2016 3rd Asia-Pacific World Congress on Computer Science and Engineering

Private Equality Test using Ring-LWE Somewhat Homomorphic Encryption

Tushar Kanti Saha* and Takeshi Koshiba†
Graduate School of Science and Engineering
Saitama University
Saitama, Japan
*Email: *s15dm054@mail.saitama-u.ac.jp,†koshiba@mail.saitama-u.ac.jp*

***Abstract*—We propose two secure protocols namely private equality test (PET) for single comparison and private batch equality test (PriBET) for batch comparisons of l-bit integers. We ensure the security of these secure protocols using somewhat homomorphic encryption (SwHE) based on ring learning with errors (ring-LWE) problem in the semi-honest model. In the PET protocol, we take two private integers input and produce the output denoting their equality or non-equality. Here the PriBET protocol is an extension of the PET protocol. So in the PriBET protocol, we take a single private integer and another set of private integers as inputs and produce the output denoting whether single integer equals at least one integer in the set of integers or not. To serve this purpose, we also propose a new packing method for doing the batch equality test using few homomorphic multiplications of depth one. Here we have done our experiments at the 140-bit security level. For the lattice dimension 2048, our experiments show that the PET protocol is capable of doing any equality test of 8-bit to 2048-bit that require at most 107 milliseconds. Moreover, the PriBET protocol is capable of doing about 600 (resp., 300) equality comparisons per second for 32-bit (resp., 64-bit) integers. In addition, our experiments also show that the PriBET protocol can do more computations within the same time if the data size is smaller like 8-bit or 16-bit.**

***Keywords*-private; equality test; batch equality; homomorphic; encryption;**

I. INTRODUCTION

Private equality test (PET) is a kind of secure computation between two users who want to compare their information for checking the equality without disclosing any information to each other in case of they do not equal. In this paper, we use the PET protocol for finding equality of two private integers. The PET protocol [2] is also known as socialist millionaire problem [3] which can be used as a sub-protocol of many large protocols where the private comparison is required. Moreover, we also show an extension of the PET protocol called the private batch equality test (PriBET) protocol for securely comparing an integer with k integers in a single computation where k represents the block size. Recently, the concept of private 'batch equality test' protocol was introduced by Couteau [4] to show the equality tests for $16 \sim 128$-bit data between two parties. Here the batch equality of integers means comparing a single integer from one party with a set of integers from another party. The PET and PriBET protocols are appealing due to its numerous applications in database queries, data mining, machine learning, and so on. On the other hand, cloud computing has established itself as an effective service after commercialization of Amazon EC2 in 2005 [1]. Now different organizations such as financial, research, medical, educational, and so on are uploading their data to the cloud. But they do not believe the cloud for securing their data. So they want to secure their data using some encryption methods. They also want to compute on the encrypted data without decrypting it. Therefore, homomorphic encryption is a solution for them which allows meaningful computation such as addition and multiplication on encrypted data.

The term homomorphic encryption was first coined by Rivest et al. in 1978 [5]. Later on, many research works had been proposed using homomorphic encryption scheme which supports either addition [6], [7], [8] or multiplication [9], [10] on encrypted data but not both. But only addition or multiplication in homomorphic computations is not enough for some extended computations in the fields of bioinformatics, data mining, machine learning, and so on. In 2005, Boneh et al. [11] proposed a homomorphic encryption which supports a number of additions and one multiplication. But one multiplication is not enough for many computations. In 2009, Gentry did the ground-breaking work of fully homomorphic encryption (FHE) which supports a number of additions and multiplications [12]. Gentry constructed FHE scheme by applying the bootstrapping technique in somewhat homomorphic encryption (SwHE). But FHE technique is still far from practical implementation due to its slowness in computations [13]. Moreover, Brakerski and Vaikuntanathan [15] proposed another somewhat homomorphic scheme using the ring learning with errors (ring-LWE) concept of Lyubashevsky et al. [14]. In this article, we use SwHE [15] because it works faster than FHE and supports many additions and a few multiplications. Now organizations are interested doing the secure computation in an on-line auction, genomic computation, machine learning, data mining, and so on where the PET protocol is necessary for doing the equality tests.

In addition, nowadays people in many countries are required to use sensitive information like social security number (SSN) to open accounts in banks, credit card companies, insurance companies, and some other financial orga-

978-1-5090-5753-5/16 $31.00 © 2016 IEEE
DOI 10.1109/APWC.on.CSE.2016.11

nizations. They are taking services from hospitals, clinics, gas company, water supplier companies, electric companies, and so on. They are also using their credit card number to pay the bills of their shopping, hospitals, utilities, and others. Furthermore, they are complaining their insurances during paying their bills wherever necessary. Therefore, these organizations need to securely verify their customers' profile using SSN with a huge dataset of another company containing many SSNs. Due to the availability of SwHE in the cloud, organizations are eager to outsource their computations where the PriBET protocol may play a major role.

II. PRIOR WORKS

In this section, we review some literature for the PET and PriBET protocols. To date, very few protocols have been proposed for the PET protocol. In 1996, Fagin et al. [2] first proposed the PET protocol to show that the two people possess same information without revealing any information to each other in case of they do not. In 2005, Li and Wu [16] presented the PET protocol using additively homomorphic encryption scheme of Paillier [8] in the semi-honest model. But additively homomorphic encryption is not enough for doing some extended computations. In 2011, Ciou and Tso [17] proposed another privacy preserved two-party equality testing protocol using commutative encryption scheme. But none of these protocols addressed the implementation showing their practicality. In 2015, Ardestani [18] showed the implementation of the PET protocol for malicious adversaries using oblivious transfers. But it took about one minute to compare 80-bit input which is impractical to implement in the cloud. In addition, Couteau [4] recently addressed 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 these protocols, some implementations are indispensable.

III. APPLICATIONS

In this section, we explore some of the application areas of the PET and PriBET protocols. These applications are discussed below.

A. *On-line auction*

For some on-line auctions, sometimes it is required to perform the verification after finalizing the winner by the auctioneer. Here winning bid value is an important information for the both the auctioneer and winner. But a bidder who lost the auction may request the auctioneer to verify his bid value with the winning bid value where our PET protocol can solve this problem. Our PET protocol is not only useful in the on-line auction verification process but also useful for checking the equality of two bid values during the auction [19].

B. *Genomic computation*

Nowadays the importance of privacy-preserving genomic computation is increased because it can discover human personal DNA, RNA or protein profile [20] from some genomic databases without disclosing those databases to the public. Furthermore, research organizations, hospitals, and laboratories are now interested in storing these kinds of sensitive information in the cloud server after encryption. They also want to do the genomic computation using these encrypted data. A genomic computation like edit distance computation [21] requires the PET protocol as a sub-protocol. Moreover, the PET protocol can be used for large DNA string matching computation. In addition, the PriBET protocol can be used for exact DNA matching in some large databases.

C. *Machine learning and Data mining*

The PET protocol has many applications in machine learning as a sub-protocol. Bost et al. [22] used argmax and argmin functions of machine learning classifiers where the PET protocol is necessary for finding the equality along with the comparison protocol. In addition, Lindell and Pinkas [23] used the PET protocol to do the multi-party computation for privacy-preserving data mining applications.

D. *Private Database Query Processing*

The private batch equality test (PriBET) protocol is useful where a single data is needed to compare with many data to find the equality. We know that credit card number and social security number (SSN) are very important information for every person. Our PriBET protocol can be used by an on-line shop to securely check the credit card number with some large databases. Moreover, a hospital or clinic can use the PriBET protocol to securely check particular customer's health insurance by using SSN with the database of the corresponding insurance company.

Notations: $\mathbb{Z}$ and $\mathbb{R}$ denote the ring of integers and the field of real numbers respectively. For a prime number q, the ring of integer is denoted by $\mathbb{Z}_q$. In addition, $\mathbb{Z}^n$ defines an n-dimensional integer vector space. For a vector $A = (a_0, a_1, \ldots, a_{n-1})$, the maximum norm of $\|A\|_\infty$ is defined as $\max |a_i|$. Let $\langle A, P \rangle$ denote the inner product between two vectors A and P. Moreover, the function $\text{Enc}(m, pk) = ct$ defines the encryption of message m using the public key pk to produce the ciphertext ct. The ciphertexts ct_{add} and ct_{mul} denote homomorphic addition and multiplication of ciphertexts $ct = \text{Enc}(m, pk)$ and $ct' = \text{Enc}(m', pk)$. The distribution $D_{\mathbb{Z}^n}$ indicates the n-dimensional discrete Gaussian distribution. Here $k \in \mathbb{Z}$ represents the block size of the comparison for the PriBET protocol.

IV. OUR PROTOCOL

Here we describe the private equality test (PET) and private batch equality test (PriBET) protocols in the following sub-section.

A. The PET Protocol

To describe this private equality test protocol, let us consider a private on-line auction blind verification scenario. Suppose Alice is an auctioneer who has decided the final bid value of her auction. On the other hand, a bidder who has lost the auction wants to verify winning bid value. Here they do not want to reveal their private values if they do not match, but want to get an equality comparison result. To solve this problem, a third party like Bob in the cloud will do the computation on behalf them without knowing the actual values. In this comparison scenario, let Alice has an l-bit integer as $a = (a_1, \dots, a_l)$ and the bidder has another l-bit integer as $b = (b_1, \dots, b_l)$. Here, we know that the Hamming distance H_{dis} between two l-bit integers can find out their equality. If $H_{dis} = 0$ for two integers comparison, then we say that those are equal. Here our equality test for single comparison can be realized by the following equation.

$$c = \sum_{i=1}^{l} |a_i - b_i| = \sum_{i=1}^{l} (a_i + b_i - 2a_i b_i) \qquad (1)$$

Here c indicates the Hamming distance between two integers a and b. So if $c = 0$ then we can say that $a = b$; otherwise $a \neq b$. Now we can describe the PET protocol by the following steps.

Inputs: $a = (a_1, \dots, a_l)$ and $b = (b_1, \dots, b_l)$.
Output: $a = b$ or $a \neq b$
PET protocol:

1) Alice generates the public key and private key by herself and sends the public key to the bidder through a secure channel. Then she encrypts the final bid value $a = (a_1, \dots, a_l)$ using her public key and sends it to Bob.
2) The bidder also uses Alice's public key to encrypt his bid value $b = (b_1, \dots, b_l)$ and sends the value to Bob.
3) Bob does the secure computation of integers equality test as in Eq. (1) and sends the encrypted result ct_{r_1} to Alice to verify whether $c = 0$.
4) Alice decrypts ct_{r_1} using her secret key and checks the value of c and sends the acknowledgment to Bob as 0 if $c = 0$; otherwise, she sends 1.
5) Then Bob decides either $a = b$ or $a \neq b$ depending on the acknowledgment.

B. The PriBET Protocol

Suppose a patient has taken a service from a hospital and complaining his health insurance during paying his bill. Now hospital (Alice) wants to check the patient's health insurance using his social security number (SSN). Since SSN is an important information for the patient, so the hospital cannot disclose its patient information along with SSN to the insurance company. On the contrary, the insurance company cannot disclose its k customers' information to the hospital. To solve this problem, a third party like Bob in the cloud will do the computation on behalf them without knowing the actual values. In this comparison scenario, let Alice has an l-bit integer as $a = (a_1, \dots, a_l)$ and insurance company has k number of l-bit integers as $b_m = (b_{m,1}, \cdots, b_{m,l})$ where $1 \leq m \leq k$. Moreover, we know that the Hamming distance H_{dis} between two l-bit integers can find out whether they are equal. If $H_{dis} = 0$ for two integers comparison, then we say that those are equal. Here, if $a = b_m$ for some $1 \leq m \leq k$, then we have two option for the security of index m; either we cannot know the value of m, or we can specify such m. In our PriBET protocol, Alice can know such index m. Since such index is not actual index that exists in the databases of the insurance company, so leakage of such information to Alice does not harm the security of our protocol. In addition, if $a = b_m$ for some $1 \leq m \leq k$ then Alice sends only the acknowledgment to Bob regarding equality without leaking any information about the index m to Bob. For some $1 \leq m \leq k$, our equality test for batch comparisons can be realized by the following equation.

$$d_m = \sum_{i=1}^{l} |a_i - b_{m,i}| = \sum_{i=1}^{l} (a_i + b_{m,i} - 2a_i b_{m,i}) \qquad (2)$$

Here, d_m defines the Hamming distance between two binary vectors a and b_m. Moreover, if d_m in Eq. (2) is 0 for some positions of m then we can say that $a = b_m$; otherwise $a \neq b_m$. In this way, Alice securely verifies her customer with the help of Bob. Now we can describe our PriBET protocol by the following steps.

Inputs: $a = (a_1, \dots, a_l)$ and $\{b_1, b_2, ..., b_k\}$, where $b_m = (b_{m,1}, \dots, b_{m,l})$ for each m in $\{1, 2, ..., k\}$.
Output: $\exists m[a = b_m]$ or $\forall m[a \neq b_m]$
PriBET protocol:

1) Alice generates the public key and private key by herself and sends the public key to the insurance company through a secure channel. Then she encrypts the SSN $a = (a_1, \dots, a_l)$ of her patient using her public key and sends it to Bob.
2) The insurance company also uses Alice's public key to encrypt k SSNs $b_m = (b_{m,1}, \dots, b_{m,l})$ where $1 \leq m \leq k$ and sends the value to Bob.
3) Bob does the secure computation of batch equality test as in Eq. (2) and sends the encrypted result ct_{r_2} to Alice to verify whether at least one of d_m's is equal to 0.
4) For $1 \leq m \leq k$, Alice decrypts ct_{r_2} using her

secret key and checks each value d_m and sends the acknowledgment to Bob as 0 for at least one of the $d_m = 0$; otherwise, she sends 1.

5) Then Bob decides the equality or non-equality depending on the acknowledgment.

Remark 1. Here the both of our protocols are secure under the assumption that Bob is semi-honest (also known as honest-but-curious), i.e., he always follows the protocols but tries to learn information from the protocols.

V. SECURITY OF THE SCHEME

In this section, we review the asymmetric SwHE scheme in [24] and its correctness. In 2011, Brakerski and Vaikuntanathan [15] proved the correctness of this scheme.

A. Asymmetric SwHE Scheme

In 2015, Lauter et al. [24] showed a SwHE scheme which is a public key variant of BV's SwHE scheme [15]. For the SwHE scheme in [24], we need to consider some parameters as follows.

- $f(x)$: is a cyclotomic polynomial where $f(x) = x^n + 1$.
- n: an integer which represents the lattice dimension of the ring $R_q = \mathbb{Z}_q/f(x)$. Here n also represents the degree of polynomials which is a power of 2 such as 1024 or 2048.
- q: modulus q is an odd prime such that $q \equiv 1 (\mathrm{mod}\ 2n)$ defining the ring $R_q = R/qR = \mathbb{Z}_q/\langle f(x)\rangle$ which denotes a ciphertext space.
- t: a prime $t < q$, which defines the message space of the scheme as $R_t = \mathbb{Z}_t[x]/\langle f(x)\rangle$, the ring of integer polynomials modulo $f(x)$ and t.
- σ: is a parameter which defines a discrete Gaussian error distribution $\chi = D_{\mathbb{Z}^n,\sigma}$ with an n-dimensional integer vector $\mathbb{Z}^n$ and a standard deviation σ where $\sigma = 4 \sim 8$.

Now we can discuss the key generation, encryption, homomorphism, and decryption properties of SwHE scheme in [24] as follows:

1) Key generation: Generate a ring element $R \ni s \leftarrow \chi$ for our secret key $sk = s$. We then sample a uniformly random element $a_1 \in R_q$ and an error $R \ni e \leftarrow \chi$. Now we get the public key pair as $pk = (a_0, a_1)$ with $a_0 = a_1 s + te$.

2) Encryption: For a given message $m \in R_t$ and a public key $pk = (a_0, a_1)$, the encryption algorithm first samples $R \ni u, f, g \leftarrow \chi$ then encryption can be defined by a ciphertext pair $(c_0, c_1) = ct$ as follows.

$$\mathrm{Enc}(m, pk) = (c_0, c_1) = (a_0 u + tg + m, -(a_1 u + tf)) \quad (3)$$

Here, the plaintext $m \in R_t$ is also in R_q because $t < q$.

3) Homomorphic Operations: Generally, homomorphic operations like addition ($\boxplus$) and multiplication ($\boxtimes$) are between two ciphertexts $ct = (c_0, \ldots, c_\alpha)$ and $ct' = (c'_0, \ldots, c'_\beta)$. So the homomorphic operations between two ciphertexts can be defined as follows.

$$\begin{cases} ct_{add} = ct \boxplus ct' = \left(c_0 + c', \ldots, c_{max(\alpha,\beta)} + c'_{max(\alpha,\beta)}\right) \\ ct_{mul} = ct \boxtimes ct' = \displaystyle\sum_{i=0}^{\alpha+\beta} \hat{c}_i z^i = \left(\sum_{i=0}^{\alpha} c_i z^i\right)\left(\sum_{j=0}^{\beta} c'_j z^j\right) \end{cases} \quad (4)$$

In addition, we can also define the subtraction as similar to component-wise addition.

4) Decryption: For a fresh or homomorphically operated ciphertext $ct = (c_0, \ldots, c_\alpha)$ and $t \in R_t$, general decryption can be defined as

$$\mathrm{Dec}(ct, sk) = [\tilde{m}]_q \bmod t \quad (5)$$

where $\tilde{m} = \sum_{i=0}^{\alpha} c_i s^i$. For the secret key vector $\mathbf{s} = (1, s, s^2, \ldots)$, we can simply rewrite $\mathrm{Dec}(ct, sk) = [ct, s]_q \bmod t$. For example, a fresh ciphertext $ct = (c_0, c_1)$ generated by (3) then we have

$$\langle ct, \mathbf{s}\rangle = (a_0 u + tg + m) - s \cdot (a_1 u + tf) = m + t \cdot (ue + g - sf) \quad (6)$$

in the ring R_q since $a_0 - a_1 s = te$. If the value $m + t \cdot (ue + g - sf)$ does not wrap-around mod q (all errors $e, f, g, u \leftarrow \chi$ must be sufficiently small) then we have $[\langle ct, \mathbf{s}\rangle]_q = m + t \cdot (ue + g - sf)$ in the base ring R. Here, it is clear that we can recover plaintext m by mod t operation. In addition, for two ciphertexts ct_1 and ct_2, we clearly have the following by the homomorphic operation if no wrap-around happens in the encrypted results after homomorphic operations.

$$\begin{cases} \langle ct_1 \boxplus ct_2, \mathbf{s}\rangle = \langle ct_1, \mathbf{s}\rangle + \langle ct_2, \mathbf{s}\rangle \\ \langle ct_1 \boxtimes ct_2, \mathbf{s}\rangle = \langle ct_1, \mathbf{s}\rangle \cdot \langle ct_2, \mathbf{s}\rangle \end{cases} \quad (7)$$

B. Security of Our Scheme

We can show the security of our scheme by the polynomial ring-LWE assumption (ring-LWE$_{n,q,\chi}$) as done by Lauter et al. [24] for the given parameters (n, q, t, σ). Let the ring $R_q = \mathbb{Z}_q/f(x)$ where $f(x) = (x^n + 1)$ be the cyclotomic polynomial of degree n. Let $s \leftarrow \chi = D_{\mathbb{Z}^n,\sigma}$ be a uniformly random ring element. The assumption holds for any polynomial number of samples of the form

$$(a_i, b_i = a_i \cdot s + e_i) \in (R_q)^2$$

where a_i is uniformly random in R_q and e_i is drawn from the error distribution χ. Here the a_i's are uniformly random in R_q and b_i's ($b_i = a_i \cdot s + e_i$) are also uniform in R_q. Therefore, it is hard to distinguish (a_i, b_i) from a uniformly random pair $(a_i, b_i) \in (R_q)^2$. Besides, Lyubashevsky et al. [14] showed that the ring-LWE assumption is reducible to the worst-case hardness of problems on ideal lattices that is believed to be secure against the quantum computer.

Remark 2. Recently, Castryck et al. [28] showed provably weak instances of ring-LWE. But these kinds of weak instances do not affect our scheme.

C. Correctness of SwHE Scheme

The correctness of our scheme depends on how the decryption can recover the original result from the ciphertext after some homomorphic operations. We can write the decryption process as follows.

$$\begin{cases} \mathrm{Dec}(ct_{add}, sk) = \mathrm{Dec}((ct \boxplus ct'), sk) = m + m' \\ \mathrm{Dec}(ct_{mul}, sk) = \mathrm{Dec}((ct \boxtimes ct'), sk) = m \cdot m' \end{cases} \tag{8}$$

Actually, the above process is already described in Section 1.1 in [15]. Here, ciphertext ct and ct' comes from $m \in R_q$ and $m' \in R_q$ respectively after encryption. The encryption scheme in Section V-A is the presentation of SwHE and its holds if the following lemma holds as shown in [27].

Lemma 1 (Condition for successful decryption). For a ciphertext ct, the decryption $\mathrm{Dec}(ct, sk)$ recovers the correct result if $\langle ct, s\rangle \in R_q$ does not wrap around mod q, namely, if the condition $\|\langle ct, s\rangle\|_\infty < \frac{q}{2}$ is satisfied where $\|a\|_\infty = \max |a_i|$ for an element $a = \sum_{i=0}^{n-1} a_i x^i \in R_q$. Specifically, for a fresh ciphertext ct, the ∞-norm $\|\langle ct, s\rangle\|_\infty$ is given by $\|m + t(ue + g - sf)\|_\infty$. Moreover, for a homomorphically operated ciphertext, the ∞-norm can be computed by (7).

VI. PACKING METHOD

Packing method is the process of encoding many bits in a single polynomial. Lauter et al. [24] used a packing method for efficient encoding of an integer in a polynomial ring to facilitate arithmetic operations (see Section 4.1 in [24] for details). For example, consider a binary vector $M = (10101011)$ with $l = 8$, it is encoded as $\mathrm{Poly}(M) = 1 + x + x^3 + x^5 + x^7$ using the packing method in [24]. Here we first review some packing methods and also discuss our packing method in the following subsections.

A. Review of Packing Method

Yasuda et al. [27] modified packing method of Lauter et al. [24] to perform secure inner product for computing the Euclidean and Hamming distances. Thereafter, Yasuda et al. [25] used the packing method in [27] to do privacy-preserving wildcards pattern matching between the text and single pattern using few polynomial multiplications. Here we like to compute the Hamming distance c as in Eq. (1) between two l-bit integer vectors $A = (a_0, \ldots, a_{l-1}) \in R_t$ and $B = (b_0, \ldots, b_{l-1}) \in R_t$. Therefore, we need two packing method as defined by Yasuda et al. [27]. Moreover, for the ease of computation of the Hamming distance c as in Eq. (1), we slightly modify packing method of [27] in the ring $R = \mathbb{Z}[x]/(x^n + 1)$ for the integer vectors A and B with $l \leq n$ as

$$\begin{cases} Poly_1(A) = \sum_{i=0}^{l-1} a_i x^i \\ Poly_2(B) = \sum_{j=0}^{l-1} b_j x^{l-(j+1)}\,. \end{cases} \tag{9}$$

Inner product property. Consider two same integer vectors A and B of length l. We already know that inner product of two vectors helps the Hamming distance computation. So the polynomial multiplications of $Poly_1(A)$ and $Poly_2(B)$ in the base ring R can be represented as

$$\begin{aligned} & \left(\sum_{i=0}^{l-1} a_i x^i\right) \times \left(\sum_{j=0}^{l-1} b_j x^{l-(j+1)}\right) \\ = \;& \sum_{i=0}^{l-1}\sum_{j=0}^{l-1} a_i b_j x^{i+l-(j+1)} \\ = \;& \sum_{i=0}^{l-1} a_i b_i x^{l-1} + \mathrm{ToHD} + \mathrm{ToLD} \\ = \;& \langle A, B\rangle x^{l-1} + \mathrm{ToHD} + \mathrm{ToLD}\,. \end{aligned} \tag{10}$$

Here, $\langle A, B\rangle$ define the inner product of two vectors A and B. Moreover, the ToHD (terms of higher degree) means $deg(x) > l - 1$ and the ToLD (terms of lower degrees) means $deg(x) < l - 1$. The result in Eq. (10) shows that one polynomial multiplication includes the inner products of $\langle A, B\rangle$. In addition, the following Proposition is needed to hold for computing the inner product over packed ciphertexts.

Here the packed ciphertexts for $Poly_i(A) \in R$ are defined for some $i = 1, 2, 3$ as

$$ct_i(A) = \mathrm{Enc}(Poly_i(A), pk) \in (R_q)^2\,. \tag{11}$$

Proposition 1. Let $A = (a_0, \ldots, a_{l-1}) \in R_t$ and $B = (b_0, \ldots, b_{l-1}) \in R_t$ be two integer vectors of length l. If the ciphertext of A and B can be represented as $ct_1(A)$ and $ct_2(B)$ respectively by Eq. (11) then under the condition of Lemma 1, decryption of homomorphic multiplication $ct_1(A) \boxtimes ct_2(B) \in (R_q)^2$ will produce a polynomial of R_t with x^{l-1} including coefficient $\langle A, B\rangle = \sum_{i=0}^{l-1} a_i b_i \mod t$. Alternatively, we can say that homomorphic multiplication of $ct_1(A)$ and $ct_2(B)$ simultaneously computes the inner product $\langle A, B\rangle$ for $0 \leq i \leq (l-1)$.

B. Our Packing Method

We want to compute the multiple Hamming distance d_m as in Eq. (2) using few polynomial multiplications. To understand the multiple Hamming distance, let us form an integer vector $A = (a_0, \cdots, a_{l-1})$ from a binary the vector $a = (a_1, \cdots, a_l)$ of length l. Moreover, let us consider the binary vectors $b_m = (b_{m,1}, \cdots, b_{m,l})$ where $1 \le m \le k$. Then we form another integer vector $P = (b_{1,0} \cdots b_{1,l-1}, \ldots, b_{k,0} \cdots b_{k,l-1})$ by taking each vector b_m where the length of each sub-vector of P is l. . Here the multiple Hamming distance means the distances between the vector A and each sub-vector in P. So we need to define another packing method than that in [27].

Consider two same integer vectors $A = (a_0, \cdots, a_{l-1}) \in R_t$ of length l and $P = (b_{1,0} \cdots b_{1,l-1}, \ldots, b_{k,0} \cdots b_{k,l-1}) \in R_t$ of length $k \cdot l$. Here we need to find the Hamming distances between $A = (a_0, \ldots, a_{l-1})$ and $P_m = (b_{m,0}, \ldots, b_{m,l-1})$ with $1 \le m \le k$. Moreover, we know from [27] that the secure inner product $\langle A, P_m \rangle$ helps to compute the Hamming distance between A and P_m. Here we pack these integer vectors by some polynomials with the highest degree$(x) = n$ in such a way so that inner product $\langle A, P_m \rangle$ does not wrap-around a coefficient of x with any degrees. For the integer vectors A and P with $n \ge k \cdot l$ and $1 \le m \le k$, the packing method of [27] in the same ring $R = \mathbb{Z}[x]/(x^n + 1)$ can be rewritten as

$$\begin{cases} Poly_1(A) = \sum_{i=0}^{l-1} a_i x^i \\ Poly_3(P) = \sum_{m=1}^{k} \sum_{j=0}^{l-1} b_{m,j} x^{l \cdot m - (j+1)} . \end{cases} \tag{12}$$

Here the multiplication of the above two polynomials helps the inner product computations which in turn helps the multiple Hamming distances computation between A and P_m. Here each Hamming distance can be found as a coefficient of x with different degrees.

Inner product property. Consider the above two vectors A and P again. We already know that inner product of two vectors helps the Hamming distance computation. So the polynomial multiplications of $Poly_1(A)$ and $Poly_3(P)$ in the same base ring R can be represented as

$$\begin{aligned} & \left(\sum_{i=0}^{l-1} a_i x^i \right) \times \left(\sum_{m=1}^{k} \sum_{j=0}^{l-1} b_{m,j} x^{l \cdot m - (j+1)} \right) \\ = \quad & \sum_{m=1}^{k} \sum_{i=0}^{l-1} \sum_{j=0}^{l-1} a_i b_{m,j} x^{i + l \cdot m - (j+1)} \\ = \quad & \sum_{m=1}^{k} \sum_{i=0}^{l-1} a_i b_{m,i} x^{l \cdot m - 1} + \text{ToHD} + \text{ToLD} \\ = \quad & \sum_{m=1}^{k} \langle A, P_m \rangle x^{l \cdot m - 1} + \text{ToHD} + \text{ToLD} . \end{aligned} \tag{13}$$

Here, A is the vector of length l and P_m is the m-th sub-vector of B with $1 \le m \le k$. Moreover, the ToHD (terms of higher degree) means $deg(x) > l \cdot m - 1$ and the ToLD (terms of lower degrees) means $deg(x) < l \cdot m - 1$. The result in Eq. (13) shows that one polynomial multiplication includes the multiple inner products of $\langle A, P_m \rangle$. In addition, the following Proposition is needed to hold for computing the multiple inner products over packed ciphertexts.

Proposition 2. Let $A = (a_0, a_1, \ldots, a_{l-1}) \in R_t$ be an integer vector where $|A| = l$ and $P = (b_{1,0} \cdots b_{1,l-1}, \ldots, b_{k,0} \cdots b_{k,l-1}) \in R_t$ be another integer vector of length $k \cdot l$. For $1 \le m \le k$, the vector P includes k sub-vectors where the length of each sub-vector is l. If the ciphertext of A and P can be represented as $ct_1(A)$ and $ct_3(P)$ respectively by Eq. (11) then under the condition of Lemma 1, decryption of homomorphic multiplication $ct_1(A) \boxtimes ct_3(P) \in (R_q)^2$ will produce a polynomial of R_t with $x^{l \cdot m - 1}$ including coefficient $\langle A, P_m \rangle = \sum_{i=0}^{l-1} a_i b_{m,i} \mod t$. Alternatively, we can say that homomorphic multiplication of $ct_1(A)$ and $ct_2(P)$ simultaneously computes the multiple inner products for $1 \le m \le k$ and $0 \le i \le (l-1)$.

VII. SECURE COMPUTATIONS OF OUR PROTOCOLS

We discuss secure computation of the PET and PriBET protocols (see Section IV for details) in the following subsections.

A. The PET Protocol

We can do the computation of the private equality test (PET) protocol using the SwHE scheme in Section V and the packing method in Section VI-A. In addition, according to Eq. (1), we need to find out the values of the Hamming distance c. Let us consider two l-bit integer $a = (a_1, \ldots, a_l)$ and $b = (b_1, \ldots, b_l)$. From these integers, we form two integer vectors as $A = (a_0, a_1, \ldots, a_{l-1}) \in R_t$ and $B = (b_0, b_1, \ldots, b_{l-1}) \in R_t$ from which c can be computed. Here, c is computed by the Hamming distance between A and B using the arithmetic computation as

$$c = \sum_{i=0}^{l-1} (a_i + b_i - 2 a_i b_i) . \tag{14}$$

Computation over packed ciphertext. The packed ciphertext is defined by an encrypted polynomial where the polynomial is generated from an integral vector using some packing methods. For the above two integer vectors A and P, the Hamming distance c in Eq. (14) can be computed by the packing method in Eq. (9) and inner product property in Eq. (10). Moreover, the packed ciphertext of the vectors A and B is computed by the Eq. (11). So c is computed from Proposition 1 and the packed ciphertext vector $ct_1(A) \in R_q$ and $ct_2(B) \in R_q$ in three homomorphic multiplications and two homomorphic additions as ct_{r_1} equals

$$ct_1(A) \boxtimes ct_2(V_l) \boxplus ct_2(B) \boxtimes ct_1(V_l) \boxplus (-2 ct_1(A) \boxtimes ct_2(B))$$

where V_l denotes another integer vector $(1, \ldots, 1)$ of length l. The above encrypted polynomial ct_{r_1} includes many Hamming distances between the sub-vectors of A and sub-vectors of B. Here we need the Hamming distance c in Eq.

(14). Bob sends ct_{r_1} to Alice for decryption. According to Proposition 1 and our PET protocol, Alice decrypts ct_{r_1} in the ring R_q using her secret key and extracts c as a coefficient of x^{l-1} from the plaintext of ct_{r_1}. Then Alice checks whether $c = 0$ or not to help Bob to decide either $a = b$ or $a \neq b$.

B. The PriBET Protocol

We can do the computation of private batch equality test (PriBET) protocol using the SwHE scheme in Section V and the packing method in Section VI-B. In addition, according to Eq. (2), we need to find out the values of the multiple Hamming distance d_m. Let us consider two l-bit integers $a = (a_1, \ldots, a_l)$ and $b_m = (b_{m,1}, \ldots, b_{m,l})$. From these integers, we form two integer vectors as $A = (a_0, \ldots, a_{l-1}) \in R_t$ and $P = (b_{1,0} \cdots b_{1,l-1}, \ldots, b_{k,0} \cdots b_{k,l-1}) \in R_t$ from which d_m can be computed. Here, for $1 \leq m \leq k$, d_m is computed by the multiple Hamming distance between A and P_m using the arithmetic computation as

$$d_m = \sum_{i=0}^{l-1} (a_i + b_{m,i} - 2a_i b_{m,i}) \,. \tag{15}$$

Computation over packed ciphertext. For the above two integer vectors A and P, the multiple Hamming distance d_m in Eq. (15) can be computed by the packing method in Eq. (12) and inner product property in Eq. (13). Moreover, the packed ciphertext of the vectors A and P is computed by the Eq. (11). So d_m is computed from Proposition 2 and the packed ciphertext vector $ct_1(A) \in R_q$ and $ct_3(P) \in R_q$ in three homomorphic multiplications and two homomorphic additions as ct_{r_2} equals

$$ct_1(A) \boxtimes ct_3(V_\alpha) \boxplus ct_3(P) \boxtimes ct_1(V_l) \boxplus (-2ct_1(A) \boxtimes ct_3(P))$$

where V_α denotes an integer vector like $(1, \ldots, 1)$ of length $k \cdot l$ and V_l denotes another integer vector $(1, \ldots, 1)$ of length l. The above encrypted polynomial ct_{r_2} includes many Hamming distances between the sub-vectors of A and sub-vectors of P_m. Here we need the Hamming distance d_m in Eq. (15). Bob sends ct_{r_2} to Alice for decryption. According to Proposition 1 and our PriBET protocol, Alice decrypts ct_{r_2} in the ring R_q using her secret key and extracts d_m as a coefficient of $x^{l \cdot m-1}$ from the plaintext of ct_{r_2}. Then Alice checks whether at least of one of the d_m contains 0 or not to help Bob to decide either equality or non-equality.

VIII. PERFORMANCE EVALUATION

In this section, we experimented our PET and PriBET protocols and measured their performance. So we discuss used parameters of our experiments and their security levels. Here, we also discuss the performance of our protocols using ring-LWE SwHE.

A. Chosen Parameters and Security Level

As discussed in Section V, we need to consider appropriate values of the parameters (n, q, t, σ) for successful decryption and to achieve a certain security level. Here, we need the lattice dimension to be greater than $n \geq l$ and $n \geq k \cdot l$ for our PET and PriBET protocols respectively as mentioned in Section VI. Here we compare two integers of length $l = 8 \sim 2048$-bit in the PET protocol. For this reason, we choose the lattice dimension $n = 2048$. We also consider $l = 2049 \sim 4096$-bit with lattice dimension $n = 4096$ for integers comparison in the PET protocol. In addition, we can do n/l comparisons in one computation using the PriBET protocol where the lengths of integers l are 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit. Furthermore, we set $t = 2048$ for our plaintext space R_t. According to the work in [24], we choose $\sigma = 8$ and value of q must be greater than $16n^2t^2\sigma^4 = 2^4 \cdot 2^{22} \cdot 2^{22} \cdot 2^{12} = 2^{60}$ for the ciphertext space R_q. Therefore, we fix our parameters as $(n, q, t, \sigma) = (2048, 61\,\text{bits}, 2048, 8)$. For our PriBET protocol, for lattice dimension $n = 2048$, we set the value of the block size k as 256, 128, 64, 32, and 16 for the integer size 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit respectively. In addition, for lattice dimension $n = 4096$, we also set the value of the block size k as 512, 256, 128, 64, and 32 for the integer size 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit respectively. As shown in Table 2 in [26], our parameters setting provides 140-bit security level to protect the security algorithm from some distinguishing attacks. In addition, NIST [29] defines various security levels for different security algorithms and their corresponding validity periods. They declare that a minimum strength of 112-bit level security has a security lifetime up to 2030. They also declare that a security algorithm with a minimum strength of 128-bit level security has a security lifetime beyond 2030.

Remark 3. For the PET protocol in Section IV, integers comparison of $8 \sim 2048$-bit is enough for practical consideration. We also considered the comparison for $2049 \sim 4096$-bit with lattice dimension $n = 4096$ in the PET protocol to show the performance of any two binary string comparison of $2049 \sim 4096$-bit.

Table I
PERFORMANCE OF THE PET PROTOCOL

Integer size (bits)	Lattice dimension (n)	q	Security level	Total time (milliseconds)
$8 \sim 2048$	2048	61 bits	140	93
$2049 \sim 4096$	4096	61 bits	391	171

Table II
PERFORMANCE OF THE PRIBET PROTOCOL

Integer size (bits)	Block size (k)	No. of comparison/sec.	Lattice dimension (n)	q	Security level
8	256	2348	2048	61 bits	140
16	128	1174			
32	64	587			
64	32	293			
128	16	146			
8	512	2994	4096	61 bits	391
16	256	1497			
32	128	748			
64	64	374			
128	32	187			

B. Implementation Results

Here Table I and Table II show the performances of our PET and PriBET protocols respectively. Here, we implemented our protocols in C programming language with Pari C library (version 2.7.5) [30] and ran on a single machine configured with one 3.6 GHz Intel core-i7 CPU and 8 GB RAM in Linux environment. Our PET protocol took only 107 ms (milliseconds) and 171 ms for comparing any two integers of $8 \sim 2048$-bit and $2049 \sim 4096$-bit respectively as shown in Table I. On the other hand, Table II shows the performance of our PriBET protocol for lattice dimension 2048 and 4096. For lattice dimension 2048, our PriBET protocol was able to do about 2348 (resp., 1174) equality comparisons per second for the small integer size of 8-bit (resp., 16-bit). In addition, it also did about 587 (resp., 293) equality comparisons per second for practical integer size of 32-bit (resp., 64-bit) integers as shown in Table II. Finally, our PriBET protocol did about 146 comparisons per second for the large integers of 128-bit. Again for lattice dimension 4096 and the integer size 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit, our PriBET protocol was able to do about 2994 comparisons per second, 1497 comparisons per second, 748 comparisons per second, 374 comparisons per second, and 187 comparisons per second with the block size as 512, 256, 128, 64, and 32 respectively. Here, the PriBET protocol showed a better performance for the lattice dimension 4096. As discussed in [26], we achieve 140-bit security level for our both protocols for the lattice dimension 2048. Furthermore, our PriBET protocol achieves 391-bit security level for the lattice dimension 4096. In addition, our PET and PriBET protocols require a communication complexity of $\mathcal{O}(l \log q)$ and $\mathcal{O}(k \cdot l \log q)$ respectively.

Remark 4. Our experiments show that our protocols are practically usable. Our PriBET protocol is also able to handle large integers with a large block size if we would increase the lattice dimension n.

IX. CONCLUSION

Throughout this article, we discussed our PET and PriBET protocols using ring-LWE based SwHE scheme for doing private equality tests of integers in the semi-honest model. Here we also presented our experimental results to show the real world scenarios. Our PET protocol is able to do any comparison between $8 \sim 2048$-bit of two integers within 107 ms. So the PET protocol is useful for large string matching computations. Furthermore, our PriBET protocol was able to perform acceptable numbers of equality comparisons per second for $8 \sim 128$-bit integers. Moreover, we can conclude from our experiments in PriBET protocol that if bit size of integers increases (resp., decreases) then the number of comparison per second decreases (resp., increases). We hope that our PriBET protocol is a fruitful solution using SwHE scheme to address different types private equalities queries in databases. Note that our code is not fully optimized; Therefore, we hope that the optimized code would produce better results.

ACKNOWLEDGMENT

This research is supported by KAKENHI Grant Numbers JP26540002, JP24106008, and JP16H0175.

REFERENCES

[1] Saha, T.K. Saha and A.B.M. Ali, “Storage cost minimizing in cloud - a proposed novel approach based on multiple key cryptography”, in 1st Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE), IEEE, 2014, pp. 1–9.

[2] R. Fagin, M. Naor, and P. Winkler, “Comparing information without leaking it,” Communications of the ACM, vol. 39, no. 5, pp.77-85, 1996.

[3] M. Jakobsson and M. Yung, “Proving without knowing: On oblivious, agnostic and blindfolded provers”, in Advances in Cryptology-CRYPTO. Springer, 1996, pp. 186-200.

[4] G. Couteau, “Efficient secure comparison protocols”, Cryptology ePrint Archive, Report 2016/544, 2016, http://eprint.iacr.org/2016/544.

[5] R. L. Rivest, L. Adleman, and M. Dertouzos, “On data banks and privacy homomorphisms,” in Foundations of Secure Computation, R. DeMillo, Ed. et al. London, U.K.: Academic, 1978, pp. 169-179.

[6] S. Goldwasser and S. Micali, “Probabilistic encryption & how to play mental poker keeping secret all partial information,” in Proc. ACM Symp. Theory Comput., San Francisco, CA, USA, 1982, pp. 365-377.

[7] J. D. Cohen and M. J. Fischer, “A robust and verifiable cryptographically secure election scheme,” in 26th Annual Symposium on Foundations of Computer Science. Portland, OR: IEEE, 1985, pp. 372-382.

[8] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proc. Eurocrypt: Advances in Cryptology, ser. Lect. Notes Comput. Sci. Berlin, Germany: Springer-Verlag, 1999, vol. 1592, pp. 223-238.

[9] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120-126, February 1978.

[10] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in Advances in Cryptology. Springer Berlin Heidelberg, 1985, vol. 196, pp. 10-18.

[11] D. Boneh, E. Goh and K. Nissim, “Evaluating 2-DNF formulas on ciphertexts”, in Proceedings of Theory of Cryptography (TCC), Springer, 2005, pp 325–341.

[12] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, 2009, pp. 169-178.

[13] Y. Hu, “Improving the efficiency of homomorphic encryption schemes,” PhD diss., May 2013, Worcester Polytechnic Institute, Massachusetts.

[14] V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and learning with errors over rings”, in proceeding of Advances in Cryptology–EUROCRYPT. Springer, 2010, vol. 6110, pp 1-23,

[15] Z. Brakerski and V. Vaikuntanathan, “Fully homomorphic encryption from ring-LWE and security for key dependent messages,” in CRYPTO, ser. Lect. Notes Comput. Sci. Berlin, Germany: Springer-Verlag, 2011, vol. 6841, pp. 505–524.

[16] R. Li and C. K. Wu, “Co-operative private equality test”, I. J. of Network Security, vol.1, no. 3, pp. 149-153, 2005.

[17] S. F. Ciou and R. Tso, “A privacy preserved two-party equality testing protocol,” in Fifth International Conference on Genetic and Evolutionary Computing (ICGEC), IEEE, August:2011, pp. 220-223.

[18] N. K. Ardestani, “Efficient Non-Interactive Secure Two-Party Computation for Equality and Comparison,” PhD diss., University of Calgary, 2015.

[19] T. Mitsunaga, Y. Manabe, and T. Okamoto, “Efficient secure auction protocols based on the Boneh-Goh-Nissim encryption”. In Proceedings of the 5th international conference on Advances in information and computer security, Springer-Verlag, November:2010, pp. 149-163.

[20] S. Jha, L. Kruger, and V. Shmatikov, “Towards practical privacy for genomic computation,” in Proc. 29th IEEE Symp. on Secur. Privacy, May 2008. pp. 216–230.

[21] J. H. Cheon, M. Kim, and K. Lauter, “Homomorphic computation of edit distance,” in proceeding of Financial Cryptography and Data Security. Berlin Heidelberg: Springer, 2015, pp. 194–212.

[22] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, “Machine learning classification over encrypted data,” Cryptology ePrint Archive, Report 2014/331, 2014, http://eprint.iacr.org/2014/331.

[23] Y. Lindell and B. Pinkas, “Secure multiparty computation for privacy-preserving data mining,” Journal of Privacy and Confidentiality, vol. 1 no. 1, pp. 59–98, Pennsylvania, 2009.

[24] K. Lauter, M. Naehrig, and V. Vaikuntanathan, “Can homomorphic encryption be practical?,” in Proceedings of ACM Cloud Computing Security Workshop (CCSW), ACM Press, 2011, pp. 113–124.

[25] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, “Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption,” in W. Susilo and Y. Mu (Eds.): ACISP 2014, LNCS, Springer, 2014, vol. 5844, pp. 338–353.

[26] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, “Secure pattern matching using somewhat homomorphic encryption”, in Proceedings of ACM workshop on Cloud Computing Security Workshop. ACM Press, 2013, pp. 65–76.

[27] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, “Practical packing method in somewhat homomorphic encryption,” in DPM/SETOP, ser. LNCS, Springer, 2013, vol. 8147, pp. 34-50.

[28] W. Castryck, I. Iliashenko, and F. Vercauteren, “Provably weak instances of ring-LWE revisited,” in Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2016, pp. 147-167.

[29] E. Barker, “Recommendation for key management,” NIST Special Publication 800-57 Part 1 Rev. 4, NIST, 2016.

[30] The PARI∼Group, PARI/GP version `2.7.5`, Bordeaux, 2014, http://pari.math.u-bordeaux.fr/