Privacy-preserving Equality Test towards Big Data

Tushar Kanti Saha and Takeshi Koshiba

[1] Division of Mathematics, Electronics, and Informatics,
Graduate School of Science and Engineering, Saitama University,
Saitama, Japan.
saha.t.k.512@ms.saitama-u.ac.jp
[2] Faculty of Education and Integrated Arts and Sciences,
Waseda University, Tokyo, Japan.
tkoshiba@waseda.jp

Abstract. In this paper, we review the problem of private batch equality test (PriBET) that was proposed by Saha and Koshiba (3rd APWConCSE 2016). They described this problem to find the equality of an integer within a set of integers between two parties who do not want to reveal their information if they do not equal. For this purpose, they proposed the PriBET protocol along with a packing method using the binary encoding of data. Their protocol was secured by using ring-LWE based somewhat homomorphic encryption (SwHE) in the semi-honest model. But this protocol is not fast enough to address the big data problem in some practical applications. To solve this problem, we propose a fixed length base-N encoding based PriBET protocol using SwHE in the same semi-honest model. Here we did our experiments for finding the equalities of $8 \sim 64$-bit integers. Furthermore, our experiments show that our protocol is able to evaluate more than one million (resp. 862 thousand) of equality comparisons per minute for 8-bit (resp. 16-bit) integers with an encoding size of base 256 (resp. 65536). Besides, our protocol works more than $8 \sim 20$ in magnitude than that of Saha and Koshiba.

Keywords: Private batch equality test, Base-N encoding, Homomorphic encryption, Packing method

1 Introduction

Since the establishment of Internet technology, data are increasing day by day in an expeditious speed. Also, users are extensively using the computers, laptops, tabs along with the Internet. Besides, smart-phones and Wi-Fi devices are helping us to use the Internet even when we are mobile. So data are becoming very big which are called big data. Managing and analyzing big data is a big challenge for its user where new tools and techniques are indispensable. In addition, local storage is not enough for the users to store their data. Recently, banks, insurance companies, hospitals, research institutes, public service centers, and so on have come forward to store their customers' data electronically and

maintain their databases. Besides, cloud computing has established itself as a reliable service provider by giving remote on-line storage to its users at an affordable price. Moreover, organizations are interested in outsourcing their data to the cloud servers to access them anytime from versatile locations. Also, these organizations want to provide security to their data. Here encryption is one of the procedures to provide data security. In addition to this, the organizations want to do their required operations on the encrypted data which is hard to perform before decryption. So homomorphic encryption [11] is a solution for them which allows meaningful computations like additions and multiplications on the encrypted data.

On the contrary, many research works in secure computation (for example, [1,6,12,14,16]) have already been conducted using ring-LWE based homomorphic encryption after the breakthrough work of fully homomorphic encryption (FHE) by Gentry [5] in 2009. However, fully homomorphic encryption allows any number of additions and multiplications on the encrypted data which makes it slower for practical use [12]. In 2011, Brakerski and Vaikuntanathan [1] proposed one more somewhat homomorphic encryption (SwHE) using the concept of ring-LWE which works little faster due to supporting many additions and few multiplications. Thereafter it had been used in the literature [6,12,16] which showed the practicality of SwHE. Now some organizations are needed to share their data with other organizations for different purpose like private data mining, machine learning, searching, information retrieval, and so on where private equality test is important. But data protection regulation does not allow these organizations to share their data with one another or even with the cloud. Here PET protocol using SwHE can be used by these organizations which was proposed by Saha et al [13]. They also proposed private batch equality test (PriBET) protocol for finding the equality of an integer with a set of integers using the batch technique. But their achieved performance is not good enough to handle a large dataset of millions of integers. Furthermore, most of the organizations deal with a large dataset which varies from several gigabytes to terabyte where a faster performance is the prerequisite to search something within their datasets.

1.1 Prior Works

The idea of secure computation was first introduced by Yao in 1980 [19]. In this section, we review some papers on secure computation for the PriBET protocols. To date, very few protocols have been proposed for the PriBET protocol. In 2016, Couteau [3] proposed the PriBET protocol in the semi-honest model that required 7 rounds communication between two-party to compare data size of $16 \sim 128$-bit. But they did not show any implementation. To understand the practicality of the protocol, some implementations are indispensable. Recently, Saha et al. [13] proposed private batch equality test (PriBET) protocol which shows some practicality of the protocol. Their protocol was able to do about 140 thousand comparisons per minute for 8-bit data only. The performance decreases if the data size is increased more. To date, existing equality protocols are not

enough to handle a large database for many equality queries. So new method is desirable which can be a big step towards big data processing.

1.2 Typical Applications

In 2016, Saha et al. [13] showed some application areas of the PriBET protocols including on-line auction, genomic computation, machine learning, data mining, and database query processing where batch equality protocol is required. Besides, PriBET protocol is useful for private information retrieval [20]. Over and above that, our protocol may be useful in some practical applications like credit card number verification, criminal database searching using social security number, insurance number verification, and so on.

1.3 Motivation

In 2016, Saha et al. [13] proposed private batch equality test (PriBET) protocol for comparing integers of $8 \sim 32$ bits with a new packing method using ring-LWE based SwHE. Here they achieved an acceptable performance for handling private equality computation of a small dataset like several megabytes to a gigabyte in size. But this performance is not enough for addressing big data which refers several gigabytes to a terabyte. Saha et al. performed equality computations on binary vectors which required a large lattice dimension to process a kilobyte of data. For example, processing 1 MB data (2^{23} bits) requires a lattice dimension of 2^{23} where they consider a lattice dimension of 2^{12} to get more efficiency. If we engage the protocol with a lattice dimension of 4096, then it would require about 351 seconds to process 1 MB of data for equality which in turns requires about 100 hours to process 1 GB of data with a single machine. This performance could be further improved by using some parallel processing techniques or engaging many computers in a distributed computing environment. But this performance is not enough for handling a large database. Furthermore, the processing speed of this protocol is mostly dependent on lattice dimension where they used binary encoding to find the equality. At this point, this protocol can be improved further if we would have a method to reduce the lattice dimension.

1.4 Our Contribution

In this paper, we propose a modified PriBET protocol for finding the equalities of some integers with $8 \sim 32$ bits along with an efficient data encoding technique to reduce the lattice dimension as well as processing time. Theoretically, we achieve a reduction of the lattice dimension by a factor $\log_2(N)$ than Saha et al.'s protocol [13] due to using an efficient encoding technique where N represents the encoding size. Also, our practical experiments show that our protocol works more than 8 times faster than Saha et al.'s protocol [13]. In addition, we have been able to process more than 1 million comparisons per minute for the 8-bit integers and 862 thousand comparisons per minute for the 16-bit integers with

the encoding size of 2^8 and 2^{16} respectively which could be helpful to process a big database. Besides, in the PriBET protocol of Saha et al., Bob needed a decryption help from Alice to check a part of his result for some decision making after his computation. In this case, he leaked some additional information to Alice due to sending whole encrypted polynomial to Alice. Here we minimize the information leakage problem through random masks that occurred in [13]. Besides, we show the practical upper bound of the encoding size of our protocol using ring-LWE SwHE.

Remark 1. Here we compare the performances of the both methods which are implemented in a single PC environment configured with one 3.6 GHz Intel core-i7 CPU and an 8 GB RAM in Linux environment.

2 Data Encoding Technique

In this section, we review the Saha et al.'s data encoding technique [13] and discuss our base-N data encoding technique. The base-N encoding was also used by Yasuda et al. [18] to pack a large integer vector of 16 $\sim$ 32-bit for an efficient statistical analysis. But we use fixed length base-N encoding where most significant digits (MSDs) are filled up by '0' if it is empty in the encoded number. The reasons for choosing the base-N encoding are described below.

Notations. In this paper, $\mathbb{Z}$ denotes the ring of integers. In addition, R denotes a ring of integer of the form $\mathbb{Z}[x]/f(x)$ where $f(x)$ denotes a cyclotomic polynomial of degree n as $f(x) = x^n+1$ with a lattice dimension n. For a prime number q, the ring of integer modulo q is denoted by $\mathbb{Z}_q$. The ciphertext space is denoted by the ring $R_q = R/qR = \mathbb{Z}_q[x]/f(x)$. For an integer $t < q$, the message space is defined as $R_t = \mathbb{Z}_t[x]/f(x)$. Besides, $\mathbb{Z}[x]$ denotes the ring of polynomials over integers. For a vector $\mathbf{A} = (a_0, a_1, \ldots, a_{n-1})$, the maximum norm of $\|\mathbf{A}\|_\infty$ is defined as $\max|a_i|$. Let $\langle \mathbf{A}, \mathbf{B} \rangle$ denote the inner product between two vectors $\mathbf{A}$ and $\mathbf{B}$. Moreover, the function $\mathrm{Enc}(m, pk) = ct(m)$ defines the encryption of message m using the public key pk to produce the ciphertext ct. Also, l and l_N denote the length of an integer in binary and base-N encoding respectively. Besides, γ and k represent the block size and the total number of records respectively where a block is a collection of records.

In 2016, Saha et al. [13] used binary encoding technique over the alphabets $\{0,1\}^l$ for their private batch equality protocol (PriBET) where they got an acceptable performance for practical use for a batch data size k. But the protocol is not fast enough for big data processing. From the Table 1 in [13], we observed that the speed of the PET protocol mostly depends on the lattice dimension. In the secure computation, they required three homomorphic multiplications over a polynomial ring R_q. In the ring-LWE lattice-based homomorphic encryption scheme, a homomorphic multiplication requires doing a large polynomial multiplication over a lattice dimension of at least $n = 2048$ to achieve a security level over 128-bit [13]. Saha et al. showed the private batch equality tests for $8 \sim 32$-bit integers within the lattice dimension of $2048 \sim 4096$. Here we observed that

the lattice dimension is increasing with the increase of data size l and batch size k. For example, to process a 16-bit integers comparison, executing PriBET on the block size of 128 required a lattice dimension of $n = l \cdot k = 16 \cdot 128 = 2048$. On the other hand, it required a lattice dimension of $n = l \cdot k = 16 \cdot 256 = 4096$ for a batch size of 256. At this point, minimizing the lattice dimension is indispensable to minimize the computation time of the PriBET protocol. If we able to use an encoding technique other than binary, then we can reduce the lattice dimension. Moreover, we call the used binary encoding in [13] as base-2 encoding where alphabet set is $\{0, (2-1)\}^l = \{0, 1\}^l$. In addition, a binary encoding is using alphabets '0' and '1' to convert any decimal number z which requires $l = \lfloor \log_N(z) \rfloor + 1$ digits where $N = 2$ in this case. Saha et al. used an l-bit binary conversion algorithm for any integer of $l = 8 \sim 32$-bit. That means, if the required number of binary digit to represent any integer is less than l then rest of the MSBs are filled up by zeros. If we can do the encoding over a large alphabet set, then we can reduce the lattice dimension.

Data: z, N and l;
Result: base-N number;
Input z, N and l;
Set $l_N = l/\log_2(N)$;
$z_{baseN} = \text{baseNConvert}(z, N, l_N)$;
Output z_{baseN};
Procedure: baseNConvert(z, N, l_N)
forall the $i \in l_N$ **do**
  set $digit[i] = 0$;
end
set $ind = 0$;
while $(z \neq 0)$ **do**
  $r = z \mod N$;
  $z = z/N$;
  $digit[ind] = r$;
  $ind++$;
end
return $digit$;

Algorithm 1: Fixed length base-N encoding algorithm

Now we show the mathematical structure how the base-N encoding can work faster than base-2 encoding where $N > 2$. Here we are dealing with lattice-based cryptography where the working speed mostly depends on the lattice dimension n. To process k data of size l-bit using binary encoding, Saha et al. required a lattice dimension n' of

$$n' = k \cdot l\,. \tag{1}$$

On the contrary, to represent an l-bit integer in base-N encoding, we need a vector of size of

$$l_N = \lceil l/\log_2(N) \rceil\,. \tag{2}$$

So using base-N encoding, the new lattice dimension n'' can be determined from batch size k and base-N vector size l_N as

$$n'' = k \cdot l_N \,. \tag{3}$$

Now by dividing Eq. (1) by Eq. (3) and with the help of Eq. (2), the relation between new lattice dimension n'' and Saha et al.'s lattice dimension n' can be obtained as

$$n'/n'' = l/l_N = \log_2(N) \,. \tag{4}$$

Here we get the lattice dimension reduction rate as a factor of $\log_2(N)$. So we use fixed length base-N encoding for any positive integer in $\mathbb{Z}_t$ using the alphabet set $\{0, 1, 2, \ldots, N-1\}^{l_N}$. Now we slightly modify the basic base-N conversion algorithm to make it fixed length to get our algorithm for base-N encoding as shown in Algorithm 1. From this algorithm, we get the fixed length base-N encoding of an integer in $\mathbb{Z}_t$ by putting '0' in the MSDs if the length of the base-N encoded vector is less than l_N. From the above discussion, it is clear that our encoding scheme also reduces the size of any integer vector from its binary representation with the ratio of $l : l/\log_2(N)$. In addition, we believe that our encoding technique can be used in other contexts where the length reduction of binary encoded vectors and batch computation of the many Euclidean distances are indispensable.

3 Our Protocol

Saha et al. [13] proposed the PriBET protocol using SwHE with binary encoding in the semi-honest model. Here we propose another protocol called base-N PriBET protocol using base-N encoding described in Section 2 to increase its efficiency that is described as follows.

Consider a bank (Alice) wants to sanction some home loans to its customers who have good credit score and are paying their taxes regularly. Suppose that a customer of the bank now applies for a new home loan who has good credit score with the required information along with his tax certificate. Furthermore, the bank (Alice) needs to verify the customer's tax identification number (TIN) to check his status while sanctioning a new loan. On the contrary, the national tax department (Charlie) is maintaining the database of all its taxpayers. Now neither the bank can disclose its customer's information to the national tax department nor the national tax department can disclose it's all customers' information to the bank. Here a third party like Bob in the cloud can solve this problem and does the verification on behalf of them without knowing the actual tax number from both parties. This is a problem of verifying the equality of a large integer with a large set of integers.

From the above scenario, let Alice has an l-bit integer which can be represented by a base-N vector as $\alpha = (\alpha_1, \ldots, \alpha_{l_N})$ by using Algorithm 1. In addition, the national tax department has k integers with the same size that can be represented by the base-N integer vectors as $\beta_\lambda = (\beta_{\lambda,1}, \ldots, \beta_{\lambda,l_N})$ by

applying same algorithm where $1 \leq \lambda \leq k$. As we know from Saha et al. [13], the Hamming distance between two l-bit integers can find out whether they are equal or not. But the Hamming distance works for only binary vectors. Therefore, we use the concept that two integers are equal when the square Euclidean distance (SED) between their vectors using fixed length base-N encoding will be 0. Now the equality test for batch comparisons can be realized by the following equation as

$$E_\lambda = \sum_{i=1}^{l_N} (\alpha_i - \beta_{\lambda,i})^2 \tag{5}$$

where $1 \leq \lambda \leq k$. Moreover, E_λ represents the square Euclidean distances (SEDs) between two base-N vectors α and β_λ. Moreover, if E_λ in Eq. (5) is 0 for some positions of λ then we can say that $\alpha = \beta_\lambda$; otherwise $\alpha \neq \beta_\lambda$. In this way, Alice securely verifies her customer with the help of Bob. Now we describe our protocol by the following steps.

Inputs: $\alpha = (\alpha_1, \ldots, \alpha_{l_N})$ and $\{\beta_1, \beta_2, ..., \beta_k\}$, where $\beta_\lambda = (\beta_{\lambda,1}, \ldots, \beta_{\lambda,l_N})$ for each λ in $\{1, 2, ..., k\}$.
Output: $\exists\lambda[\alpha = \beta_\lambda]$ or $\forall\lambda[\alpha \neq \beta_\lambda]$
Base-N PriBET protocol:

1. By using SwHE, Alice constructs the public key and private key by herself and sends the public key to Charlie through a secure channel.
2. Then Alice encrypts the customer's TIN $\alpha = (\alpha_1, \ldots, \alpha_{l_N})$ using her public key and sends it to Bob.
3. The national tax department (Charlie) also uses the public key given by Alice to encrypt k TINs $\beta_\lambda = (\beta_{\lambda,1}, \ldots, \beta_{\lambda,l_N})$ where $1 \leq \lambda \leq k$ and sends the value to Bob.
4. Bob does the computation in Eq. (5) on the encrypted TINs and sends the encrypted result $ct(E_\lambda)$ to Alice to verify whether at least one of E_λ is equal to 0.
5. For $1 \leq \lambda \leq k$, Alice decrypts $ct(E_\lambda)$ using her secret key and checks each value E_λ.
6. If Alice finds at least one of the $E_\lambda = 0$ then she decides the match; otherwise, she decides no match.

Remark 2. Here our protocol provides the passive security under the assumption that Bob is semi-honest. In other words, Bob follows the protocol but tries to learn information from the protocol. Furthermore, we use the same ring-LWE based SwHE scheme used in Saha et al [13] for the security of our protocol. In this section, we skip its review due to page limitation. Besides, the security assumption of the scheme is based on the ring-LWE assumption which is reducible to the worst-case hardness of problems on ideal lattices that is believed to be secure against the quantum computer as mentioned by Lyubashevsky et al. [9].

Remark 3. The goal of our protocol is to find $\alpha = \beta_\lambda$ or $\alpha \neq \beta_\lambda$ for some $1 \leq \lambda \leq k$. Now we also think about the security of index λ. In our base-N PriBET protocol, Alice can know such index λ. Since such index is not actual index that exists in the databases of the national tax department, so leakage of such information to Alice does not harm the security of our protocol.

4 Packing Method

Packing method is the process of representing many data in a single polynomial. We know from some existing literature [6,12,13,16] that packing method makes many secure computations using ring-LWE SwHE more practical. Recently, Saha et al. [13] used binary vectors to address their packing. Here we consider the same packing with fixed length base-N vectors. Now we review the packing method of PriBET protocol in [13] for our protocol using our base-N encoding by the following way.

4.1 Packing Method for Our Protocol

As mentioned in our protocol of Section 3, we need to compute the SEDs E_λ in Eq. (5) using few polynomial additions and multiplications to reduce the cost where $1 \leq \lambda \leq k$. Let us construct a base-N integer vector $\mathbf{A} = (\alpha_0, \ldots, \alpha_{l_N-1}) \in R_t$ from a base-N vector $\alpha = (\alpha_1, \ldots, \alpha_{l_N})$ of length l_N. Furthermore, let us consider another base-N integer vectors $\mathbf{B}$ which is constructed by combining all base-N vectors in $\beta_\lambda = (\beta_{\lambda,1}, \ldots, \beta_{\lambda,l_N})$ as $\mathbf{B} = (\beta_{1,0}, \ldots, \beta_{1,l_N-1}, \ldots, \beta_{k,0}, \ldots, \beta_{k,l_N-1}) \in R_t$ of length $k \cdot l_N$. Here we want to compute many SEDs E_λ in one computation which can be done by measuring the SEDs between the vector $\mathbf{A}$ and each sub-vector in $\mathbf{B}$. Moreover, existing literature showed [12,16] that the secure inner product $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle$ helps to compute the SED between $\mathbf{A}$ and $\mathbf{B}_\lambda$. Hence, we pack these integer vectors by some polynomials with the highest degree $(x) = n$ in such a way so that inner product $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle$ does not wrap-around a coefficient of x with any degrees. For the integer vectors $\mathbf{A}$ and $\mathbf{B}$ with $n \geq k \cdot l_N$ and $1 \leq \lambda \leq k$, the packing method of Saha et al. [13] in the same ring $R = \mathbb{Z}[x]/(x^n + 1)$ is rewritten as

$$Poly_1(\mathbf{A}) = \sum_{i=0}^{l_N-1} \alpha_i x^i, \qquad Poly_2(\mathbf{B}) = \sum_{\lambda=1}^{k} \sum_{j=0}^{l_N-1} \beta_{\lambda,j} x^{l_N \cdot \lambda - (j+1)}. \tag{6}$$

Here the coefficients α_i and $\beta_{\lambda,j}$ are in the alphabets $\{0, 1, 2, \ldots, N-1\}^{l_N}$ instead of alphabets $\{0, 1\}^l$ as in [13]. If we multiply the above two polynomials, we can get the inner product computations which in turn helps the many square Euclidean distances computation between $\mathbf{A}$ and $\mathbf{B}_\lambda$. Moreover, this multiplication will produce another big polynomial where each of the SEDs can be obtained as a coefficient of x with different degrees. According to the SwHE described in

Section 5 of [13], the packed ciphertexts for $Poly_i(\mathbf{A}) \in R$ are defined for some $i = \{1, 2\}$ using the public key pk as

$$ct_i(\mathbf{A}) = \mathrm{Enc}(Poly_i(\mathbf{A}), pk) \in (R_q)^2 . \tag{7}$$

To get the inner product of the vectors $\mathbf{A}$ and $\mathbf{B}_\lambda$, we multiply the polynomials $Poly_1(\mathbf{A})$ and $Poly_2(\mathbf{B})$ in the same base ring R as follows.

$$\Big(\sum_{i=0}^{l_N-1} \alpha_i x^i\Big) \times \Big(\sum_{\lambda=1}^{k}\sum_{j=0}^{l_N-1} \beta_{\lambda,j} x^{l_N\cdot\lambda-(j+1)}\Big) = \sum_{\lambda=1}^{k}\sum_{i=0}^{l_N-1}\sum_{j=0}^{l_N-1} \alpha_i\beta_{\lambda,j} x^{i+l_N\cdot\lambda-(j+1)}$$
$$= \sum_{\lambda=1}^{k}\sum_{i=0}^{l_N-1} \alpha_i\beta_{\lambda,i} x^{l_N\cdot\lambda-1} + \mathrm{ToHD} + \mathrm{ToLD} = \sum_{\lambda=1}^{k} \langle \mathbf{A}, \mathbf{B}_\lambda\rangle x^{l_N\cdot\lambda-1} + \cdots \tag{8}$$

Here $\mathbf{A}$ is the vector of length l_N and $\mathbf{B}_\lambda$ is the λ-th sub-vector of $\mathbf{B}$ of the same length with $1 \le \lambda \le k$. Moreover, the ToHD (terms of higher degree) means $deg(x) > l_N\cdot\lambda - 1$ and the ToLD (terms of lower degrees) means $deg(x) < l_N\cdot\lambda - 1$. The result in Eq. (8) shows that one polynomial multiplication includes the many inner products of $\langle \mathbf{A}, \mathbf{B}_\lambda\rangle$. In addition, the following proposition is needed to hold for computing the many inner products over packed ciphertexts.

Proposition 1. *Let $\mathbf{A} = (\alpha_0, \alpha_1, \ldots, \alpha_{l_N-1}) \in R_t$ be an integer vector where $|\mathbf{A}| = l_N$ and $\mathbf{B} = (\beta_{1,0}, \ldots, \beta_{1,l_N-1}, \ldots, \beta_{k,0}, \ldots, \beta_{k,l_N-1}) \in R_t$ be another integer vector of length $k \cdot l_N$. For $1 \le \lambda \le k$, the vector $\mathbf{B}$ includes k sub-vectors where the length of each sub-vector is l_N. If the ciphertext of $\mathbf{A}$ and $\mathbf{B}$ can be represented as $ct_1(\mathbf{A})$ and $ct_2(\mathbf{B})$ respectively by Eq. (7) then under the condition of Lemma 1 (See Section 5 in [13] for details), decryption of homomorphic multiplication $ct_1(\mathbf{A}) \boxtimes ct_2(\mathbf{B}) \in (R_q)^2$ will produce a polynomial of R_t with $x^{l_N\cdot\lambda-1}$ including coefficient $\langle \mathbf{A}, \mathbf{B}_\lambda\rangle = \sum_{i=0}^{l_N-1} \alpha_i\beta_{\lambda,i} \bmod t$. Alternatively, we can say that homomorphic multiplication of $ct_1(\mathbf{A})$ and $ct_2(\mathbf{B})$ simultaneously computes the many inner products for $1 \le \lambda \le k$ and $0 \le i \le (l_N - 1)$.*

5 Secure Computation using Euclidean Distance

We perform the computation of base-N PriBET protocol of Section 3 using the SwHE scheme used in [13] and the packing method in Section 4.1. In addition, according to Eq. (5), we need to find out the values of the many SEDs E_λ. Let us consider two same base-N integers vectors $\mathbf{A}$ and $\mathbf{B}$ constructed by Algorithm 1 where $\mathbf{B}_\lambda = (\beta_{\lambda,0}, \ldots, \beta_{\lambda,l_N-1})$ is the λ-th sub-vector of $\mathbf{B}$ with $1 \le \lambda \le k$. From these integer vectors, E_λ can be computed with the help of the arithmetic computation between $\mathbf{A}$ and $\mathbf{B}_\lambda$ as

$$E_\lambda = \sum_{i=0}^{l_N-1} (\alpha_i^2 + \beta_{\lambda,i}^2 - 2\alpha_i\beta_{\lambda,i}) . \tag{9}$$

Now we construct $Poly_1(\mathbf{A})$ and $Poly_1(\mathbf{A}^2)$ (resp., $Poly_2(\mathbf{B})$ and $Poly_2(\mathbf{B}^2)$) from vector $\mathbf{A}$ (resp., $\mathbf{B}$) using the packing method in Eq. (6). With the help of inner product property in Eq. (8), now we compute $ct(E_\lambda)$ of the Eq. (9) over packed ciphertext $ct_1(\mathbf{A})$, $ct_1(\mathbf{A}^2)$, $ct_2(\mathbf{B})$, and $ct_2(\mathbf{B}^2)$ which are obtained from $Poly_1(\mathbf{A})$, $Poly_1(\mathbf{A}^2)$, $Poly_2(\mathbf{B})$, and $Poly_2(\mathbf{B}^2)$) respectively by the Eq. (7). Moreover, we calculate $ct(E_\lambda)$ from Proposition 1 and the packed ciphertext vector $ct_1(\mathbf{A}) \in R_q$, $ct_1(\mathbf{A}^2) \in R_q$, $ct_1(\mathbf{B}) \in R_q$, and $ct_2(\mathbf{B}^2) \in R_q$ in three homomorphic multiplications and two homomorphic additions. Here $ct(E_\lambda)$ equals

$$ct_1(\mathbf{A}^2) \boxtimes ct_2(\mathbf{V}_\epsilon) \boxplus ct_2(\mathbf{B}^2) \boxtimes ct_1(\mathbf{V}_{l_N}) \boxplus (-2ct_1(\mathbf{A}) \boxtimes ct_2(\mathbf{B})) \tag{10}$$

where $\mathbf{V}_\epsilon$ denotes an integer vector like $(1, \ldots, 1)$ of length $k \cdot l_N$ and $\mathbf{V}_{l_N}$ denotes another integer vector $(1, \ldots, 1)$ of length l_N. In addition, $\boxplus$ (resp. $\boxtimes$) stands for homomorphic addition (resp., multiplication). The above-encrypted polynomial $ct(E_\lambda)$ includes many SEDs as the coefficients of different degrees of x. Bob sends $ct(E_\lambda)$ to Alice for decryption. According to Proposition 1 and our protocol, Alice decrypts $ct(E_\lambda)$ in the ring R_q using her secret key and extracts E_λ as a coefficient of $x^{l_N \cdot \lambda - 1}$ from the plaintext of $ct(E_\lambda)$. Then Alice checks whether at least of one of the E_λ contains 0 or not to help Bob to decide either equality or non-equality.

Concealing Extra Information from Leakage. In the PriBET protocol of Saha et al. [13], Bob in the cloud sent the whole encrypted polynomial to Alice for decryption to decide something after the computation. Here Bob could see every coefficient of the polynomial whereas she needs to check some coefficients with a particular degree of x. For this reason, some extra information leakage problem exists in the Saha et al.'s protocol. To compute E_λ for our protocol by Bob in the cloud, he also needs a decryption help from Alice for some decision making since he does not have the secret key. From the above discussion of secure computation, Alice needs to check only the coefficient of $x^{l_N \cdot \lambda - 1}$ for the large polynomial $ct(E_\lambda)$ produced by Bob. Also, all other coefficients of our n degree polynomial can be published to Alice if Bob does not conceal those coefficients. In our protocol, we conceal the extra information from leakage to Alice by adding some random masks at the cloud (Bob) ends. We can conceal $ct(E_\lambda)$ by adding a random polynomial r in the base ring R as $r = \sum_{h=1}^{n/l_N} \sum_{i=l_N(h-1)}^{l_N \cdot h - 2} r_i x^i$. Now Bob adds r to the ciphertext $ct(E_\lambda)$ as $ct(E'_\lambda) = ct(E_\lambda) \boxplus r$. Besides, the resulting ciphertext $ct(E'_\lambda)$ contains all required information as a coefficient of $x^{l_N \cdot \lambda - 1}$ and conceals all other coefficients using the random masks. In this way, we protect $ct(E_\lambda)$ from leaking any information to Alice except the coefficient of $x^{l_N \cdot \lambda - 1}$.

6 Experimental Analysis

In this section, we show the parameter settings of our experiments along with security level. We also show the selection process of our base-N encoding size and the performance of our protocol towards big data.

Table 1. Performance of base-N PriBET protocol for a data size of 16384

Data size (bits)	Encoding size (N)	Block size (γ)	Plaintext space (t)	Ciphertext space (q)	Total computation time (ms)	Lattice dimension(n)	Security level
8	2^4	1024	2^{11}	61-bit	1469	2048	≥ 140
16		512			2953		
32		256			5860		
8	2^8	4096	2^{16}	71-bit	875	4096	
16		2048	2^{17}	73-bit	1454		
32		1024	2^{18}	75-bit	2844		
16	2^{16}	4096	2^{32}	103-bit	982		
32		2048	2^{33}	105-bit	1718		
64		1024	2^{34}	107-bit	3157		
32	2^{32}	8192	2^{64}	167-bit	1312	8192	
64		4096	2^{65}	169-bit	1937		

6.1 Parameters Settings

As discussed in Section 5 of [13], we selected proper values of the parameters (n, t, q, σ) of our used security scheme for successful decryption and to achieve a certain security level. In addition, we need to select the appropriate value for our encoding size N. The security analysis of this protocol is skipped due to page limitation which can be addressed in the full version of the paper.

Correctness Side. Here we show the correctness of our protocol for computing $ct(E_\lambda)$ for different lattice dimensions. According to the Lemma 1 in [13], the correctness of ciphertext $ct(E_\lambda)$ holds if

$$||\langle ct(E_\lambda), s\rangle|| \leq q/2\,. \tag{11}$$

As mentioned in [16], we consider the upper bound Φ of ∞-norm size $\|\langle ct, s\rangle\|_\infty$ for any fresh ciphertext $ct \in (R_q)^2$. In addition, the value of the upper bound Φ is $2t\sigma^2\sqrt{n}$ (see Theorem 3.3 in [6]). Here the ∞-norm size of $ct(E_\lambda)$ in Eq. (10) is defined by the inequality as $\|\langle ct(E_\lambda), s\rangle\|_\infty < 2n\Phi^2 + 2n\Phi^2$ (see [16] for details). Furthermore, we take the value of Φ as $2t\sigma^2\sqrt{n}$ (see [6] for details). Now the inequality in Eq. (11) can be represented as $\|\langle ct(E_\lambda), s\rangle\|_\infty < 2n\Phi^2 + 2n\Phi^2 \approx 8n^2t^2\sigma^4$. The correctness for the inequality in Eq. (11) for the ciphertext $ct(E_\lambda)$ can be found if it satisfies

$$16n^2t^2\sigma^4 \leq q\,. \tag{12}$$

Chosen Parameters. Here we need the lattice dimension n to be greater than $k \cdot l_N$ for our protocol. Since we required to compute the SED between two base-N integer vectors of length l_N. Now the plaintext space t should satisfy the relation

$$t \geq l_N \cdot N^2\,. \tag{13}$$

As shown in Table 1, we consider encoding size $N = 2^4 \sim 2^{32}$ for the lattice dimension 2048, 4096, and 8192 with the data size $k = 16384$. We also consider integer data size l to be 8-bit, 16-bit, 32-bit, and 64-bit for comparison in the base-N PriBET protocol. Furthermore, we set t according to Eq. (13) for our plaintext space R_t. According to the work in [6], we choose $\sigma = 8$ and the value of q must be greater than $16n^2p^2\sigma^4$ for the ciphertext space R_q as in Eq. (12). Therefore, we fix our parameters as (n, t, q, σ, N) as shown in Table 1 and Table 2. We did a block-wise computation to manage our dataset of 16384 integers within lattice dimension of 2048, 4096, and 8192. We set the block size γ as 256, 512, 1024, 2048, 4096, and 8192 for the lattice dimension of 2048, 4096, and 8192.

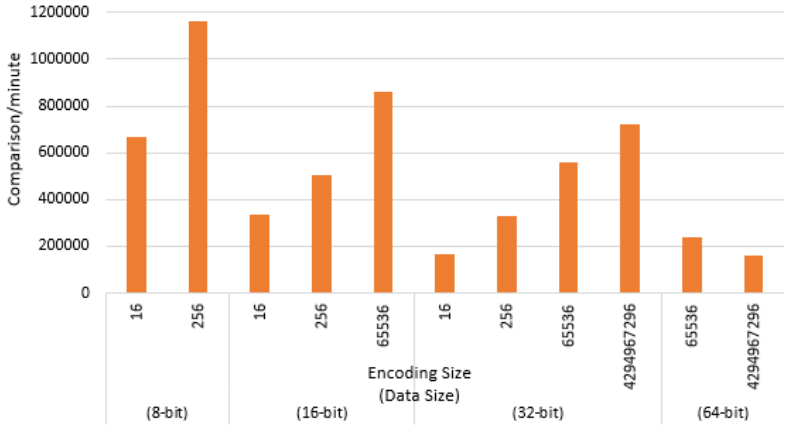

Fig. 1. Comparative performances of our protocol for different encodings (base-16, base-256, base-65536, and base-4294967296) using the lattice dimensions of 2048, 4096, and 8192 with $8 \sim 64$-bit integers.

Security Level. In our experiment, we consider the security of the encryption scheme against two attacks namely distinguishing attack [10] and decoding attack [8]. According to the discussion of Lindner and Peikert [8], we consider every parameter setting to provide more than 128-bit security level to secure our protocol against the distinguishing attack and more powerful decoding attack with the advantage $\epsilon = 2^{-64}$. In addition, a root Hermite factor $\delta < 1.0050$ is required to achieve an 80-bit security level that was shown by Chen and Nguyen [2] in lattice-based cryptographic schemes. As discussed in [6], the running time t_{adv} is defined as $lg(t_{adv}) = 1.8/\lg(\delta) - 110$ where the root Hermite factor δ is expressed as

$$c \cdot q/\sigma = 2^{2\sqrt{n. \cdot \lg(q) \cdot \lg(\delta)}}. \tag{14}$$

As shown in Eq. (12) and Eq. (13), both t and q should be increased with the increase of the encoding size N. If we use a low lattice dimension for a high

encoding size, we will get security level less than 128-bit according to Eq. (14) which is not desirable. As shown in Table 1, if the encoding size N is 16 and lattice dimension is 2048 then we get a security level of 140. But if $N = 256$ and $n = 2048$ again then we get a security level 104 which is not acceptable for our case. So we increase the lattice dimension with the increase of encoding size to get a better security level. According to data of Table 2 in [17], our parameters setting provides more than 140-bit security level to protect the security algorithm from some distinguishing attacks as shown Table 1.

6.2 Implementation Details

We implemented both Saha et al. [13] and our protocols in C programming language with Pari C library (version 2.7.5) [15] and ran the programs on a single machine configured with 3.6 GHz Intel core-i7 processor and 8GB RAM using Linux environment. Here we did two types of experiments. One is for selection of encoding parameter and another for comparative analysis with the existing method. To do these experiments we selected suitable values of our parameters for our security scheme in [13] and encoding technique described in Section 2 respectively. We considered maximum data size of 16384 with $8 \sim 64$-bit integers for our experiments.

Table 2. Parameter settings of Saha et al.'s protocol [13] and our protocol

Integer size (l)	Data size (k)	Encoding size (N)		Lattice dimension (n)		Plaintext space (t)		Ciphertext space (q)	
		Saha et al.	Our method	Saha et al.	Our method	Saha et al.	Our method	Saha et al.	Our method
8	4096	2	2^8	32768	4096	2048	2^{16}	69 bits	73 bits
16	4096		2^{16}	65536	4096		2^{32}	71 bits	105 bits
32	2048			65536	4096		2^{33}	71 bits	107 bits

6.3 Selection of Encoding Size (N) and Performance towards Big Data

Table 1 shows the performance of our base-N PriBET protocol for the lattice dimension of 2048, 4096, and 8192 with a data size of 16384. Here we did the experiments for different values of our encoding size $N(2^4 \sim 2^{32})$. Furthermore, we show a comparative performance of our different encoding size for the lattice dimension of 2048, 4096, and 8192 as shown in Fig. 1. Also, we were able to select the value of our encoding size N as low as $2^4 = 16$ and as high as $2^{32} = 4294967296$. We tried to select the maximum value of encoding 2^{64} where the

computation is out of the capacity our machine due to a buffer overflow. It also happens due to increasing the value of plaintext space t and ciphertext space q. From this figure, it is clear that batch equality comparison is faster if data size and encoding size are same for most of the cases. In addition, we achieved a good performance for the encoding size of $2^8 = 256$ and $2^{16} = 65536$ with the data size of 8 and 16-bit respectively. So we chose two effective values of base-N encoding size as 2^8 and 2^{16}.

Moreover, our experiments also showed that our protocol was able to do over 1.1 million and 862 thousand of equality comparisons per minute with the encoding size of $2^8 = 256$ and $2^{16} = 65536$ for a data size of 8-bit and 16-bit respectively. Moreover, our protocol was able to compute more than 700 thousand (resp. 200 thousand) equality comparisons per minute for 32-bit (resp. 64-bit) data with an encoding size of 2^{32}.

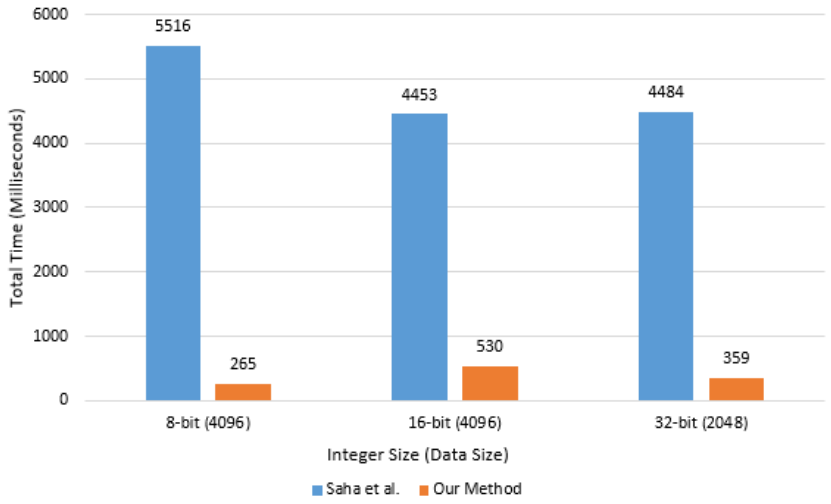

Fig. 2. Performance comparison between Saha et al. [13] and our method for the data size 2048 and 4096 with length of 8 $\sim$ 32-bit using base-256 and base-65536 encodings

6.4 Comparative Analysis

In this section, we show comparative performances of our protocol with respect to Saha et al.'s protocol [13] for batch comparison to find out the equalities of an integer with a set of k integers. Saha et al. used the Hamming distance computation for their PriBET protocol over binary encoding because Hamming distance computation works only for binary data. Here we used the base-N encoding to minimize the cost of computation by reducing lattice dimension. As mentioned in Section 2, we achieved the lattice dimension reduction by a factor

of $\log_2 N$ than Saha et al. which reflects in the parameter settings of lattice dimension for both of the protocols as shown in Table 2. Due to using base-N encoding, we use the SED computation to find the distance between a given query and an existing dataset. According to Section 6.3, we used the two best encoding size of 2^8 and 2^{16} for getting the better performance. Table 2 shows the used parameters settings for both of the protocols. Furthermore, we took the integers set of 2048 and 4096 with a practical bit size of 8-bit, 16-bit, and 32-bit for the comparison. For the data size of 2048 and 4096 using base-256 and base-65536 encoding, the comparative performance of our protocol with respect to Saha et al.'s protocol is shown in Fig.2 where timing was taken in milliseconds. Our protocol showed the best performance than that of Saha et al. for 8-bit integer comparison with a data size of 4096 and a less good performance for a 16-bit integer with the same data size. Overall, our protocol performed more than $8 \sim 20$ times as fast as Saha et el.'s protocol for the batch equality comparison. Besides, we achieved more than 140-bit security using our parameter settings described in Section 6.1.

7 Conclusions

In this paper, we discussed an efficient base-N PriBET protocol using ring-LWE based somewhat homomorphic encryption in the semi-honest model. For this purpose, we have shown a fixed length base-N encoding algorithm to reduce the cost of equality comparison. In addition, we experimented our protocol using different encoding size to find out the best value of our encoding size N. Our protocol was able to do more than 1.1 million (resp. 862 thousand) comparisons per minute for 8-bit (16-bit) integer batch comparison. Also, we have been able to show that our protocol works more than $8 \sim 20$ times faster than the protocol of Saha et al.'s protocol. We also believe that this achievement of around a million of comparisons per minute is big a step towards big data processing. We hope that our research will inspire future researches to use base-N encoding rather than binary encoding for many computation purposes because of reducing the lattice dimension by a factor of $\log_2(N)$.

Acknowledgments. This work is supported in part by JSPS Grant-in-Aids for Scientific Research (A) JP16H01705 and for Scientific Research (B) JP17H01695.

References

1. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_29
2. Chen, Y., Nguyen, P. Q.: BKZ 2.0: Better lattice security estimates. In Advances in Cryptology — ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Berlin Heidelberg (2011)

3. Couteau, G.: Efficient secure comparison protocols, Cryptology ePrint Archive, Report 2016/544, 2016, http://eprint.iacr.org/2016/544.
4. Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it Communications of the ACM, vol. 39, no. 5, pp. 77–85, 1996.
5. Gentry, C.: Fully homomorphic encryption using Ideal lattices. In: Symposium on Theory of Computing - STOC 2009, pp. 169–178. ACM, New York (2009)
6. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: ACM Workshop on Cloud Computing Security Workshop, CCSW 2011, pp. 113–124, ACM, New York (2011)
7. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality. 1(1), 59–98 (2009)
8. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (eds) Topics in Cryptology — CT-RSA 2011. CT-RSA 2011. LNCS, vol 6558, pp. 319–339. Springer, Berlin Heidelberg (2011)
9. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gilbert, H. (eds) Advances in Cryptology — EUROCRYPT 2010. EUROCRYPT 2010. LNCS, vol. 6110. pp 1–23. Springer, Heidelberg (2010)
10. Micciancio, D., Regev, O.: Lattice-based cryptography. In Post-Quantum Cryptography, pp. 147–191, Springer, Heidelberg (2009)
11. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphism. In: DeMillo, R. A., Dobkin, D. P., Jones, A. K., Lipton, R. J. (eds.) Foundations of Secure Computation, pp. 169–177. Academic Press, New York (1978)
12. Saha, T.K., Koshiba, T.: An enhancement of privacy-preserving wildcards pattern matching. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (eds) Foundations and Practice of Security. FPS 2016. LNCS, vol. 10128. pp. 145–160, Springer, Cham (2017). doi:10.1007/978-3-319-51966-1_10
13. Saha, T. K., Koshiba, T.: Private equality test using ring-LWE somewhat homomorphic encryption, In 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWConCSE), pp. 1–9, IEEE (2016)
14. Saha T.K., Mayank, Koshiba, T.: Efficient protocols for private database queries. In: Livraga, G., Zhu, S. (eds) Data and Applications Security and Privacy XXXI. DBSec 2017. LNCS, vol. 10359. pp. 337–348. Springer, Cham (2017)
15. The PARI∼Group, PARI/GP version `2.7.5`, Bordeaux, 2014, `http://pari.math.u-bordeaux.fr/`
16. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Practical packing method in somewhat homomorphic encryption. In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W. (eds.): DPM 2013 and SETOP 2013. LNCS, vol. 8247, pp. 34–50, Springer, Heidelberg (2014)
17. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure pattern matching using somewhat homomorphic encryption. In Proceedings of the 2013 ACM workshop on Cloud computing security workshop, pp. 65–76, ACM (2013)
18. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure statistical analysis using RLWE-based homomorphic encryption. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144, pp. 471–487. Springer, Cham (2015). doi:10.1007/978-3-319-19962-7_27
19. Yao, A. C.: Protocols for secure computations. In 23rd Annual Symposium on Foundations of Computer Science, 1982. pp. 160–164. IEEE (1982)
20. 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 (2013)