Efficient Protocols for Private Database Queries

Tushar Kanti Saha[1(✉)], Mayank[2], and Takeshi Koshiba[3]

[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] Department of Computer Science and Engineering,
Indian Institute of Technology (Banaras Hindu University), Varanasi, India
mayank.cse14@iitbhu.ac.in
[3] Faculty of Education and Integrated Arts and Sciences,
Waseda University, Tokyo, Japan
tkoshiba@waseda.jp

Abstract. We consider the problem of processing private database queries over encrypted data in the cloud. To do this, we propose a protocol for conjunctive query and another for disjunctive query processing using somewhat homomorphic encryption in the semi-honest model. In 2016, Kim et al. [IEEE Trans. on Dependable and Secure Comput.] showed an FHE-based query processing with equality conditions over encrypted data. We improve the performance of processing private conjunctive and disjunctive queries with the low-depth equality circuits than Kim et al.'s circuits. To get the low-depth circuits, we modify the packing methods of Saha and Koshiba [APWConCSE 2016] to support an efficient batch computation for our protocols with a few multiplications. Our implementation shows that our protocols work faster than Kim et al.'s protocols for both conjunctive and disjunctive query processing along with a better security level. We are also able to provide security to both attributes and values appeared in the predicate of the conjunctive and disjunctive queries whereas Kim et al. provided the security to the values only.

Keywords: Private Database Queries · Conjunctive · Disjunctive · Packing method · Homomorphic Encryption · Batch technique

1 Introduction

Private database queries (PDQ) plays an important role for accessing these data securely from any part of the world. In addition, users are not interested to disclose their queries and results to the database owners or any other parties. At the same time, database owners are not interested to disclose their whole database to their users. Besides, they do not like to keep their data in their personal computer or server because of high maintenance cost. They are now interested in storing their data to another party like the cloud so that database owners and

G. Livraga and S. Zhu (Eds.): DBSec 2017, LNCS 10359, pp. 337–348, 2017.
DOI: 10.1007/978-3-319-61176-1_19

their allowed users can access the data from anywhere in the world with a low cost. However, they also want to secure their data at the same time. Moreover, database owners want to secure their data using the encryption method of a cryptographic scheme. But the encrypted data needs to be decrypted by some trusted parties before utilizing it for some purposes which raises another security problem. In reality, it is hard to find such trusted parties. So it is desirable to execute some queries on encrypted data without decryption. On the contrary, homomorphic encryption (HE) is the encryption scheme which allows the meaningful operation like addition and multiplication on encrypted data without decryption. Therefore, we use homomorphic encryption scheme in case of private database queries. The concept of privacy homomorphism was coined by Rivest et al. in 1978 [10]. Also, the role of homomorphic encryption was limited to either addition or multiplication before introducing Gentry's revolutionary work in 2009 [6]. Moreover, the homomorphic encryption scheme can be classified into three types. Firstly, partial homomorphic encryption (PHE) allows either addition or multiplication but not both. Secondly, somewhat homomorphic encryption (SwHE) allows many additions and few multiplications. Finally, fully homomorphic encryption (FHE) allows any number of additions and multiplications. Here Gentry proposed the fully homomorphic encryption scheme by applying bootstrapping technique into somewhat homomorphic encryption scheme. But fully homomorphic encryption scheme is far behind from practical implementation due to its speed [11]. Therefore, we use somewhat homomorphic encryption scheme [9] which is faster than FHE due to supporting a limited number of multiplications. In this paper, we consider the security of the attributes and values appeared in the predicate of a conjunctive or disjunctive query with equality conditions. An example of the conjunctive query is that a managing staff of a hospital is trying to find out the patient's information from a hospital database who are suffering from 'Leukemia' and age is less than 30. In addition, a doctor is searching for the patients who are suffering from fever or cold, which is an example of the disjunctive query. In 2016, Kim et al. [8] showed an approach to address private database queries like conjunctive, disjunctive, and threshold queries using leveled FHE [3] with SIMD techniques. They also showed its implementation in [7] which took about 26.54 s to perform a query on 326 elements (0.08 sec./per record) including 11 attributes of 40-bit values with the 93-bit security level. But this speed of processing query is not satisfactory to process big data stored in the cloud. So there should be an efficient method to improve the performances of conjunctive and disjunctive queries with a better security level.

1.1 Reviews of Recent Works

In 2013, Boneh et al. [2] showed an efficient method of processing conjunctive queries only with somewhat homomorphic encryption. But the performance of their scheme is still far from practicality. In 2016, Cheon et al. [5] showed an approach to private query processing on encrypted databases using leveled FHE in [3] with a better security level. But their performances of query processing

was highly time-consuming in a practical sense. They also declared performance improvement challenge for processing the private queries. At the same time, Kim et al. [8] also used BGV scheme in [3] to describe another protocol for processing conjunctive, disjunctive, and threshold queries over encrypted data in the cloud. They showed the practical implementation of the protocol in another paper [7]. It took about 0.11 s to access each record with 11 attributes of 40-bit values. In another paper, Kim et al. [7] showed a better security for processing these same types of queries. Here they provide security to both attributes and values in the predicate of a query. But it took about 0.12 s to access each record with 11 attributes of 40-bit values. Here they used equality circuits of depth $\lceil \log l \rceil$ to compare two l-bit integers which can be improved by the private batch equality protocol in [11]. Besides, none of the above protocols were able to achieve a remarkable efficiency regarding practicality. Recently, Saha and Koshiba [12] showed an efficient protocol than that in [5] for processing a conjunctive query. But their computation technique is only useful for processing a conjunctive query.

1.2 Our Contribution

In this paper, we consider the problem of processing private conjunctive and disjunctive queries with equality conditions over encrypted database. We also think the security both attributes α_i and values v_i with $1 \leq i \leq k$ appeared in the predicate of a conjunctive and disjunctive query. Here we follow the conventional approach of processing conjunctive and disjunctive queries. For example, let us consider the conjunctive and disjunctive queries with k equality conditions as "*select V from Record where* $\alpha_1 = v_1$ *and* $\alpha_2 = v_2$ *and* ... *and* $\alpha_k = v_k$" and "*select V from Record where* $\alpha_1 = v_1$ *or* $\alpha_2 = v_2$ *or* ... *or* $\alpha_k = v_k$" respectively. To process these queries with k equality conditions, the conventional solution is that client needs to send k queries firstly to the database server. After that, database server does the equality matching of attributes and the values of its Record table and sends back the results to the client. Then client needs to do the intersection and union of those k results from the database to get the actual result of conjunctive and disjunctive query respectively. In addition, most of the existing solutions with homomorphic encryption [4,5,7,8] used the equality circuits of depth $\lceil \log l \rceil$ for comparing two l-bit binary values. By developing new batch technique and packing methods, our equality circuit is reduced to a constant-depth circuit which includes many equality comparisons. Then we propose an efficient method to improve the performances of private conjunctive and disjunctive queries using ring learning with errors (RLWE) based SwHE of a better security level.

2 Our Protocols

In this section, we describe the protocols of processing private database queries for conjunctive and disjunctive cases. To address private database query (PDQ), we consider the security of both attributes and values in the predicate of a

conjunctive or disjunctive query. Here we use the same protocol settings as in Kim et al. [7] with a different scenario.

2.1 Attribute Matching

Suppose a medical research institute (MRI) is maintaining its database of some patients in the cloud. Since patient's information are sensitive, MRI has uploaded its database using a public key encryption scheme. Here consider Bob has m encrypted records $\{\mathcal{R}_1, \ldots, \mathcal{R}_m\}$ in its Record table of the MRI's database with λ attributes where $\lambda \geq k$. Here we require only the k attributes and their values from the predicate of a query to process that query. Furthermore, we denote each attribute name with a δ-bit binary vector $\alpha_i = (g_{i,0}, \cdots, g_{i,\delta-1})$ where $g_{i,c}$ is the c-th bit of the i-th attribute with $1 \leq i \leq k$ and $0 \leq c \leq \delta - 1$. Also, we need to consider k attributes among λ attributes in each record required for our conjunctive and disjunctive query processing. We also denote each attribute name in the Record table using a δ-bit binary vector $\beta_j = (h_{j,0}, \cdots, h_{j,\delta-1})$ where $h_{j,e}$ is the e-th bit of the j-th attribute with $1 \leq j \leq \lambda$ and $0 \leq e \leq \delta - 1$. Since we consider the security of attributes, first the protocol matches encrypted attribute α_i with any attribute β_j in the Record table. Here Bob does the matching by computing the Hamming distance $\mathbb{H}_{i,j}$ between α_i and β_j as $\mathbb{H}_{i,j} = |\alpha_i - \beta_j|$. Here if $\mathbb{H}_{i,j} = 0$, then we can say that $\alpha_i = \beta_j$; otherwise $\alpha_i \neq \beta_j$. According to our protocol, α_i must be matched with any β_j for some $1 \leq i \leq k$ and $1 \leq j \leq \lambda$.

2.2 Batch Processing

For our Record table, each record is represented as $\mathcal{R}_\mu = \{w_{\mu,1} \ldots, w_{\mu,\lambda}\}$ where each value $w_{\mu,j} = (b_{\mu,j,0}, \ldots, b_{\mu,j,l-1})$ is considered as a binary vector of the same length l with $1 \leq \mu \leq m$ and $1 \leq j \leq \lambda$. We know that our Record table contains m records. If we want to compute the Hamming distance of each v_i from each $w_{\mu,j}$ one by one then it is more time-consuming. Here we use the batch technique of the private batch equality protocol in [11]. Actually, batch processing is the method of executing a single instruction on multiple data. The performance of our protocols can be increased by using the batch technique within the lattice dimension n. Generally, a big database consists of many tables where each table contains numerous records. For our conjunctive query processing with batch technique, if we compare all the values of a certain attribute of a particular table using a single computation then we will be required higher lattice dimension n which requires more memory to compute. This high requirement of memory may exceed the usual capacity of a machine in the cloud. So we divide all records of a table into blocks. For our given m records, we divide the total records m into p blocks as $p = \lceil m/\eta \rceil$. Here each block consists of η records with λ attributes from which we have to access k attributes. If we access each record of our Record table one after another then it requires $m \cdot k$ rounds communication between Alice and Bob in the cloud for accessing m records. On the contrary, the batch technique allows us to access all values of any attribute β_j at a time. By utilizing the batch

technique, we reduce the communication complexity between Alice and Bob in the cloud from $m \cdot k$ to $\lceil (m \cdot k)/\eta \rceil$. Now we can pack the η values of the β_j-th attribute of each block in a single polynomial to support batch computation where $1 \leq j \leq \lambda$.

2.3 Protocol for Conjunctive Query

A conjunctive query is a query which contains multiple conditions in the predicate of the query connected by 'and'/'∧'. For instance, a research staff is trying to find the information of the patients who suffer from Leukemia and are 30 years old. This is a conjunctive query request to the cloud. In this scenario, consider Alice has a conjunctive query with k conditions in its predicate as "*select V from Record where* $\alpha_1 = v_1$ *and* $\alpha_2 = v_2$ *and* ... *and* $\alpha_k = v_k$". Here we follow the conventional approach of processing a conjunctive query. So it can be computed by intersecting 'IDs' from the result the k sub-queries as $\bigcap_{i=1}^{k} Q(\alpha_i = v_i)$ where $Q(\alpha_i = v_i) = \{\text{ID} \mid \text{the attribute } \alpha_i \text{ of ID takes } v_i \text{ as the value.}\}$ Moreover, the values of k attributes $\{\alpha_1, \ldots, \alpha_k\}$ appeared in the predicate of the query is represented as a set $V = \{v_1, \ldots, v_k\}$ where $v_i = (a_{i,0}, \ldots, a_{i,l-1})$ is considered as a binary vector of length l. Here we consider the security of both attributes α_i and values v_i appeared in the predicate of the query. Firstly, Alice sends the encrypted attributes to find the required column in the Record table that are needed in the conjunctive query computation. Then she sends the encrypted values v_i to Bob to be matched with some $w_{\mu,j}$ using the multiple Hamming distance computation where $1 \leq \mu \leq m$ and $1 \leq j \leq \lambda$. To speed up the computation using batch processing as discussed in Sect. 2.2, let us form a query vector $\mathbf{A}_i = (a_{i,0}, \ldots, a_{i,l-1})$ from the values of the i-th condition of the query where $1 \leq i \leq k$. We also assume the i-th attribute of query condition matches with β_j-th attribute of the Record table where $1 \leq j \leq \lambda$. Again we form another record vector from η values of the attribute β_j of each block σ as $\mathbf{B}_{\sigma,j} = (w_{\sigma,j,1}, \ldots, w_{\sigma,j,\eta})$ where $w_{\sigma,j,d} = (b_{\sigma,j,d,0}, \ldots, b_{\sigma,j,d,l-1})$ with $1 \leq \sigma \leq p$ and $1 \leq d \leq \eta$. Here $|\mathbf{A}_i| = l$ and $|\mathbf{B}_{\sigma,j}| = \eta \cdot l$. Here we find the distance between two vectors $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$ by the multiple Hamming distance computation as

$$\mathbb{H}_{\sigma,d,i} = \sum_{r=1}^{l-1} |a_{i,r} - b_{\sigma,j,d,r}| = \sum_{r=1}^{l-1} (a_{i,r} + b_{\sigma,j,d,r} - 2a_{i,r}b_{\sigma,j,d,r}) \tag{1}$$

where $1 \leq d \leq \eta$, $1 \leq \sigma \leq p$, and $j \in \{1, 2, \ldots, \lambda\}$. Moreover, if $\mathbb{H}_{\sigma,d,i}$ in Eq. (1) is 0 for some position d in the block σ then we can say that $\mathbf{A}_i = \mathbf{B}_{\sigma,j,d}$; otherwise $\mathbf{A}_i \neq \mathbf{B}_{\sigma,j,d}$ where $\mathbf{B}_{\sigma,j,d}$ is the d-th sub-vector of $\mathbf{B}_{\sigma,j,d}$. Here the multiple Hamming distance means the distances between the vector $\mathbf{A}_i$ and each sub-vector in $\mathbf{B}_{\sigma,j}$. So we need to define another packing method than that in [14]. In this way, Alice gets some IDs for each value v_i in the predicate of a query. Then she gets conjunctive query matched IDs after the intersection of all IDs for each v_i. She then sends the IDs to Bob in the cloud again and Bob returns the corresponding records to Alice. Now we explain our protocol for conjunctive query by the following steps.

1. Alice generates the public key and secret key by herself and encrypts each column of the database D' along with attributes. Then she uploads the database to the cloud.
2. Then she also parses both attributes and values from the predicate part of her query. She encrypts attributes α_i and v_i using her public key and sends it to Bob in the cloud.
3. For $1 \leq i \leq k$ and $1 \leq j \leq \lambda$, Bob tries to find out the β_j-th attribute of the Record table that matches α_i using the Hamming distance $\mathbb{H}_{i,j}$ with every attribute in the database. Here Alice helps Bob to find the β_j-th attribute after decryption of the Hamming distance result of Bob.
4. For $1 \leq \sigma \leq p$, Bob does secure computation of batch equality test as in Eq. (1) and sends the encrypted result $\mathbb{H}_{\sigma,d,i}$ to Alice to verify whether at least one of $\mathbb{H}_{\sigma,d,i}$'s is equal to 0.
5. For $1 \leq i \leq k$ and $1 \leq d \leq \eta$, Alice decrypts $\mathbb{H}_{\sigma,d,i}$ using her secret key and checks each value $\mathbb{H}_{\sigma,d,i}$ and gets the IDs for some position d where $\mathbb{H}_{\sigma,d,i} = 0$; In this way, she gets k sets of IDs for k conditions in the query.
6. Then Alice computes the intersection of k sets of IDs and sends the result to Bob to get her desired result.
7. Bob sends the encrypted data to Alice depending on those IDs given by Alice. Then Alice decrypts the data and gets her desired result.

2.4 Protocol for Disjunctive Query

A disjunctive query is a query which contains multiple conditions in the predicate of the query connected by 'or'/'$\vee$'. As discussed in Sect. 2.3, we consider the same database settings for processing a disjunctive query. Let us look at a disjunctive query with k conditions in its predicate as "*select V from Record where* $\alpha_1 = v_1$ *or* $\alpha_2 = v_2$ *or* $\ldots$ *or* $\alpha_k = v_k$". Here we also follow the conventional approach of processing a disjunctive query. Now we can compute by taking union 'IDs' from the result the k sub-queries as $\bigcup_{i=1}^{k} Q(\alpha_i = v_i)$. We process this query with the same multiple Hamming distance computation as in Eq. (1). Our protocol for processing the disjunctive query is same as discussed by 7 steps in Sect. 2.3 except step 6. In case of disjunctive query, Alice needs to compute the union of k sets of IDs instead of intersection (see step 6 in Sect. 2.3) required for conjunctive query protocol.

Remark 1. Here 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. Here we use somewhat homomorphic encryption scheme in [12] and skip its review due to page limitation.

3 Packing Method

In information theory, the method of encoding many bits in a single polynomial is called packing method. In 2011, Lauter et al. [9] used a packing method for

an efficient encoding of an integer in a polynomial ring to facilitate arithmetic operations (see Sect. 4.1 in [9] for details). Here we need the packing methods for both attributes and value matching. Let us consider a binary vector $M = (11001101)$ with $l = 8$ which can be encoded as $Poly(M) = 1 + x^2 + x^3 + x^6 + x^7$ using the packing method in [9]. Here we review and modify the packing methods in Saha et al. [11] which was used in their private batch equality test protocol. Here we skip the discussion of our packing method for attribute matching due to page limitation which is a variant of the following packing method.

3.1 Our Packing Method for Value Matching

First, let us review some parameters in [12]. Let t (resp. q) defines the ring for a message space (resp. ciphertext space) as $R_t = \mathbb{Z}_t[x]/(x^n + 1)$ (resp. $R_q = \mathbb{Z}_q[x]/(x^n + 1)$) which is a ring of integer polynomials of degree less than n with coefficient modulo t (resp. q) (see [12] for details). To accelerate the processing of a conjunctive and disjunctive query, we need to compute the multiple Hamming distance $\mathbb{H}_{\sigma,d,i}$ in Eq. (1) with few polynomial multiplications. As discussed in Sect. 2.3, for $1 \leq i \leq k$ and $j \in \{1, \ldots, \lambda\}$, we consider two same integer vectors $\mathbf{A}_i = (a_{i,0}, \cdots, a_{i,l-1}) \in R_t$ of length l and $\mathbf{B}_{\sigma,j} = (w_{\sigma,j,1}, \ldots, w_{\sigma,j,s}) \in R_t$ where $w_{\sigma,j,d} = (b_{\sigma,j,d,0}, \ldots, b_{\sigma,j,d,l-1})$ of length $\eta \cdot l$ with $1 \leq \sigma \leq p$ and $1 \leq d \leq \eta$. Here we need to find the Hamming distances between $\mathbf{A}_i = (a_{i,0}, \ldots, a_{i,l-1})$ and $\mathbf{B}_{\sigma,j} = (b_{\sigma,j,1,0}, \ldots, b_{\sigma,j,1,l-1}, \ldots, b_{\sigma,j,\eta,0}, \ldots, b_{\sigma,j,\eta,l-1})$. Furthermore, we know from [14] that the secure inner product $\langle \mathbf{A}_i, \mathbf{B}_{\sigma,j} \rangle$ helps to compute the Hamming distance between $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$. Here we pack these integer vectors by some polynomials with the highest degree$(x) = n$ in such a way so that inner product $\langle \mathbf{A}_i, \mathbf{B}_{\sigma,j} \rangle$ does not wrap-around a coefficient of x with any degrees. For the integer vectors $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$ with $n \geq \eta \cdot l$ and $1 \leq d \leq \eta$, the packing method of [11] in the same ring $R = \mathbb{Z}[x]/(x^n + 1)$ can be rewritten as

$$Poly_1(\mathbf{A}_i) = \sum_{c=0}^{l-1} a_{i,c} x^c, Poly_2(\mathbf{B}_{\sigma,j}) = \sum_{d=1}^{s} \sum_{e=0}^{l-1} b_{\sigma,j,d,e} x^{l \cdot d - (e+1)}. \tag{2}$$

Here if we multiply the above two polynomials, it will help us to find the inner product $\langle \mathbf{A}_i, \mathbf{B}_{\sigma,j} \rangle$ which in turn helps the multiple Hamming distances computation between the vectors $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$. Here each Hamming distance can be found as a coefficient of x with different degrees. Now the polynomial multiplications of $Poly_1(\mathbf{A}_i)$ and $Poly_2(\mathbf{B}_{\sigma,j})$ in the same base ring R can be represented as follows.

$$\Big(\sum_{c=0}^{l-1} a_{i,c} x^c\Big) \times \Big(\sum_{d=1}^{s} \sum_{e=0}^{l-1} b_{\sigma,j,d,e} x^{l \cdot d - (e+1)}\Big) = \sum_{d=1}^{s} \sum_{c=0}^{l-1} \sum_{e=0}^{l-1} a_{i,c} b_{\sigma,j,d,e} x^{c + l \cdot d - (e+1)}$$

$$= \sum_{d=1}^{s} \sum_{c=0}^{l-1} a_{i,c} b_{\sigma,j,d,c} x^{l \cdot d - 1} + \text{ToHD} + \text{ToLD} = \sum_{d=1}^{s} \langle \mathbf{A}_i, \mathbf{B}_{\sigma,j,d} \rangle x^{l \cdot d - 1} + \cdots \tag{3}$$

Here, $\mathbf{A}_i$ is the i-th vector of length l that appeared in the predicate of a conjunctive or disjunctive query where $1 \leq i \leq k$. Also, $\mathbf{B}_{\sigma,j,d}$ is the d-th

sub-vector of $\mathbf{B}_{\sigma,j}$ of the block σ and β_j attribute of the Record table with $1 \leq \sigma \leq p$, $1 \leq d \leq \eta$ and $j \in \{1, \ldots, \lambda\}$. Moreover, the ToHD (terms of higher degree) means $deg(x) > l \cdot d - 1$ and the ToLD (terms of lower degrees) means $deg(x) < l \cdot d - 1$. The result in Eq. (3) shows that one polynomial multiplication includes the multiple inner products of $\langle \mathbf{A}_i, \mathbf{B}_{\sigma,j,d} \rangle$. According to the SwHE in Sect. 2 of [12], the packed ciphertexts for $Poly_\tau(A) \in R$ are defined for some $\tau \in \{1, 2\}$ as

$$ct_\tau(A) = \text{Enc}(Poly_\tau(A), pk) \in (R_q)^2 . \tag{4}$$

Proposition 1. *Let $\mathbf{A}_i = (a_{i,0}, \cdots, a_{i,l-1})$ be an integer vector where $|\mathbf{A}_i| = l$ and $\mathbf{B}_{\sigma,j} = (b_{\sigma,j,1,0}, \ldots, b_{\sigma,j,1,l-1}, \ldots, b_{\sigma,j,\eta,0}, \ldots, b_{\sigma,j,\eta,l-1})$ be another integer vector of length $\eta \cdot l$. For $1 \leq d \leq \eta$, the vector $\mathbf{B}_{\sigma,j}$ includes η sub-vectors where the length of each sub-vector is l. If the ciphertext of $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$ can be represented as $ct_1(\mathbf{A}_i)$ and $ct_2(\mathbf{B}_{\sigma,j})$ respectively by Eq. (4) then under the condition of Lemma 1 (see Sect. 2.3 in [12] for details), decryption of homomorphic multiplication $ct_1(\mathbf{A}_i) \boxtimes ct_2(\mathbf{B}_{\sigma,j}) \in (R_q)^2$ will produce a polynomial of R_t with $x^{l \cdot d-1}$ including coefficient $\langle \mathbf{A}_i, \mathbf{B}_{\sigma,j,d} \rangle = \sum_{d=1}^{s} \sum_{c=0}^{l-1} a_{i,c} b_{\sigma,j,d,e} x^{l \cdot d-1}$ mod t. Alternatively, we can say that homomorphic multiplication of $ct_1(\mathbf{A}_i)$ and $ct_2(\mathbf{B}_{\sigma,j})$ simultaneously computes the multiple inner products for $1 \leq i \leq k$, $1 \leq \sigma \leq p$, $1 \leq d \leq \eta$, $0 \leq c \leq (l-1)$, and $j \in \{1, \ldots, \lambda\}$.*

4 Secure Computation Procedure

We need to securely compute both attribute and value matching as discussed in our protocol in Sect. 2.3. Now we present the matching technique of both attributes and values of a conjunctive and disjunctive query with the Record table in the following sub-sections. Due to page limitation, we skip the discussion of secure computation procedure of attribute matching (similar to the following secure computation of value matching).

4.1 Matching the Values in the Record

Now we compute our protocol using the SwHE scheme in [12] and the packing method in Sect. 3.1 for matching the records. In addition, according to Eq. (1), we need to find out the values of the multiple Hamming distance $\mathbb{H}_{\sigma,d,i}$. As discussed in Sect. 3.1, we consider two same integer vectors $\mathbf{A}_i = (a_{i,0}, \cdots, a_{i,l-1}) \in R_t$ and $\mathbf{B}_{\sigma,j} = (b_{\sigma,j,1,0}, \ldots, b_{\sigma,j,1,l-1}, \ldots, b_{\sigma,j,\eta,0}, \ldots, b_{\sigma,j,\eta,l-1}) \in R_t$ from which $\mathbb{H}_{\sigma,d,i}$ can be computed. Here, for $1 \leq d \leq \eta$, $\mathbb{H}_{\sigma,d,i}$ is computed by the multiple Hamming distance between $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$ using Eq. (1). For these two integer vectors $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$, the multiple Hamming distance $\mathbb{H}_{\sigma,d,i}$ in Eq. (1) can be computed by the packing method in Eq. (2) and inner product property in Eq. (3). Moreover, the packed ciphertext of the vectors $\mathbf{A}_i$ and $\mathbf{B}_{\sigma,j}$ is computed by the Eq. (4). So $\mathbb{H}_{\sigma,d,i}$ is computed from Proposition 1 and the packed ciphertext vector $ct_1(\mathbf{A}_i)$ and $ct_2(\mathbf{B}_{\sigma,j})$ in three homomorphic multiplications and two homomorphic additions as $ct(\mathbb{H}_{\sigma,d,i})$ equals

$$ct_1(\mathbf{A}_i) \boxtimes ct_2(V_2) \boxplus ct_2(\mathbf{B}_{\sigma,j}) \boxtimes ct_1(V_1) \boxplus (-2ct_1(\mathbf{A}_i) \boxtimes ct_2(\mathbf{B}_{\sigma,j})) \tag{5}$$

where V_1 denotes an integer vector like $(1, \ldots, 1)$ of length l and V_2 denotes another integer vector like $(1, \ldots, 1)$ of length $\eta \cdot l$. The above encrypted polynomial $ct(\mathbb{H}_{\sigma,d,i})$ includes many Hamming distances between the sub-vectors of $\mathbf{A}_i$ and sub-vectors of $\mathbf{B}_{\sigma,j}$. Here we need the Hamming distance $\mathbb{H}_{\sigma,d,i}$ in Eq. (1). Bob sends $ct(\mathbb{H}_{\sigma,d,i})$ to Alice for decryption. According to Proposition 1 and our protocols, Alice decrypts $ct(\mathbb{H}_{\sigma,d,i})$ in the ring R_q using her secret key and extracts $\mathbb{H}_{\sigma,d,i}$ as a coefficient of $x^{l \cdot d-1}$ from the plaintext of $ct(\mathbb{H}_{\sigma,d,i})$. Then Alice checks whether at least one of the $\mathbb{H}_{\sigma,d,i}$ contains 0 or not to decide whether $\mathbf{A}_i = \mathbf{B}_{\sigma,j,d}$ or $\mathbf{A}_i \neq \mathbf{B}_{\sigma,j,d}$.

4.2 Secure Computation of Our Protocols

For the secure computation of conjunctive query protocol, Alice sends both encrypted attributes and values from the predicate to Bob in the cloud. Bob first securely matches attributes. Then Bob matches each v_i with the j-th column of Record table to find the equalities according to Eq. (5) and sends result $ct(\mathbb{H}_{\sigma,d,i})$ to Alice. Then Alice decrypts the results $ct(\mathbb{H}_{\sigma,d,i})$ and gets some IDs where she gets $\mathbb{H}_{\sigma,d,i}$=0 for some d of the σ-th block. In this way, Alice gets k sets of IDs for k values in the predicate of the query. After that, she does the intersection of those sets of IDs to support conjunctive query computation and sends IDs to Bob. Later, Bob sends the corresponding encrypted records from the Record table depending on those IDs. Finally, Alice decrypts the encrypted records using her secret key to get her desired result. On the contrary, to support disjunctive query computation, Alice and Bob do the same thing as required for conjunctive query except that Alice does the union of those sets of IDs and sends those IDs to Bob. In this way, we process secure computation for both of our protocols.

4.3 Hiding Additional Information from Leakage

During decryption, Alice can know some additional information from the computation of the Hamming distance $\mathbb{H}_{\sigma,d,i}$ in Eq. (1) than she needs due to sending encrypted polynomials $ct(\mathbb{H}_{\sigma,d,i})$ to her. But Alice needs to know only those coefficients which has degree $x^{l \cdot d-1}$. We solve the problems by adding a random polynomial at the cloud (Bob) ends separately. For securing the polynomial $\mathbb{H}_{\sigma,d,i}$, Bob also adds another random polynomial r_b to $ct(\mathbb{H}_{\sigma,d,i})$ for masking extra information. Since Alice needs to check only the coefficient of $x^{l \cdot d-1}$ from the large polynomial $ct(\mathbb{H}_{\sigma,d,i})$ produced by Bob, then random polynomial in the ring R can be represented by $r_b = \sum_{d=0}^{n/l} \sum_{i=0}^{l-2} r_{b,l \cdot d+i} x^{l \cdot d+i}$. Here $ct(\mathbb{H}_{\sigma,d,i})$ consists of three ciphertext components as $ct(\mathbb{H}_{\sigma,d,i}) = (c_0, c_1, c_2)$. So Bob adds r_b to the ciphertext as $ct(\mathbb{H}'_{\sigma,d,i}) = ct(\mathbb{H}_{\sigma,d,i}) \boxplus r_b = (c_0 \boxplus r_b, c_1, c_2)$. Here the ciphertext $ct(\mathbb{H}'_{\sigma,d,i})$ contains all required information as a coefficient of $x^{l \cdot d-1}$ and hide all other coefficients using the randomization. In this way, we hide $ct(\mathbb{H}'_{\sigma,d,i})$ to disclose any information to Alice except the coefficient of $x^{l \cdot d-1}$.

5 Performance Analysis

In this section, we present the both theoretical and practical performance of our protocols in comparison to Kim et al. [8] protocol. Here, we experimented our two protocols and compared their performances with conjunctive and disjunctive query results in [8]. Here we use the same scenario as Kim et al. [8] protocol along with the database and queries.

5.1 Theoretical Evaluation

In this section, we figure out the multiplicative depth of equality circuits for Kim et al. [8] and our protocol. To measure the equalities of attributes and values as discussed in Sect. 4, we required the Hamming distance computation in Eq. (1). In addition, the encrypted computation of these Hamming distances required only three polynomial multiplications as in Eq. (5). Furthermore, Kim et al. needed a multiplicative depth of $\lceil \log l \rceil + \lceil \log(1+\rho) \rceil$ for their equality circuits comparing two l-bit message with ρ attributes. On the contrary, our method required only $\log 3$ due to using our packing method. Also, the communication complexity of our protocols is $\mathcal{O}(k \cdot m \cdot l \log q)$.

5.2 Parameter Settings and Security Level

Here, we used the same database settings as shown in [7]. So we consider a database where each record includes 11 attributes with $l = 40$ bits data. Besides, we also consider two cases of 100 and 1000 records in the Record table for our conjunctive and disjunctive query processing with $k = 10$ conditions. Moreover, we encoded the name of each attribute with $\delta = 8$ bits integer. Furthermore, we also set the values of some other security parameters required for the SwHE in the experiments. We also considered the equality as a comparison operator. Moreover, we took the block size $\eta = 100$. We also considered appropriate values for the parameters (n, q, t, ω) of our security scheme as discussed in Sect. 2 of [12] for successful decryption and achieving a certain security level. As mentioned in Sect. 3 of our protocols, we need the lattice dimension $n \geq (\lambda \cdot \delta)$ for attributes comparison and $n \geq (\eta \cdot l)$ for values comparison. For this reason, we set $n = 100 \cdot 40 = 4000$ for values matching. In addition, we set $n = 2048$ for attribute matching to provide better security in the computation. Furthermore, we set $t = 2048$ for our plaintext space R_t. According to the work in [9], we choose the standard deviation $\omega = 8$ and $q \geq 16n^2t^2\omega^4 = 2^4 \cdot 2^{22} \cdot 2^{22} \cdot 2^{12} = 2^{60}$ for the ciphertext space R_q during attribute matching. Therefore, we fix our parameters as $(n, q, t, \omega) = (2048, 61\text{-bits}, 2048, 8)$. Similarly, $q \geq 16n^2t^2\omega^4 = 2^4 \cdot 2^{24} \cdot 2^{22} \cdot 2^{12} = 2^{62}$ for values matching. So we set $(n, q, t, \omega) = (4096, 63\text{-bits}, 2048, 8)$. According to computation procedure in [14], our parameters settings provide 364-bit security level to protect our protocols from some distinguishing attacks. Also, NIST [1] showed different security levels for many security algorithms and their corresponding validity periods. Furthermore, they declared that a minimum strength of 112-bit level security has a

Table 1. Performance of our protocols for 40-bit data

m (# of record)	k (# of conditions)	Timing (seconds)			Security level	
		Kim et al. [8]	Our protocol		Kim et al. [8]	Our protocol
			Conj	Disj		
100	10	8	2.948	3.058	93	364
1000	10	80	17.191	17.768	93	364

security lifetime up to 2030. They also disclosed that a security algorithm with a minimum strength of 128-bit level security has a security lifetime beyond 2030.

5.3 Implementation Details

Here Table 1 shows the performances of conjunctive and disjunctive query protocols compared to that of Kim et al. [8]. Here, we have implemented our protocols in C++ programming language with Pari C library (version 2.9.1) [13] and ran the programs on a single machine configured with 3.6 GHz Intel core-i5 processor and 8 GB RAM using Linux environment. For a database of 100 records (resp., 1000 records), our conjunctive query protocol took only 2.948 s (resp., 17.191 s). Also, our disjunctive query protocol took only and 3.058 s (resp., 17.768 s) for 100 records (resp., 1000 records). On the other hand, Kim et al. [8] needed 8 sec (resp., 80 s) for both conjunctive and disjunctive query processing over 100 records (resp., 1000 records). Furthermore, we achieve 364-bit security level for both our protocols whereas Kim achieved a security level of 93-bit. They also achieved a security level of maximum 125-bit which made computation time to twice of the timing with a 93-bit security level. Apart from the above advantages, we are also able to provide security to both values and attributes in the predicate of our queries whereas Kim et al. [8] provided only security to values appeared in the predicate of the query. Besides, Kim et al. [7] also tired to provide security to the attributes, but their performance was lower than that in [8] as shown in Table 4 of [7].

6 Conclusions

In this paper, we have shown two efficient protocols for processing private conjunctive and disjunctive queries over encrypted database using RLWE based somewhat homomorphic encryption in the semi-honest model. Our experiments proved that our protocols achieved a remarkable efficiency than Kim et al. [7,8] with a better security level. Furthermore, we have achieved the efficiency due to using low-cost equality circuits and batch technique with the packing methods. Moreover, our protocols can support a larger data size for both query and database by increasing the lattice dimension n.

Acknowledgment. This research is supported by KAKENHI Grant Numbers JP16H01705, JP17H01695, and JP24106008 for Scientific Research on Innovative Areas.

References

1. Barker, E.: Recommendation for key management. In: NIST Special Publication 800–57 Part 1 Rev. 4. NIST (2016)
2. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 102–118. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38980-1_7
3. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theor. **6**(3), 13 (2014)
4. Cheon, J.H., Kim, M., Kim, M.: Search-and-compute on encrypted data. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 142–159. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48051-9_11
5. Cheon, J.H., Kim, M., Kim, M.: Optimized search-and-compute circuits and their application to query evaluation on encrypted data. IEEE Trans. Inf. Forensics Secur. **11**(1), 188–199 (2016). IEEE Press, New York
6. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Symposium on Theory of Computing - STOC 2009, pp. 169–178. ACM, New York (2009)
7. Kim, M., Lee, H.T., Ling, S., Ren, S.Q., Tan, B.H.M., Wang, H.: Better security for queries on encrypted databases. IACR Cryptology ePrint Archive, 2016/470 (2016)
8. Kim, M., Lee, H.T., Ling, S., Wang, H.: On the efficiency of FHE-based private queries. IEEE Trans. Dependable Secure Comput. 10.1109/TDSC.2016.2568182. (to appear)
9. 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)
10. 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)
11. 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)
12. Saha, T.K., Koshiba, T.: Private conjunctive query over encrypted data. In: Joye, M., Nitaj, A. (eds.) Progress in Cryptology - AFRICACRYPT 2017. Lecture Notes in Computer Science, vol. 10239, pp. 149–164. Springer, Cham (2017)
13. The PARI∼Group, PARI/GP version 2.9.1, Bordeaux (2014). http://pari.math.u-bordeaux.fr/
14. 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.M. (eds.) DPM/SETOP -2013. LNCS, vol. 8247, pp. 34–50. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54568-9_3