Hindawi
Security and Communication Networks
Volume 2021, Article ID 5553256, 14 pages
https://doi.org/10.1155/2021/5553256

Research Article

Efficient Private Information Retrieval Protocol with Homomorphically Computing Univariate Polynomials

Wenju Xu,[1] **Baocang Wang**,[1,2] **Rongxing Lu**,[3] **Quanbo Qu**,[1] **Yange Chen**,[1,4] **and Yupu Hu**[1]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China*
[2]*Cryptographic Research Center, Xidian University, Xi'an 710071, China*
[3]*Faculty of Computer Science, University of New Brunswick, Fredericton NB E3B 5A3, Canada*
[4]*School of Information Engineering, Xuchang University, Xuchang 461000, China*

Correspondence should be addressed to Baocang Wang; bcwang79@aliyun.com

Received 14 January 2021; Revised 7 April 2021; Accepted 15 April 2021; Published 28 April 2021

Academic Editor: Leandros Maglaras

Private information retrieval (PIR) protocol is a powerful cryptographic tool and has received considerable attention in recent years as it can not only help users to retrieve the needed data from database servers but also protect them from being known by the servers. Although many PIR protocols have been proposed, it remains an open problem to design an efficient PIR protocol whose communication overhead is irrelevant to the database size N. In this paper, to answer this open problem, we present a new communication-efficient PIR protocol based on our proposed single-ciphertext fully homomorphic encryption (FHE) scheme, which supports unlimited computations with single variable over a single ciphertext even without access to the secret key. Specifically, our proposed PIR protocol is characterized by combining our single-ciphertext FHE with Lagrange interpolating polynomial technique to achieve better communication efficiency. Security analyses show that the proposed PIR protocol can efficiently protect the privacy of the user and the data in the database. In addition, both theoretical analyses and experimental evaluations are conducted, and the results indicate that our proposed PIR protocol is also more efficient and practical than previously reported ones. To the best of our knowledge, our proposed protocol is the first PIR protocol achieving $O(1)$ communication efficiency on the user side, irrelevant to the database size N.

1. Introduction

Private information retrieval (PIR) protocol [1] is a cryptographic primitive run between database servers and a user. The salient feature of PIR is that it ensures the user can obtain some data from the database servers, while the database servers cannot learn anything about the queries of the user. To obtain the feature, a trivial solution for the user is to download all the data from the database servers and obtain the data he wants to ask at any time. However, this solution wastes plenty of time and storage space for the user since the database servers usually store a huge volume of items. In addition, considering that there are continuous interactions with multiservers at the price of communication costs for the user, many research studies have been focused on the single-server PIR protocol that is composed of only one database server and one query user [1–7].

In 1997, the first single-server PIR protocol was proposed by Kushilevitz and Ostrovsky [2]. They constructed a PIR protocol based on group homomorphism and the quadratic residuosity problem and achieved the communication complexity $O(N^{\varepsilon}\log N)$ bits (the symbols $O(\cdot), \Omega(\cdot)$, and $o(\cdot)$ are commonly used asymptotic complexity notations. We denote an asymptotic upper bound, noncompact upper bound, and lower bound with $O(\cdot), o(\cdot)$, and $\Omega(\cdot)$, respectively) on the user side for database size N and any constant ε. After that, some single-server PIR protocols were also proposed [3–5]. Kushilevitz and

Ostrovsky [3] applied the trapdoor permutation approach to the single-server PIR protocol with communication overhead $N - cN/k + O(k^2)$ bits, where c is a constant and k is the security parameter of the one-way trapdoor permutation. Gentry and Ramzan [4] presented a single-server PIR protocol based on a slight variation of the computational difficulty of deciding whether a small prime divides Euler's totient function of any composite integer. The total communication cost of the protocol is 3 messages, each of the size of $\Omega(\log^{3-o(1)} N)$ bits. A PIR protocol was proposed based on group homomorphism by Melchor et al. [5] of communication $O(\sqrt{N})$ bits.

In recent years, with the development of fully homomorphic encryption (FHE) [8, 9], many researchers have turned into utilizing the FHE schemes to construct the single-server PIR protocols [6, 7, 10, 11]. Brakerski and Vaikuntanathan [10] proposed a brief PIR protocol based on learning with errors (LWE) by using FHE. The FHE DGHV [12] over the integers was applied to the PIR protocols by Yi et al. [6]. The communication overhead of the PIR protocol is $O(\log N)$ bits and also relies on the size of ciphertext $O(\lambda^5)$ (the security parameter λ) in DGHV. Li et al. [7] modified Brakerski and Vaikuntanathan's PIR protocol [10] and united the HAO scheme in [13] to construct a PIR protocol. However, the main idea of the protocol is similar to invoking the decryption circuit homomorphically, which is expensive and of extremely low efficiency. Aiming at single-server PIR protocols, we notice that all the aforementioned PIR protocols depend on the database size N in terms of communication cost. When the size N becomes larger, the communication will not be efficient. Therefore, how to efficiently design a PIR protocol with communication overhead $O(1)$, i.e., independent on the database size N, becomes an open problem.

In this paper, to address the above open problem, we propose a new FHE scheme with special properties and utilize it to design a new single-server PIR protocol with $O(1)$ communication efficiency for any user. To the best of our knowledge, our single-server PIR protocol is the most efficient one in terms of the communication efficiency. In addition, our single-server PIR protocol also allows a user to retrieve positive integer data from the database server, instead of a single bit for every query. Specifically, the main contributions of this paper are threefold:

(i) First, in order to achieve $O(1)$ communication efficiency on the user side, we design a new kind of FHE scheme called single-ciphertext FHE, which supports unlimited computations with single variable over a single ciphertext without access to the secret key. Our proposed single-ciphertext FHE scheme is characterized with extremely efficient in terms of both encryption and decryption dependent on the truncated polynomial ring. Detailed security analysis illustrates that the proposed FHE scheme is one-way secure, which is exactly equivalent to the 3rd RSA problem.

(ii) Second, we take the single-ciphertext FHE as a symmetric encryption scheme and the Lagrange interpolating polynomial technique to construct our single-server PIR protocol. Security analyses show that our proposed PIR protocol can efficiently protect the privacy of the user and the data in the database in our defined security model.

(iii) Third, we conduct both theoretical analyses and experimental evaluations to demonstrate that our proposed PIR protocol is indeed efficient in terms of computational complexity and communication overhead. In particular, our proposed protocol is the first PIR protocol, which can achieve $O(1)$ communication efficiency, irrelevant to the database size N.

The remainder of this paper is organized as follows. We describe some preliminaries in Section 2. Then, in Section 3, we formalize our system model, security model, and design goal. In Section 4, we first present a new single-ciphertext FHE scheme, followed by our single-server communication-efficient PIR protocol. After that, the security analyses and the performance evaluation of our single-server PIR protocol are given in Sections 5 and 6, respectively. Some related works are also discussed in Section 7. Finally, we draw our conclusion in Section 8.

2. Preliminaries

In this section, we first give some notations that will be used throughout this paper and then describe the definitions of the truncated polynomial rings and our proposed single-ciphertext FHE scheme.

2.1. Notations. In this paper, we denote row vectors by bold letters (e.g., **DB** and **c**), and the symbol $\mathbf{DB}[i]$ represents the i-th data in **DB**. Some other notations that will be used in this work are listed in Table 1.

2.2. Truncated Polynomial Rings. The truncated polynomial rings will be used as a building block for constructing a special FHE scheme in this work. Essentially, the concept of truncated polynomials is not quite complicated, e.g., an extension field is constructed from $\mathbb{F}[x]$ defined over a finite field $\mathbb{F}$ modulo a monic irreducible polynomial [14], and the NTRU public key cryptosystem [15] also utilizes a univariate truncated polynomial ring modulo $X^N - 1$. Though the above examples only involve univariate polynomials, we can extend the situations to the case of bivariate polynomials.

To be specific, we can set $n = pq$ to be a standard RSA modulus, namely, n is the product of two large primes p and q. In order to make our proposal more efficient, we consider the RSA cryptosystem [16] with the encryption public key $e = 3$, from which we define two polynomials $f(x) = x^3 - u \pmod n$ and $g(y) = y^3 - v \pmod n$ with $u, v \in \mathbb{Z}_n$. We also define a bivariate polynomial set

$$S = \left\{ h(x, y) = \sum_{i=0}^{2} \sum_{j=0}^{2} h_{ij} x^i y^j \mid \quad h_{ij} \in \mathbb{Z}_n \right\} \tag{1}$$

Table 1: Notations.

Notation	Explanation
$\lvert a\rvert$	The binary length of an integer a
$\gcd(a,b)$	The greatest common divisor of integers a, b
$\varphi(n)$	The Euler function of integer n
$\mathbb{Z}$	The integer ring
$\mathbb{Z}_n$	$\{0,1,\ldots,n-1\}$, the ring of integers modulo n
$\mathbb{Z}_n[x,y]$	The bivariate polynomial ring on $\mathbb{Z}_n$

and the additive and multiplicative operations on S. Given two bivariate polynomials $h_1(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} h_{ij}^{(1)} x^i y^j$ and $h_2(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} h_{ij}^{(2)} x^i y^j$, the sum of $h_1(x,y)$ and $h_2(x,y)$ is defined as $h^+(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} h_{ij}^{+} x^i y^j$, where $h_{ij}^{+} \equiv h_{ij}^{(1)} + h_{ij}^{(2)} \pmod n$. The multiplication can also be defined as $h^*(x,y) \equiv h_1(x,y)h_2(x,y) \pmod{n, f(x), g(y)}$. To perform the multiplication, we first carry out the standard polynomial multiplication $h_1(x,y)h_2(x,y)$ on $\mathbb{Z}_n[x,y]$. Because the maximum degree with respect to x (y, respectively) is 2 in $h_1(x,y)$ ($h_2(x,y)$, respectively), the maximum degree of x (y, respectively) in the multiplication $h_1(x,y)h_2(x,y)$ becomes 4. Thus, in the second step, we perform modulo $f(x) = x^3 - u$ and $g(y) = y^3 - v$ on the multiplication $h_1(x,y)h_2(x,y)$ to truncate it back to the set S as follows: replace x^3 (y^3, respectively) with u (v, respectively), and replace x^4 (y^4, respectively) with ux (vy, respectively). From the definition, one can easily verify that $\mathbb{Z}_n[x,y] \pmod{f(x), g(y)}$ also forms a ring called truncated polynomial ring, and it is denoted as $\mathbb{Z}_n[x,y]/\langle f(x), g(y)\rangle$.

2.3. Single-Ciphertext FHE Scheme. In the following, we will formalize the definition of single-ciphertext FHE, together with its security notion. Before that, we first give some necessary descriptions of the special FHE.

The proposed single-ciphertext FHE is a special kind of FHE, which supports unlimited computations with single variable over a single ciphertext without access to the secret key. Different from the general FHE, the evaluation algorithm of our single-ciphertext FHE is subject to performing upon a single ciphertext rather than any multiciphertexts. In other words, our single-ciphertext FHE skips (or aborts) any circuits with multivariables for the general FHE and allows any computations over any circuits with single variable. Compared with the general FHE, our single-ciphertext FHE possesses less functionality due to the single ciphertext, but it still permits any computations on any circuits with single variable. Hence, our single-ciphertext FHE, as a well-suitable cryptographic tool, is enough for the requirements of single-server PIR protocols since the evaluation of the single-server PIR protocols can be regarded as univariate polynomials.

Definition 1. (single-ciphertext FHE scheme). A single-ciphertext FHE scheme consists of four probabilistic polynomial time (PPT) algorithms, namely, key generation, encryption, decryption, and homomorphic evaluation algorithm. The details are as follows:

(i) Key generation ($\mathrm{pk}, \mathrm{evk}, \mathrm{sk} \leftarrow \mathrm{KeyGen}(\lambda)$): take the security parameter λ as the input, and output a public key pk, an evaluation key evk, and a secret key sk

(ii) Encryption ($c \leftarrow \mathrm{Enc}(m, \mathrm{pk})$): using the public key pk, encrypt a message $m \in \mathbb{M}$ into a ciphertext c, where $\mathbb{M}$ is the message space

(iii) Decryption ($m \leftarrow \mathrm{Dec}(c, \mathrm{sk})$): using the secret key sk, decrypt a ciphertext c to recover the corresponding message $m \in \mathbb{M}$

(iv) Evaluation ($\hat{c} \leftarrow \mathrm{Eval}(\mathscr{C}, c, \mathrm{evk})$): given a circuit with single variable $\mathscr{C}$ and a ciphertext c with the underlying plaintext m, i.e., $c = \mathrm{Enc}(m, \mathrm{pk})$, the algorithm utilizes the evaluation key evk to compute a new ciphertext $\hat{c} = \mathrm{Eval}(\mathscr{C}, c, \mathrm{evk})$

Note that the correctness of decryption requires that the plaintext m can be correctly decrypted from the ciphertext, i.e., $m = \mathrm{Dec}(\mathrm{Enc}(m, \mathrm{pk}), \mathrm{sk})$. The correctness of the homomorphic evaluation requires that the ciphertext $\hat{c}$ can be correctly decrypted into the plaintext $\mathscr{C}(m)$, namely, $\mathrm{Dec}(\hat{c}, \mathrm{sk}) = \mathscr{C}(m)$.

Actually, our proposal single-ciphertext FHE scheme performs no noises. Every time an evaluation on the ciphertext is performed, there is no noise to obscure the underlying plaintext. In terms of the noiseless FHE schemes, there is a main drawback: none can be strictly proved secure and feasible in the framework of provable security. For more introduction of noiseless FHE, one can refer to Section 7. Hence, we give the following security definition for our noiseless single-ciphertext FHE scheme.

Definition 2. (the one-way security of single-ciphertext FHE). Given the security parameter λ, the public key pk, the evaluation key evk, and a ciphertext c with the underlying plaintext m, it should be difficult for any PPT adversary to find $m \in \mathbb{M}$ from the ciphertext c such that $c = \mathrm{Enc}(m, \mathrm{pk})$. Formally, we require that for any PPT adversary $\mathscr{A}$, we have

$$\Pr[m \in \mathbb{M} \mid \mathrm{pk}, \mathrm{evk} \leftarrow \mathrm{KeyGen}(\lambda), c = \mathrm{Enc}(m, \mathrm{pk})] \le \varepsilon(\lambda), \tag{2}$$

where $\varepsilon(\lambda)$ represents a negligible function.

Different from general security notions such as indistinguishability under chosen-plaintext attack (IND-CPA) and indistinguishability under chosen-ciphertext attack (IND-CCA1) of known FHE schemes [8–10, 12, 17–22] (FHE essentially supports malleability on ciphertexts and hence cannot obtain the highest security goal, namely, indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2)), we only consider the one-way security due to the following observations. Firstly, the security notion is tailored for the single-ciphertext FHE scenarios, where no distinguishability games are permitted on distinct plaintexts. Secondly, in the PIR scenario, the single-ciphertext FHE scheme is used as a symmetric encryption algorithm without considering the IND-CPA security in the public key encryption schemes.

3. System Model, Security Model, and Design Goal

In this section, we formalize our system model and security model and identify our design goal.

3.1. System Model. In our system model, we consider a typical single-server PIR protocol, which includes two entities, namely, a user and a database (DB) server, as shown in Figure 1.

(i) DB server: the database server is powerful in both storing and computing data. In our system model, the database server stores and processes a database $\mathbf{DB} = \{(i, \mathbf{DB}[i]) | 0 \le i \le N-1\}$ with totally N items. For simplicity of our PIR protocol discussed later, we assume the value of each item $\mathbf{DB}[i]$ is a positive integer, not just one bit. In addition, the server will offer a PIR response to a query user after the latter makes a PIR query with unbounded computations.

(ii) User: in our system model, we consider a query user can directly make a PIR query to the DB server and obtain the desirable result from the DB server. Meanwhile, the user does not want to reveal the queried value i to the DB server when asking the corresponding data $\mathbf{DB}[i]$ from $\mathbf{DB}$ and hopes the communication of PIR should be efficient.

Formally, a single-server PIR protocol in our system model comprises three phases as follows:

(i) Query generation phase ($Q(i) \leftarrow \mathrm{QG}(i)$): taking the index i as the input, the user sends a query $Q(i)$ to the server

(ii) Response generation phase ($R(i) \leftarrow \mathrm{RG}(Q(i), \mathbf{DB})$): using the query $Q(i)$ and the database $\mathbf{DB}$, the server returns a response $R(i)$ to the user

(iii) Response retrieval phase ($\mathbf{DB}[i] \leftarrow \mathrm{RR}(R(i))$): upon receiving a response $R(i)$, the user outputs the data $\mathbf{DB}[i]$ corresponding to the index i

A single-server PIR protocol is correct if for any database $\mathbf{DB}$ with any size N and any index i for $0 \le i \le N-1$, $\mathbf{DB}[i] = \mathrm{RR}(\mathbf{DB}, i, Q(i), R(i))$ holds, where $Q(i) = \mathrm{QG}(i)$ and $R(i) = \mathrm{RG}(Q(i), \mathbf{DB})$.

3.2. Security Model. In our security model, we consider the DB server is honest but curious, and there is no collusion between the DB server and any other third parties. In other words, the DB server will faithfully follow the protocol; however, he is curious about the queried value of the user. Note that, in case the DB server is compromised by some attackers, the compromised DB server may launch other active attacks and return a response with errors to the user who is not able to verify. However, since we focus on the communication-efficient PIR protocol for the user in this paper, those active attacks from the compromised DB server are beyond the major work of this paper, though it is not difficult to apply some verifiable techniques to tackle these attacks. For details, one can refer to Remark 3 in Section 5.

3.3. Design Goal. Our design goal is to present a communication-efficient PIR protocol on the user side to address the requirements mentioned in the above system model and security model. The communication-efficient PIR protocol is the center of our attention; hence, we assume the power of the server is unlimited, and the computation burden of the server is less important than the one of any user. Specifically, the following two objectives should be included:

(i) The proposed PIR protocol should be privacy preserving: the queried index i should be private, and no one, except the query user, can determine the value of i. In addition, no one, except the query user, can retrieve the data $\mathbf{DB}[i]$ after receiving the response $R(i)$ returned by the DB server.

(ii) The proposed PIR protocol should be communication efficient: in order to achieve the above privacy requirement, additional communication costs will be incurred in the PIR protocols. Therefore, in the proposed PIR protocol, we aim to make the query's communication efficient, i.e., achieving less communication costs for the user.

4. Our Proposed Scheme

In this section, we will describe our communication-efficient PIR protocol. Before delving into the details, we first present our new single-ciphertext FHE scheme based on the aforementioned truncated polynomial rings.

4.1. Our New Single-Ciphertext FHE Scheme. Our new single-ciphertext FHE scheme comprises four algorithms, namely, KeyGen, Enc, Dec, and Eval algorithms. The detailed descriptions are as follows:

(i) KeyGen(λ): taking the security parameter λ (even for simplicity) as the input, randomly generate two $\lambda/2$-bit large primes p and q satisfying $\gcd(p-1, 3) = 1$ and $\gcd(q-1, 3) = 1$, and compute $n = pq$, $\varphi(n) = (p-1)(q-1)$ and the inverse d of 3 modulo $\varphi(n)$, namely, $3d \equiv 1 \pmod{\varphi(n)}$. The modulus n is set as the public key $\mathrm{pk} = n$, the evaluation key is set as $\mathrm{evk} = n$, and the integer d is set as the secret key, i.e., $\mathrm{sk} = d$.

(ii) Enc(m, pk): given a plaintext $m \in \mathbb{M} = \mathbb{Z}_n$, randomly choose $a, b \in \mathbb{Z}_n$ and compute $u \equiv a^3 \pmod{n}$ and $v \equiv b^3 \pmod{n}$. Also, randomly choose 9 integers $a_{ij} \in \mathbb{Z}_n$ for $i, j \in \{0, 1, 2\}$, and construct a polynomial $f(x, y) = \sum_{i=0}^{2} \sum_{j=0}^{2} a_{ij} x^i y^j$. Set a polynomial $F(x, y) \equiv f(x, y) - f(a, b) \pmod{n}$, and compute $c(x, y) \equiv F(x, y) + m \pmod{n}$. The ciphertext is $\mathbf{c} = \mathrm{Enc}(m, n) = (u, v, c(x, y))$.

(iii) Dec($\mathbf{c}$, sk): upon the receipt of a ciphertext $\mathbf{c} = (u, v, c(x, y))$, compute $a \equiv u^d \pmod{n}$ and $b \equiv v^d \pmod{n}$ with the secret key d to obtain the two

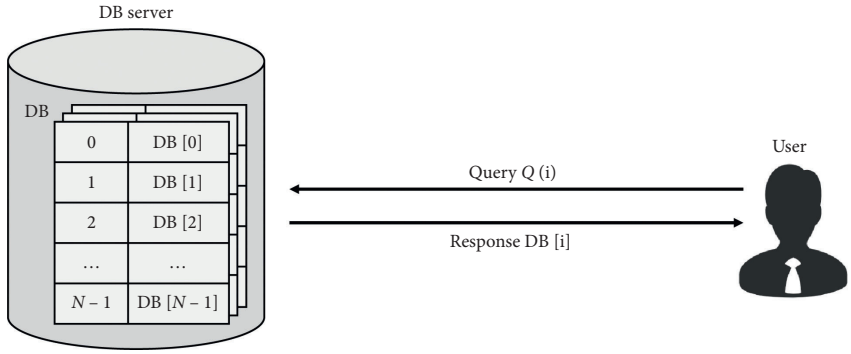

Figure 1: System model under consideration.

random numbers a, b. The plaintext can be recovered by substituting a, b into $c(x, y)$, that is, $c(a, b) \equiv F(a, b) + m \equiv m (\mathrm{mod} n)$ due to $F(a, b) \equiv f(a, b) - f(a, b) \equiv 0 (\mathrm{mod} n)$.

(iv) Eval$(\widehat{f}, \mathbf{c}, \mathrm{evk})$): given the ciphertext $\mathbf{c} = \mathrm{Enc}(m, n) = (u, v, c(x, y))$ and a univariate polynomial $\widehat{f}(x) = \sum_{i=0}^{\alpha} \beta_i x^i \in \mathbb{Z}_n[x]$, the evaluation algorithm is described in Algorithm 1. We remind that the involved addition and multiplication operations are performed over the truncated polynomial ring $\mathbb{Z}_n[x, y]/\langle x^3 - u, y^3 - v\rangle$. Especially, for each iteration, $c_{\widehat{f}}(x, y)$ will be truncated back to the truncated polynomial ring via the reduction operation modulo $x^3 - u$ and $y^3 - v$. Hence, the final result of $c_{\widehat{f}}(x, y)$ remains in the truncated polynomial ring $\mathbb{Z}_n[x, y]/\langle x^3 - u, y^3 - v\rangle$, i.e., it remains a bivariate polynomial in S.

Remark 1. Note that, in order to ensure the one-way security of our single-ciphertext FHE scheme, the length of the modulus n should be larger than 2048 bits, i.e., $\lambda \geq 2048$.

Correctness: in order to demonstrate the correctness of the homomorphic evaluation algorithm, we need to show that $\mathrm{Dec}(c_{\widehat{f}}(x, y), d) = \widehat{f}(m)$. From Algorithm 1, one can easily verify that

$$c_{\widehat{f}}(x, y) \equiv \widehat{f}(c(x, y)) \equiv \sum_{i=0}^{\alpha} \beta_i c(x, y)^i \left(\mathrm{mod} n, x^3 - u, y^3 - v\right). \tag{3}$$

Then, there must exist two bivariate polynomials $A(x, y), B(x, y) \in \mathbb{Z}_n[x, y]$ such that

$$\begin{aligned} c_{\widehat{f}}(x, y) \equiv \widehat{f}(c(x, y)) &+ A(x, y)\left(x^3 - u\right) \\ &+ B(x, y)\left(y^3 - v\right) (\mathrm{mod} n). \end{aligned} \tag{4}$$

Since $a^3 \equiv u (\mathrm{mod} n)$ and $b^3 \equiv v (\mathrm{mod} n)$, we have

$$\begin{aligned} \mathrm{Dec}\left(c_{\widehat{f}}(x, y), d\right) = c_{\widehat{f}}(a, b) &\equiv \widehat{f}(c(a, b)) \\ &+ A(a, b)\left(a^3 - u\right) \\ &+ B(a, b)\left(b^3 - v\right) (\mathrm{mod} n). \end{aligned} \tag{5}$$

Recall that $c(a, b) \equiv m (\mathrm{mod} n)$, $a^3 - u \equiv 0 (\mathrm{mod} n)$, and $b^3 - v \equiv 0 (\mathrm{mod} n)$, and we immediately have $\mathrm{Dec}(c_{\widehat{f}}(x, y), d) = c_{\widehat{f}}(a, b) \equiv \widehat{f}(m) (\mathrm{mod} n)$ as desired.

Security: in the following, we prove our proposed single-ciphertext FHE scheme is one-way secure based on the hardness of the 3rd RSA problem.

Definition 3. (the 3rd RSA problem). The e-th RSA problem is defined as follows: given the RSA public key $n = pq$ and e, and a ciphertext π, to find the plaintext μ such that $\pi \equiv \mu^e (\mathrm{mod} n)$. The 3rd RSA problem is the special case with $e = 3$.

Theorem 1. *The one-way security of our proposed single-ciphertext FHE scheme is polynomially equivalent to the 3rd RSA problem.*

Proof. Both directions ($\Leftrightarrow$) need to be proven. The direction from the right to the left ($\Leftarrow$) is trivial. If an adversary can break the 3rd RSA problem, then given a ciphertext $\mathbf{c} = (u, v, c(x, y))$ of the single-ciphertext FHE scheme, the adversary can solve two 3rd RSA problems $a^3 \equiv u (\mathrm{mod} n)$ and $b^3 \equiv v (\mathrm{mod} n)$ to derive two integers $a, b \in \mathbb{Z}_n$ and finally breaks the one-way security by computing $m \equiv c(a, b) (\mathrm{mod} n)$.

In order to prove the direction from the left to the right ($\Rightarrow$), we assume there is a PPT adversary $\mathcal{A}$ which can break the one-way security of our scheme, i.e., $m \leftarrow \mathcal{A}(\mathbf{c}, n)$. Then, we can construct another algorithm $\mathcal{B}$ which can utilize $\mathcal{A}$ to break the 3rd RSA problem, i.e., $\mu \leftarrow \mathcal{B}(\pi, n)$, as shown in Algorithm 2.

To prove the correctness of the reduction in Algorithm 2, we first note that $u \equiv \pi \equiv \mu^3 (\mathrm{mod} n)$ and that m is the plaintext corresponding to $\mathbf{c} = (u, v, c(x, y))$, so there must exist a bivariate polynomial $F(x, y) \in \mathbb{Z}_n[x, y]/\langle x^3 - u, y^3 - v\rangle$ such that $c(x, y) \equiv F(x, y) + m (\mathrm{mod} n)$ and $F(\mu, b) \equiv 0 (\mathrm{mod} n)$. So, $x \equiv \mu (\mathrm{mod} n)$ is a common root for both congruences $x^3 - u \equiv x^3 - \pi \equiv 0 (\mathrm{mod} n)$ and $c(x, b) - m \equiv 0 (\mathrm{mod} n)$. Thus, we can efficiently perform the Euclidean algorithm [23] to compute the greatest common divisor $x - \mu \equiv \gcd(x^3 - u (\mathrm{mod} n), c(x, b) - m (\mathrm{mod} n))$. So, the plaintext μ of the RSA problem is recovered, i.e., we can construct an algorithm $\mathcal{B}$ for solving the 3rd RSA problem.

Input: $\widehat{f} = \sum_{i=0}^{\alpha} \beta_i x^i \pmod n$, $\mathbf{c} = (u, v, c(x, y))$, evk $= n$.
(1) Set $c_{\widehat{f}}(x, y) = 0$.
(2) For each $i = 0, \ldots, \alpha$, compute
$c_{\widehat{f}}(x, y) = c_{\widehat{f}}(x, y)c(x, y) + \beta_{\alpha-i} \pmod{n, x^3 - u, y^3 - v}$.
Output: $\mathbf{c}_{\widehat{f}} = (u, v, c_{\widehat{f}}(x, y))$.

Algorithm 1: Univariate polynomial evaluation algorithm.

Input: the public modulus n, and the RSA ciphertext $\pi \in \mathbb{Z}_n$.
(1) Randomly choose an integer $b \in \mathbb{Z}_n$ and compute $v \equiv b^3 \pmod n$.
(2) Set $u = \pi$ and randomly generate a bivariate polynomial $c(x, y) \in \mathbb{Z}_n[x, y]/\langle x^3 - u, y^3 - v\rangle$.
(3) Set $\mathbf{c} = (u, v, c(x, y))$ and run $m \leftarrow \mathscr{A}(\mathbf{c}, n)$.
(4) Compute the greatest common divisor $x + \theta \equiv \gcd(x^3 - u \pmod n, c(x, b) - m \pmod n)$.
Output: $\mu \equiv -\theta \pmod n$.

Algorithm 2: Algorithm $\mathscr{B}$ with access to $\mathscr{A}$.

Note that Theorem 1 establishes an exact equivalence between the one-wayness of the proposed single-ciphertext FHE scheme and the 3rd RSA problem. One may doubt that choosing the RSA encryption key as 3 will produce serious threats on the security of the single-ciphertext FHE scheme. In fact, in many implementations, choosing a relatively small encryption key such as $e = 3$ or $2^{17} + 1$ is widely suggested to reduce the encryption costs.

Computational complexity: next, we analyze the computational costs of our single-ciphertext FHE scheme.

During the Enc phase, there are 2 modular multiplications to compute u (v, respectively). Computing the polynomial $F(x, y)$ needs 18 modular multiplications and some modular additions since there are $i + j$ modular multiplications to compute the monomial $a_{ij}x^i y^j$ for $i, j = 0, 1, 2$ in $F(x, y)$. Compared with the calculation of the modular multiplication, the time cost of modular addition can be negligible. Hence, there are totally 22 modular multiplications and some negligible modular additions in the Enc phase. Considering the computational complexity of a multiplication modulo $n = pq$ is $O(\lambda^2)$, we conclude that the computational complexity of the Enc phase is $O(\lambda^2)$.

During the Dec phase, the main operations are to output a, b from u, v by 2 modular exponentiation operations of exponentiation d. Considering that the computational complexity of a modular exponentiation is $O(\lambda^3)$, the total computational complexity of the Dec phase is $O(\lambda^3)$ when ignoring some modular additions.

During the Eval phase, the output $c_{\widehat{f}}(x, y)$ is actually a truncated bivariable polynomial. There are α-iterations, and every iteration performs a modular multiplication besides a negligible modular addition. Hence, there are totally α-modular multiplications and some negligible modular additions. As a result, the computational complexity of the Eval phase is $O(\alpha\lambda^2)$ subject to the value of α in ciphertext evaluations.

In summary, the computational complexity is $O(\lambda^2)$ for encryption, $O(\lambda^3)$ for decryption, and $O(\alpha\lambda^2)$ for evaluation, respectively, where $\lambda \geq 2048$ is the length of the RSA modulus n.

Comparisons of several noiseless FHE: comparisons of several noiseless FHE schemes among [24–26] with ours are shown in Table 2. Nuida utilized the commutator and an encoding scheme by a homomorphic mapping φ from noncommutative group G to noncommutative group $\widehat{G}$ to construct the noiseless FHE. The ciphertext is composed of two elements from G and Ker (φ) which is a subset of G. However, the security is based on the open sampling of group G, and the assumption that judging whether an element is in the kernel Ker (φ) is difficult. Yagisawa [25] is an improved version of [26] with smaller ciphertext size; hence, we only discuss about [25]. The octonion ring over the finite field was used by Yagisawa to achieve 1: 8 length ratio of the plaintext and ciphertext. Yagisawa's noiseless FHE is immune from the Grobner basis attacks, which is weaker than our one-way security. With respect to ciphertext space and length ratio, our noiseless FHE is more efficient than [25] while less than [24]. Totally speaking, our single-ciphertext FHE scheme is more superior to [24–26], especially considering that the security is more important than other factors for noiseless FHE. □

4.2. Description of Our Communication-Efficient PIR Protocol. Before delving our communication-efficient PIR protocol, we first give a brief overview of how our single-ciphertext FHE scheme is utilized to construct the single-server PIR protocol.

(i) The single-server PIR protocol aims to help the user to obtain the ith data from the server possessing the whole database, without leaking the index i to the server. Obviously, the server performs an evaluation algorithm on a single ciphertext corresponding to

Table 2: Comparisons of several noiseless FHE schemes.

Schemes	Algebraic structure	Ciphertext space	Length ratio a	Security
Nuida [24]	Noncommutative group	Vector of dimension 2	$1:2\|G\|^b$	Judgement of the kernel element
Yagisawa [25]	Octonion ring	Octonion ring	1 : 8	Immune from the Grobner basis attacks
Ours	Truncated polynomial ring	Vector of dimension 3	1 : 11	One-way security

Length ratio a: the length ratio between the plaintext and ciphertext. $|G|^b$: the number of elements in group G.

the queried index. So, our single-ciphertext FHE scheme is well suitable for the single-server PIR protocol.

(ii) In our protocol, the user can encrypt the index i with our single-ciphertext FHE scheme and then send the ciphertext to the server. For the consideration of efficiency, we directly encrypt the index with a symmetric encryption scheme. The parameters a, b connect the partial ciphertext $c(x, y)$ and its corresponding plaintext. In particular, the parameters a, b invoke a polynomial, and the polynomial is used to encrypt the queried index; meanwhile, the polynomial ciphertext $c(x, y)$ can be directly decrypted with the parameters a, b ignoring the parameters u, v as the auxiliary information. In turn, the server outputs a function about the ith data a_i relative to the polynomial ciphertext. Then, the user decrypts the function using the parameters a, b, and he will exactly obtain the ith data a_i corresponding to the index i. During the process, the server provides some computation and storage space and is unable to acquire the information of the index i. Consequently, our single-server PIR protocol achieves the goal as desired.

(iii) Moreover, we prefer the communication complexity on the user side rather than on the server side. Hence, in Section 6, the communication complexity on the user side is much more important than the overheads on the server side. In the future, we will delve the communication-efficient single-server PIR protocol which can attain the tradeoff overheads of the communication and the computation between the user and the server.

In the following, we employ the single-ciphertext FHE scheme proposed in Section 4.1 and the Lagrange interpolating polynomial to construct our communication-efficient single-server PIR protocol. The detailed three algorithms are described as follows:

(i) Query generation phase: taking the index $i\,(0 \le i \le N-1)$ as the input, the user sends a query $\mathbf{Q}(i)$ to the DB server. The details are described in Algorithm 3.

(ii) Response generation phase: upon receiving the query $\mathbf{Q}(i)$, the DB server outputs $R(i) = g(x, y)$ to the user in Algorithm 4. Note that even if (u, v) are obtained in the query $\mathbf{Q}(i) = (n, u, v, c(x, y))$, the DB server cannot recover the index i due to not knowing the symmetric key (a, b).

(iii) Response retrieval phase: refer to Algorithm 5. Upon receiving the response $R(i) = g(x, y)$, the user retrieves the data $\mathbf{DB}[i] \equiv g(a, b) (\mathrm{mod} n)$ corresponding to the index i by using the symmetric key (a, b).

Correctness: now, we illustrate the correctness of our proposed single-server PIR protocol, namely, $\mathbf{DB}[i] = \mathrm{RR}(\mathbf{DB}, i, \mathbf{Q}(i), R(i))$, for any database $\mathbf{DB} = \{a_0, a_1, \ldots, a_{N-1}\}$ with any size N and any index $0 \le i \le N-1$.

During the response generation phase, the response $R(i) = g(x, y)$ is an evaluation of encryption of index i. Meanwhile, the response $R(i) = g(x, y)$ is N numbers of addition operations about the whole data a_l for $0 \le l \le N-1$. When decrypting the response $R(i)$ correctly, the user will obtain that

$$
\begin{aligned}
g(a,b) &\equiv \sum_{l=0}^{N-1} a_l \prod_{0 \le j \le N-1, j \ne l} \frac{c(a,b)-j}{l-j} (\mathrm{mod} n) \\
&\equiv \sum_{l=0}^{N-1} a_l \prod_{0 \le j \le N-1, j \ne l} \frac{i-j}{l-j} (\mathrm{mod} n) \\
&\quad \because c(a,b) \equiv i (\mathrm{mod} n) \\
&\equiv a_0 \cdot \frac{i-1}{0-1} \cdot \frac{i-2}{0-2} \cdots \frac{i-(N-1)}{0-(N-1)} + \\
&\quad a_1 \cdot \frac{i-0}{1-0} \cdot \frac{i-2}{1-2} \cdots \frac{i-(N-1)}{1-(N-1)} + \\
&\quad \cdots\cdots + \\
&\quad a_{N-1} \cdot \frac{i-0}{N-1-0} \cdot \frac{i-1}{N-1-1} \\
&\quad \cdots \frac{i-(N-2)}{N-1-(N-2)} (\mathrm{mod} n).
\end{aligned} \tag{6}
$$

When we assume $i = 1$ for an example, it is obvious that the above items (6) in $g(a, b)$ all equal 0 since there is an item $i-1$ in the molecule, while item (6) in $g(a, b)$ equals a_1 since the molecule is equal to the denominator. Therefore, we can conclude that once decrypting the response $R(i)$ correctly, the user will obtain that $g(a, b) \equiv a_i (\mathrm{mod} n)$ since $c(a, b) \equiv i (\mathrm{mod} n)$. As a result, the correctness of our proposed single-server PIR protocol holds, as desired.

Input: the index $i\,(0 \le i \le N-1)$.
(1) Randomly generate $\lambda/2$-bit-long primes p, q subject to gcd $(p-1,3)=1$ and gcd $(q-1,3)=1$ and compute $n=pq$.
(2) Randomly choose $a_{ij} \in \mathbb{Z}_n$ for $i, j = 0, 1, 2$ and set $f(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} a_{ij}x^i y^j$.
(3) Randomly choose $a, b \in \mathbb{Z}_n$ and compute $u \equiv a^3 \pmod{n}, v \equiv b^3 \pmod{n}$.
(4) Set $F(x,y) \equiv f(x,y) - f(a,b) \pmod{n}$.
(5) Compute $c(x,y) \equiv F(x,y) + i \pmod{n}$.//*The symmetric key* (a,b) *is kept by the user and private for the DB server.*
Output: $\mathbf{Q}(i) = (n, u, v, c(x,y))$.

Algorithm 3: The query generation algorithm (user).

Input: a database $\mathbf{B} = \{a_0, a_1, \ldots, a_{N-1}\}$ of size N and a query $\mathbf{Q}(i) = (n, u, v, c(x,y))$.
(1) Compute $g(x,y) \equiv \sum_{l=0}^{N-1} a_l \prod_{0 \le j \le N-1, j \ne l} c(x,y) - j/l - j \pmod{n, x^3 - u, y^3 - v}$.
Output: $R(i) = g(x,y)$.

Algorithm 4: Response generation algorithm (DB server).

Input: the response $R(i) = g(x,y)$ and the symmetric key (a,b).
Output: $\mathbf{DB}[i] \equiv g(a,b) \pmod{n}$.

Algorithm 5: The response retrieval algorithm (user).

Remark 2. Note that the length of each item $\mathbf{DB}[i]$, for any $0 \le i \le N-1$, in $\mathbf{DB}$ should be smaller than λ. Otherwise, what the user would obtain from the above response retrieval algorithm is not the value of $\mathbf{DB}[i]$ as $\mathbf{DB}[i]$ had been damaged by the operation of modulo n.

5. Security Analyses

In this section, we will discuss the security of our single-server PIR protocol. We particularly focus on the privacy properties, i.e., the query index should be privacy preserving, and the response is also privacy preserving in the proposed single-server PIR protocol.

(i) The query index is privacy preserving in the proposed single-server PIR protocol: our design goal is to require that the queried index i should be private, and no one, except the query user, can determine the value of i. As we know, the query index is encrypted by our single-ciphertext FHE scheme, and only the query user can obtain the index. Because the security of our single-ciphertext FHE scheme can be reduced to the 3rd RSA problem, without knowing the private key, no one can retrieve the query index. As a result, the query index can be hidden, and the privacy-preserving requirement on the query index can be achieved in the proposed single-server PIR protocol.

(ii) The response is also privacy preserving in the proposed single-server PIR protocol: since we consider there is no collusion on the DB server, the server will not forge the data in $\mathbf{DB}$. Instead, the server will follow Algorithm 4 and output correct responses. Moreover, according to the correctness in Section 4.2, it is easy to find that the response $R(i)$ is a polynomial about the encryption of $\mathbf{DB}[i]$. Automatically, the data $\mathbf{DB}[i]$ can be hidden in the response $R(i)$. No one, except the query user, can retrieve $\mathbf{DB}[i]$ by correctly decrypting the response $R(i)$. Therefore, the response is privacy preserving in the proposed single-server PIR protocol (Algorithm 5).

Now, we will present the security of our single-server PIR protocol by the simulation-based framework.

Theorem 2. *Our single-server PIR protocol is secure against the adversaries* $\mathcal{A} = \{User, Server\}$.

Proof. We will elaborate that there is a probabilistic polynomial time simulator $\mathcal{S}^{\text{Server}}$ playing the role of the DB server such that the real view and the ideal one are computationally indistinguishable for User.

The interactions between User and $\mathcal{S}^{\text{Server}}$ are defined by the following steps:

(1) Following Algorithm 3, User sends $\mathbf{Q}(i)$ of the index i to $\mathcal{S}^{\text{Server}}$
(2) $\mathcal{S}^{\text{Server}}$ sends the encryption of a_i back to User
(3) Decrypt the result from $\mathcal{S}^{\text{Server}}$ with his own secret key, and User will obtain a_i as desired

The real view for User is $(\mathbf{Q}(i), R_i, a_i)$, while the ideal view for User is $(\mathbf{Q}(i), \text{Enc}(a_i), a_i)$. Considering that R_i and

$\mathrm{Enc}(a_i)$ are indistinguishable, we can conclude that the ideal view of User is indistinguishable from the real view. Then, we can claim that User can learn nothing about the data from the database server except a_i, which implies that the single-server PIR protocol is secure for the DB server.

From the above analyses, we can see our proposed single-server PIR protocol is confidential and can protect the information of the index i and the corresponding data a_i.

Remark 3. In our security model, we consider the DB server is honest but curious. However, we cannot avoid the semimalicious DB servers. To prevent semimalicious servers from forging the data in **DB** as responses, we can add a verifiable procedure during the response generation phase. The following is a desirable attempt: we will use a hash function h to act on the data because of its one-wayness. During the response generation phase, we require the server should substitute a_l with $a_l \| h(a_l)$ in Algorithm 4 and send a correct result in the response $g(x, y)$ to the user, where .‖ represents concatenation. There is no doubt that the length of $a_l \| h(a_l)$ is smaller than n, that is, $|a_l| + |h(a_l)| < \lambda$. Then, the new response $R'(i) = g'(x, y)$ is an encryption of the data $\mathbf{DB}[i] \| h(\mathbf{DB}[i])$. Therefore, the user can verify whether the server forges the data. After decrypting the response $R(i)$ to obtain a_i and $h(a_i)$, the user can compute the hash value $h(a_i)$ due to knowing a_i. If it equals the value $h(a_i)$ the server sends, the data a_i are exactly corresponding to the index i without errors. If not, the server is dishonest. The details are omitted here.

6. Performance Evaluation

In this section, we evaluate the performance of our proposed single-server PIR protocol from two perspectives, i.e., the theoretical analyses and experimental evaluation by comparing it with two existing PIR protocols in [6, 7].

6.1. Theoretical Analyses of Our PIR Protocol on the User Side. Here, we first illustrate that our single-server PIR protocol is much more efficient and practical than the PIR protocols in [6, 7] in terms of the computational complexity, the extension ratio of the query (similar to the length ratio between the ciphertext and its underlying plaintext, denoted by $|R(i)|/|\mathbf{DB}[i]|$), and the communication overhead (denoted by $|\mathbf{Q}(i)| + |R(i)|$).

Our PIR protocol: in our proposed PIR protocol, since the query generation phase applies our single-ciphertext FHE scheme as the basic symmetric encryption scheme, from the computational complexity analysis in Section 4.1, we can see the computational complexity is $O(\lambda^2)$ for both the query generation and the response retrieval. In addition, we can find that $|\mathbf{Q}(i)| = 12\lambda, |R(i)| = 9\lambda$, and $|\mathbf{DB}[i]| = \lambda$. Hence, the extension ratio of the query is $|R(i)|/|\mathbf{DB}[i]| = 9$ in our single-server PIR protocol. The communication overhead represents the length sum of $\mathbf{Q}(i)$ and $R(i)$, i.e., $|\mathbf{Q}(i)| + |R(i)|$. Hence, the communication overhead is 21λ in our single-server PIR protocol.

Yi et al.'s PIR protocol [6]: in Yi et al.'s PIR protocol, the computational complexity depends on the modular addition operations, and thus, we consider the computational complexity is $O(1)$. Since the data corresponding to the index are one bit and $|R(i)|$ equals the length of the DGHV ciphertext [12], i.e., $|R(i)| = O(\lambda^5)$ and $|\mathbf{DB}[i]| = 1$, the extension ratio of the query is $|R(i)|/|\mathbf{DB}[i]| = O(\lambda^5)$. Finally, the communication overhead is $O(\gamma \log N)$, where γ is the size of the ciphertext and N is the size of **DB**. Again, because the DGHV scheme with the ciphertext length $O(\lambda^5)$ is utilized to construct the PIR protocol, the communication overhead is $O(\lambda^5 \log N)$.

Li et al.'s PIR protocol [7]: the computational complexity of Li et al.'s PIR protocol mainly relies on the total $\lambda^7 \log^4 N$ modular multiplications of the matrix multiplication in the HAO scheme [13]. From the computational complexity of modular multiplication mentioned in Section 4.1 and the parameters in [7], we can easily see that the valid computational complexity of Li et al.'s PIR protocol is $O(\lambda^9 \log^4 N)$. On the contrary, the data underlying the index are one bit, e.g.,$|\mathbf{DB}[i]| = 1$. The query user needs to send two ciphertexts to the DB server: one is an encryption of the query with communication overhead $O(\lambda^2 \log^2 N)$, and the other is an encryption of the key with communication overhead $(\log N + 1)^2 \cdot \lambda^4 = O(\lambda^4 \log^2 N)$, while the DB server needs to send back an encryption of $\mathbf{DB}[i]$ with communication overhead $O(\lambda^4 \log^2 N)$ to the user, i.e., $|Q(i)| = |R(i)| = O(\lambda^4 \log^2 N)$. As a result, both the extension ratio of the query and the communication overhead in [7] are $O(\lambda^4 \log^2 N)$.

Table 3 summarizes the differences among the above three PIR protocols, where the second column "Batching" captures whether the PIR protocol can directly encrypt the index from $\mathbb{Z}_n$. If the PIR protocol can, we output "Yes" and "No," otherwise, and the symbol λ is the security parameter and N is the database size. It is obvious that our proposed single-server PIR protocol, which has access to the database **DB** composed of items from $\mathbb{Z}_n$, can directly encrypt the index from $\mathbb{Z}_n$ and perform the processing batch, while the PIR protocols in [6, 7] cannot. This fact makes our single-server PIR protocol more practical. In addition, from the table, we can see, in terms of the communication overhead, our single-server PIR protocol is far superior to [6, 7] since ours is independent on the database size N.

When setting the security parameter $\lambda = 2048$ in our PIR protocol and $\lambda = 128$ in PIR protocols [6, 7] for achieving certain security level, Figure 2 compares the communication overheads of the three PIR protocols varying with N from 2^1 to 2^{20}. From the figure, we can see that our proposed single-server PIR protocol is much more efficient, especially for a larger N. Furthermore, no one can deny that when N is considered in the range $[2^1, 2^{20}]$, the communication overheads of the PIR protocols in Yi et al. [6] and Li et al. [7] are largely subject to the security parameter λ in comparison to the database size N, and the communication overhead $O(\lambda^4 \log^2 N)$ in Li et al.'s protocol is better than $O(\lambda^5 \log N)$ in Yi et al.'s protocol from Figure 2. To the best of our knowledge, for a fixed security parameter λ, our proposed protocol is the first single-server PIR protocol, which can achieve $O(1)$ communication efficiency.

Table 3: The theoretical performance analyses of PIR protocols.

Schemes	Batching	CC[a]	ERQ[b]	CO[c]/bits
Yi et al. [6]	No	$O(1)$	$O(\lambda^5)$	$O(\lambda^5 \log N)$
Li et al. [7]	No	$O(\lambda^9 \log^4 N)$	$O(\lambda^4 \log^2 N)$	$O(\lambda^4 \log^2 N)$
Ours	Yes	$O(\lambda^2)$	9	21λ

CC[a]: CC represents a valid computational complexity for the user. ERQ[b]: we use ERQ to denote the extension ratio of the query, i.e., $|R(i)|/|\mathbf{DB}[i]|$. CO[c]: CO of unit bit represents the communication overhead, i.e., $|Q(i)| + |R(i)|$.

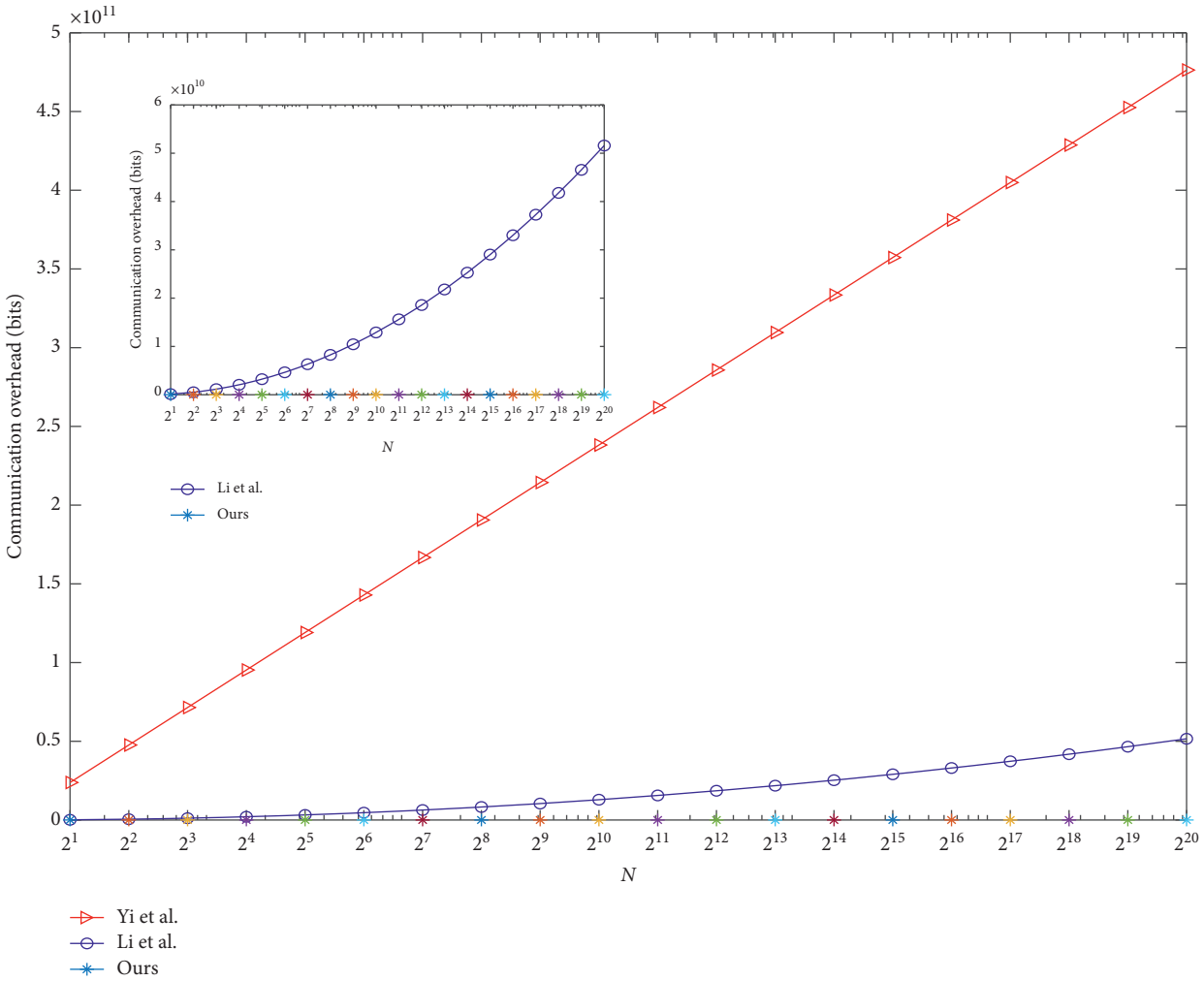

Figure 2: Communication cost comparisons varying with N from 2^1 to 2^{20} when setting the security parameter $\lambda = 2048$ in our PIR protocol and $\lambda = 128$ in PIR protocols [6, 7] for achieving certain security level.

6.2. Theoretical Analyses of Our PIR Protocol on the Server Side. Although we prefer the communication for the user than the computation complexity on the server to evaluate the efficiency of our single-server PIR protocol, the theoretical analysis on the server side is necessary to be illustrated in this section. In brief, we will present the computation burden on the server compared with the PIR protocols in [6, 7].

Our PIR protocol: the server mainly performs operations upon the special bivariate polynomials, i.e., the degree of either variable x (or y) is no more than 2. Specifically, N number of additions upon the bivariate polynomials for the server are enough, where N is the number of databases. Meanwhile, every bivariate polynomial also includes operations of polynomials modulo $n, x^3 - u$, and $y^3 - v$. And N number of bivariate polynomials can be performed in parallel or in a preprocessing way. Quantitatively speaking, the computational complexity is near to $O(N \cdot \lambda^2)$, where $\lambda \geq 2048$ bits.

Yi et al.'s PIR protocol [6]: Yi et al. encrypted the index with binary strings of length $l = \lfloor \log N \rfloor + 1$. Every bit is protected with an FHE scheme called DGHV10 [12]. During the response generation in the PIR protocol, the server mainly computes $2l$ number of modulus additions and $l - 1$

number of modulus multiplications upon the integers of length $O(\lambda^5)$. In addition, the server also provides l number of ciphertexts. In a nutshell, the computational burden of the server is $O(\log N \cdot \lambda^{10})$.

Li et al.'s PIR protocol [7]: after receiving the ciphertexts of queried index i and the secret key, the server performs a bootstrapping operation, i.e., homomorphically evaluate the decryption circuits, which is a very expensive process and occupies numerous overhead of computations for the server. The best result of bootstrapping at present is not exceeding 10 ms [27] when homomorphically implementing a single gate. It remains to be far from being practical to homomorphically evaluating a computing circuit. We will not describe the computation complexity of the bootstrapping but claim that the decryption circuit is of depth almost $O(\log \lambda)$ [20].

Totally speaking, the computational burden of our single-server PIR protocol is relatively less than [6, 7]. However, the experimental performance of the server will not be analysed in the following Section 6.3 since we regard the server powerful and can provide unrestricted computations. Furthermore, the computation burden of the server in our single-server PIR can be relaxed in parallel or in a preprocessing way.

6.3. Experimental Evaluations of Our PIR Protocol. In this section, we further present some experimental evaluations of our PIR protocol in comparison with the PIR protocols in [6, 7]. It is obvious to see that there are two common factors for the PIR protocols in [6, 7], i.e., the queried index is resolved into its binary presentation and the data in the database **DB** only consist of one bit 0 or 1, while in our PIR protocol, we can directly encrypt the index, not in its binary representation, and the data in **DB** belonging to $\mathbb{Z}_n$ are more practical. The details of experimental settings are as follows.

Our PIR protocol: we implement our proposed protocol on a personal computer by utilizing the NTL [28] and the C++ language. The environment is listed as follows:

(i) CPU: Intel(R) Core(TM) i3-7100 3.90 GHz

(ii) RAM: 4.00 GB

(iii) OS: Windows 10, 64 bits

The length of the modulus of n in our experiment includes 2048 bits, 2560 bits, and 3072 bits. The size N of **DB** varies from 800, 1000, to 1200. Although the number of items in the database N seems a little small, the whole space of the database is not small at all. Considering that the response generation (RG) phase is performed by the DB server and the query generation (QG) phase and response retrieval (RR) phase are run at the user side, we test 100 instances on $\mathbb{Z}_n$ for every phase. The average results are given in Tables 4–6.

The first column called "$|n|$" represents the length of RSA modulus, and the data in **DB** are from $\mathbb{Z}_n$. The second column means the number of our tested instances. We use the time of the query generation phase, response retrieval phase, and response generation phase to illustrate the performance of our single-server PIR protocol.

From Tables 4 to 6, it is easy to see that the size N of **DB** has a little bit effect on the user side in our single-server PIR protocol, which can almost be ignored. We also see that the time in the query generation phase and response retrieval phase increases a little with the modulus growing under the same situations. On the contrary, the time cost on the DB server side largely depends on the size N. When the database size N is fixed, the DB server takes more time with n increasing. Similarly, when the modulus n is fixed, the DB server also takes more time with N increasing. In brief, it shows that our single-server PIR protocol is efficient. For example, even when the modulus is $n = 3072$ bits, the query generation phase only costs 5.4 μs, and the response retrieval phase costs 2.82 ms at most when all data in the database **DB** are drawn from $\mathbb{Z}_n$.

The effects of database size N on the user and the DB server are readily comprehensible. Theoretically speaking, there are just modular multiplications and some negligible modular additions for the user, all of which are irrelevant to the size N during the query generation phase, let alone the response retrieval phase. On the contrary, the response generation phase completely relies on all the data in database **DB**. Hence, the size N is the main factor for the time cost at the DB server side. Nevertheless, the server can perform parallel computations to reduce the computational complexity from $O(N^2 \log^2 n)$ to $O(N \log N \log^2 n)$. Furthermore, when a powerful DB server is employed, the time costs at the DB server side should be reduced greatly.

Yi et al.'s PIR protocol [6]: Yi et al. [6] experimented on a PBR protocol (an extension of a PIR protocol) with 10,000 blocks instead of a PIR protocol. The query generation phase costs 10 μs when the modulus is of 882 bits. The overhead is obviously larger than our single-server PIR protocol. In addition, Yi et al.'s scheme did not discuss the time cost of the response retrieval phase. On this basis, our single-server PIR protocol has obvious advantages over [6]. Moreover, their PBR protocol cannot encrypt the index from $\mathbb{Z}_n$, let alone within a few milliseconds.

Li et al.'s PIR protocol [7]: Li et al. [7] proposed a PIR protocol based on the lattice assumption. However, they did not use simulation to evaluate their PIR protocol. Nevertheless, we can claim that our single-server PIR protocol is more efficient than [7] based on the aforementioned theoretic analyses of computational complexity, the extension ratio of the query, and the communication overhead. In addition, the performance of Li et al.'s PIR protocol [7] relies on the efficiency of the bootstrapping and the size of the secret key. The size of the secret key in [7] is $O(\lambda \log N)$, and by now, the best result for bootstrapping does not exceed 10 ms to evaluate a single gate in [27], which is impractical.

To sum up, our experimental evaluation further demonstrates that our single-server PIR protocol is more efficient and practical.

7. Related Work

In this section, we will briefly review some FHE schemes and some other existing single-server PIR protocols, which are closely related to our proposal.

Table 4: The computational costs of our PIR protocol under $N = 800$.

$\|n\|$	Instances	QG phase	RR phase	RG phase
2048 bits	100	3.7 μs	1.36 ms	1.65 h
2560 bits	100	4.6 μs	2.00 ms	2.41 h
3072 bits	100	5.2 μs	2.82 ms	3.16 h

Table 5: The computational costs of our PIR protocol under $N = 1000$.

$\|n\|$	Instances	QG phase	RR phase	RG phase
2048 bits	100	3.7 μs	1.34 ms	2.41 h
2560 bits	100	4.5 μs	2.01 ms	3.89 h
3072 bits	100	5.4 μs	2.80 ms	4.98 h

Table 6: The computational costs of our PIR protocol under $N = 1200$.

$\|n\|$	Instances	QG phase	RR phase	RG phase
2048 bits	100	3.5 μs	1.38 ms	3.55 h
2560 bits	100	4.5 μs	2.04 ms	5.45 h
3072 bits	100	5.3 μs	2.80 ms	7.09 h

Fully homomorphic encryption: FHE enables meaningful process over encrypted data without access to the original plaintext data. In the past years, many generic FHE constructions have been proposed [10, 12, 13, 18–20]. For example, the first generation is represented by the DGHV FHE scheme [12], which serves as a vital tool in building a PIR protocol in [6]. However, most of them turn out to be impractical. The main reason is that the noises are added to the ciphertexts for the consideration of the security. Later, a new class of FHE schemes without noises have naturally been exploited to avoid complicated noise management [24–26, 29]. For example, Nuida [24] declared a beautiful public key FHE frame without noises, employing a commutator and an encoding scheme over two noncommutative groups. The security is based on an assumption that judging whether an element is in the kernel is difficult, which is not standard. Yagisawa [25, 26] proposed noiseless FHE schemes with the underlying octonion ring over the finite field, which are immune from the Grobner basis attacks. Nevertheless, it remains an open problem to prove strictly secure and feasible in the framework of provable security. In this work, motivated by the noiseless FHE schemes, we define a special kind of algebraic structure called truncated polynomial rings to construct a single-ciphertext FHE scheme. Our proposed scheme is noiseless, and hence, it inherently supports fully homomorphic computations on any univariate polynomials, such as the single-server PIR protocols. In addition, there is a security reduction between the one-wayness of our single-ciphertext FHE scheme and the 3rd RSA problem we define. Compared with the FHE schemes in [6, 7], our single-ciphertext FHE scheme is noiseless and of the smallest ciphertext size, which offer enormous convenience for the single-server PIR protocols.

Single-server PIR protocols: a single-server PIR protocol allows a user to retrieve the i-th data from a database server without revealing the index i. The past years have witnessed the development of the single-server PIR protocols, especially in communication cost [30–33]. For example, Cachin et al. [31] proposed a PIR protocol based on the φ-hiding assumption with communication complexity $O(\log^4 N)$. And the Damgard–Jurik scheme [34] was utilized by Lipmaa [32] to construct a PIR protocol, which achieved $O(\log^2 N)$ communication complexity. However, most of them are inefficient due to depending on the database size N, especially when N is million or even larger magnitude in real life. Hence, it will be a great work to construct a single-server PIR protocol with communication overhead $O(1)$, which is irrelevant to the database size N. In this work, we devote ourselves to designing a single-server PIR protocol with communication efficiency $O(1)$. First, on the basis of the single-server PIR protocols in [6, 7], we tend to resort to the FHE schemes since the FHE schemes not only keep privacy preserving but also support direct computations on encrypted data. Second, the Lagrange interpolating polynomial is a suitable tool for the database $\mathbf{DB} = \{(i, \mathbf{DB}[i]) | 0 \le i \le N - 1\}$ owing to its property. Hence, an FHE scheme and the Lagrange interpolating polynomial technique are used to construct our single-server PIR protocol. Meanwhile, theoretical analyses and experimental evaluations are performed to demonstrate that our single-server PIR protocol is efficient, which is the first one of communication overhead $O(1)$.

8. Conclusions

In this paper, we have proposed a new communication-efficient PIR protocol by using homomorphically computing univariate polynomials. Specifically, we first propose a new cryptographic primitive called single-ciphertext FHE and instantiate the special kind of FHE supporting evaluations of a single ciphertext. Then, we illustrate how the single-ciphertext FHE scheme works in our single-server PIR protocol. Theoretical analyses and experimental evaluations

are both conducted to demonstrate that it is more efficient and practical to apply our single-ciphertext FHE scheme to the PIR protocol. To the best of our knowledge, our proposed protocol is the first PIR protocol, which can achieve $O(1)$ communication efficiency on the user side, irrelevant to the database size N. In future work, we will study other FHE techniques to exploit more efficient PIR protocols, which can achieve the tradeoff overheads of the communication and the computation between the user and the server.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant no. 2017YFB0802000), the National Natural Science Foundation of China (Grant nos. U19B2021 and 61972457), the National Cryptography Development Fund (Grant no. MMJJ20180111), and Key Research and Development Program of Shaanxi (Grant no. 2020ZDLGY08-04).

References

[1] R. Ostrovsky, "A survey of single-database private information retrieval: techniques and applications," in *Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography*, pp. 393–411, Beijing, China, April 2007.

[2] E. Skeith and R. Ostrovsky, "Replication is NOT needed: SINGLE database, computationally-private information retrieval," in *Proceedings of the 38th Annual Symposium On Foundations Of Computer Science*, pp. 364–373, FOCS '97, Miami Beach, FL, USA, October 1997.

[3] E. Skeith and R. Ostrovsky, "One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval," in *Proceedings of the Advances In Cryptology - EUROCRYPT 2000, International Conference On the Theory And Application Of Cryptographic Techniques*, pp. 104–121, Bruges, Belgium, May 2000.

[4] C. Gentry and Z. Ramzan, "Single-database private information retrieval with constant communication rate," *Automata, Languages and Programming*, pp. 803–815, 2005.

[5] C. A. Melchor, J. Barrier, L. Fousse, and M. Killijian, "Private information retrieval for everyone," *PoPETs*, vol. 2, pp. 155–174, 2016.

[6] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1125–1134, 2013.

[7] Z. Li, C. Ma, D. Wang, and G. Du, "Toward single-server private information retrieval protocol via learning with errors," *Journal of Information Security and Applications*, vol. 34, pp. 280–284, 2017.

[8] C. Gentry, "Fully homomorphic encryption scheme" PhD Thesis, Stanford University, Kunnamangalam, India, 2009.

[9] C. Gentry, *Fully Homomorphic Encryption Using Ideal Lattices*, pp. 169–178, ACM, New York, NY, USA, 2009.

[10] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 97–106, Palm Springs, CA, USA, October 2011.

[11] Y. Doröz, B. Sunar, and G. Hammouri, "Bandwidth efficient PIR from NTRU," *Financial Cryptography and Data Security*, vol. 8438, pp. 195–207, 2014.

[12] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of the Advances In Cryptology - EUROCRYPT 2010, 29th Annual International Conference On the Theory And Applications Of Cryptographic Techniques*, pp. 24–43, Monaco, Europe, May 2010.

[13] R. Hiromasa, M. Abe, and T. Okamoto, "Packing messages and optimizing bootstrapping in GSW-FHE," in *Proceedings of the 18th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 699–715, Gaithersburg, MD, USA, March 2015.

[14] R. McEliece, "Finite fields for computer scientists and engineer," in *Kluwer International Series In Engineering And Computer Science,* Springer, New York, NY, USA, 1989.

[15] B. Wang, H. Lei, and Y. Hu, "D-NTRU: more efficient and average-case IND-CPA secure NTRU variant," *Information Sciences*, vol. 438, pp. 15–31, 2018.

[16] D. R. L. Brown, "Breaking RSA may be as difficult as factoring," *Journal of Cryptology*, vol. 29, no. 1, pp. 220–241, 2016.

[17] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 309–325, Santa Barbara, CA, USA, 2012.

[19] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," in *Proceedings of the Advances In Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference*, pp. 75–92, Santa Barbara, CA, USA, August 2013.

[20] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Proceedings of the Advances In Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference*, pp. 868–886, Santa Barbara, CA, USA, August 2012.

[21] J. Loftus, A. May, N. P. Smart, and F. Vercauteren, "on cca-secure somewhat homomorphic encryption," in *Proceedings of the Selected Areas In Cryptography - 18th International Workshop*, pp. 55–72, Toronto, Canada, August 2011.

[22] R. Canetti, S. Raghuraman, S. Richelson, and V. Vaikuntanathan, "Chosen-ciphertext secure fully homomorphic encryption," in *Proceedings of the Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pp. 213–240, Amsterdam, The Netherlands, March 2017.

[23] B. Yang and G. Xiao, *Modern Cryptography*, Tsinghua University Press, Beijing, China, 2015.

[24] K. Nuida, "A simple framework for noise-free construction of fully homomorphic encryption from a special class of non-commutative groups," *IACR Cryptology ePrint Archive*, vol. 97, 2014.

[25] M. Yagisawa, "Improved fully homomorphic public-key encryption with small ciphertext size," *IACR Cryptology ePrint Archive*, vol. 232, 2018.
[26] M. Yagisawa, "Fully homomorphic encryption on octonion ring," *IACR Cryptology ePrint Archive*, vol. 733, 2015.
[27] T. Zhou, X. Yang, L. Liu, W. Zhang, and N. Li, "Faster bootstrapping with multiple addends," *IEEE Access*, vol. 6, 2018.
[28] V. Shoup, "The number theory library (ntl)," 2017, http://www.shoup.net.
[29] K. Nuida, "Candidate constructions of fully homomorphic encryption on finite simple groups without ciphertext noise," *IACR Cryptology ePrint Archive*, vol. 97, 2015.
[30] J. P. Stern, "A new efficient all-or-nothing disclosure of secrets protocol," in *Advances In Cryptology*, pp. 357–371, Beijing, China, 1998.
[31] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," in *Proceedings of the Advances In Cryptology - EUROCRYPT '99, International Conference On the Theory And Application Of Cryptographic Techniques*, pp. 402–414, Prague, Czech Republic, May 1999.
[32] H. Lipmaa, "First CPIR protocol with data-dependent computation," in *Proceedings of the in Information, Security and Cryptology - ICISC 2009, 12th International Conference*, pp. 193–210, Seoul, Korea, 2009.
[33] Y. Chang, "Single database private information retrieval with logarithmic communication," in *Proceedings of the In Information Security And Privacy: 9th Australasian Conference, ACISP 2004*, pp. 50–61, Sydney, Australia, July 2004.
[34] J. M. Damgard, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptosystems*, Seoul, Korea, 2001.