with each entry being 256 bytes in size. In Figure 4, the database size is set as  $2^{20}$ .

As shown in Figure 3, the communication overhead of the SealPIR scheme increases with the size of the database. In contrast, the communication overhead of NewtonPIR, OnionPIR, and Spiral remains relatively stable, with NewtonPIR demonstrating significantly lower communication cost compared to the other two schemes. In Figure 4, as the entry size increases, the communication overhead of the SealPIR scheme increases significantly. The overhead of NewtonPIR and Spiral shows a slight rise, with NewtonPIR's communication volume starting to increase after the entry size exceeds 256 bytes. Although OnionPIR maintains a relatively steady overhead, its communication volume is higher than that of NewtonPIR. Overall, as the database and entry sizes increase, Newton-PIR's communication overhead remains stable and outperforms the other three schemes.

# 7 Conclusion

In this paper, we introduce the NewtonPIR protocol, a communication efficient single-server PIR scheme. The communication overhead of our scheme is independent of the database size N. Specifically, NewtonPIR's communication overhead is  $7.5 \times$  better than that of state-of-the-art PIR scheme and 35.9 to  $75 \times$  better than the other schemes. By leveraging the simplicity of constructing Newton interpolation polynomial bases and precomputing coefficients during the offline phase, NewtonPIR's computational overhead is comparable to that of RLWE-based PIR schemes. Additionally, we extend and construct a PSI protocol. Our future work will focus on further improving the computational efficiency of PIR protocols and exploring how to support database updates within PIR.

# References

- 1. NFLlib. https://github.com/quarkslab/NFLlib (2016)
- Amazon EC2 On-Demand Pricing. https://aws.amazon.com/ec2/pricing/ on-demand/ (2021), Amazon.
- Ahmad, I., Yang, Y., Agrawal, D., El Abbadi, A., Gupta, T.: Addra: Metadata-private voice communication over fully untrusted infrastructure. In: 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21) (2021)
- 4. Ali, A., Lepoint, T., Patel, S., Raykova, M., Schoppmann, P., Seth, K., Yeo, K.: {Communication–Computation} trade-offs in {PIR}. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1811–1828 (2021)
- Angel, S., Chen, H., Laine, K., Setty, S.: Pir with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 962–979 (2018). https://doi.org/10.1109/SP.2018.00062
- Angel, S., Setty, S.: Unobservable communication over fully untrusted infrastructure. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 551–569 (2016)
- Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers computation in private information retrieval: Pir with preprocessing. In: Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings 20. pp. 55–73. Springer (2000)
- Boyle, E., Ishai, Y., Pass, R., Wootters, M.: Can we access a database both locally and privately? In: Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15. pp. 662–693. Springer (2017)
- Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18. pp. 402–414. Springer (1999)
- Canetti, R., Holmgren, J., Richelson, S.: Towards doubly efficient private information retrieval. In: Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15. pp. 694–726. Springer (2017)

<sup>18</sup> Pengfei  $Lu^{(\boxtimes)}$  and Hongyuan Qu

- Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled psi from fully homomorphic encryption with malicious security. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1223–1237 (2018)
- Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1243–1255 (2017)
- Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of IEEE 36th Annual Foundations of Computer Science. pp. 41–50. IEEE (1995)
- Chor, B., Gilboa, N.: Computationally private information retrieval. In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. pp. 304–313 (1997)
- Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. In: 32nd USENIX security symposium (USENIX Security 23). pp. 3835–3851 (2023)
- Cong, K., Moreno, R.C., da Gama, M.B., Dai, W., Iliashenko, I., Laine, K., Rosenberg, M.: Labeled psi from homomorphic encryption with reduced computation and communication. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1135–1150 (2021)
- 17. Corrigan-Gibbs, H., Henzinger, A., Kogan, D.: Single-server private information retrieval with sublinear amortized time. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–33. Springer (2022)
- Corrigan-Gibbs, H., Kogan, D.: Private information retrieval with sublinear online time. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39. pp. 44–75. Springer (2020)
- 19. Davidson, A., Pestana, G., Celi, S.: Frodopir: Simple, scalable, single-server private information retrieval. Proceedings on Privacy Enhancing Technologies (2023)
- Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2. pp. 303–324. Springer (2005)
- 21. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
- Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Symposium on Theory of Computing-STOC 2009. pp. 169–178. ACM (2009)
- Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Automata, Languages and Programming: 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005. Proceedings 32. pp. 803–815. Springer (2005)
- Green, M., Ladd, W., Miers, I.: A protocol for privately reporting ad impressions at scale. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1591–1601 (2016)
- Gupta, T., Crooks, N., Mulhern, W., Setty, S., Alvisi, L., Walfish, M.: Scalable and private media consumption with popcorn. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). pp. 91–107 (2016)
- Henzinger, A., Hong, M.M., Corrigan-Gibbs, H., Meiklejohn, S., Vaikuntanathan, V.: One server for the price of two: Simple and fast single-server private information retrieval. In: Usenix Security. vol. 23 (2023)
- 27. Howarth, J.: Alarming data privacy statistics for 2024. https://explodingtopics.com/blog/data-privacy-stats (2024)

- 20 Pengfei  $Lu^{(\boxtimes)}$  and Hongyuan Qu
- Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: Annual Cryptology Conference. pp. 18–35. Springer (2013)
- Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. pp. 262–271 (2004)
- Kales, D., Rechberger, C., Schneider, T., Senker, M., Weinert, C.: Mobile private contact discovery at scale. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1447–1464 (2019)
- Kiraz, M., Schoenmakers, B.: A protocol issue for the malicious case of yao's garbled circuit construction. In: 27th Symposium on Information Theory in the Benelux. vol. 29, pp. 283–290. IEEE (2006)
- Kogan, D., Corrigan-Gibbs, H.: Private blocklist lookups with checklist. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 875–892 (2021)
- 33. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: Proceedings 38th annual symposium on foundations of computer science. pp. 364–373. IEEE (1997)
- 34. Liu, J., Li, J., Wu, D., Ren, K.: PIRANA: Faster multi-query PIR via constantweight codes. Cryptology ePrint Archive, Paper 2022/1401 (2022), https:// eprint.iacr.org/2022/1401
- 35. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 1–23. Springer (2010)
- Melchor, C.A., Barrier, J., Fousse, L., Killijian, M.O.: Xpir: Private information retrieval for everyone. Proceedings on Privacy Enhancing Technologies pp. 155–174 (2016)
- Menon, S.J., Wu, D.J.: Spiral: Fast, high-rate single-server pir via fhe composition. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 930–947. IEEE (2022)
- Mughees, M.H., Chen, H., Ren, L.: Onionpir: Response efficient single-server pir. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2292–2306 (2021)
- Mughees, M.H., Ren, L.: Vectorized batch private information retrieval. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 437–452. IEEE (2023)
- 40. Sion, R., Carbunar, B.: On the computational practicality of private information retrieval. In: Proceedings of the network and distributed systems security symposium. pp. 2006–06. Internet Society Geneva, Switzerland (2007)
- Tuleuov, B.I., Ospanova, A.B.: Crypto++. In: Beginning C++ Compilers: An Introductory Guide to Microsoft C/C++ and MinGW Compilers, pp. 149–156. Springer (2024)
- Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 24–43. Springer (2010)
- 43. Wu, D.J., Zimmerman, J., Planul, J., Mitchell, J.C.: Privacy-preserving shortest path computation. arXiv preprint arXiv:1601.02281 (2016)
- 44. Xu, W., Wang, B., Lu, R., Qu, Q., Chen, Y., Hu, Y.: Efficient private information retrieval protocol with homomorphically computing univariate polynomials. Security and Communication Networks **2021**, 1–14 (2021)

45. Yi, X., Kaosar, M.G., Paulet, R., Bertino, E.: Single-database private information retrieval from fully homomorphic encryption. IEEE Transactions on Knowledge and Data Engineering **25**(5), 1125–1134 (2012)