Privacy-preserving Min and k-th Min Computations with Fully Homomorphic Encryption

Bingbing Jiang
Department of Computer Science and Technology
Nanjing University
Nanjing,China
Email:njubing.jiang@gmail.com

Yuan Zhang
Department of Computer Science and Technology
Nanjing University
Nanjing,China
Email:zhangyuan05@gmail.com

***Abstract*—We design a secure protocol that a server can compute the minimum number of all participants' data while kept unknown additional information about the users' data in its execution. Ours protocol is based on fully homomorphic encryption and we utilize it to fix the problem of securely computing the minimum value. Besides, the used FHE scheme is a lattice-based cryptosystem that can resist some quantum adversaries who can carry out quantum computations on classical queries. We also expand the secure min computing protocol to the privacy-preserving the k-th min computing protocol, and present the security analysis of the two protocols on the assumption of the semi-honest model. In the previous literature, their solutions are based on secure arithmetic sum computations as well as secure bitwise XOR computations. Our protocols are more secure with the comparison to them.**

I. Introduction

A sensing job can be outsourced to many smartphone users due to the rapid development of the smart phones that have decent computing and sensing ability now. Then, the researchers can perform some computations on the data collected by the users. Recently, there exists many works [1], [2], [3], [4], [5] about smartphones' sensing in the previous research.

However, the users can reveal their private information in the process of sending the collected data to the server. Thus, some smartphone users may refuse to participate in these assignments or they are not willing to submit their data to the publishers of the tasks. Sequentially, it results in the loss of the promulgators' time and money. To fix the above trouble, the users can transmit the encrypted data to the server, and the server is not able to compute the extra knowledge about the plain data in the execution of the protocol.

We just research how the server securely computes the minimum and the k-th minimum value of the users' data in this paper. In the mobile sensing jobs, the researchers usually computes the min and the k-th min value. The investigators, for example, calculates the minimum temperature or the minimum air quality indexs of a city. There are serval works [6], [7], [8], [10] that research the secure computing the min and the k-th min number in the mobile sensing system. In these protocols' execution, the users have to interact with the server other than sending the information and assisting the server to decrypt the ciphertext of the minimum value. The authors of [6] , to find the minimum vaue, propose the protocol that is based on additionally homomorphic encryption and binary searches. In [7], [8], all users' data is limited in an interval, and then the server traverses the entire interval from the smallest to the largest to find out a number that is owned by at least one user. When the range of the data is very large, the server have to waste more time for computing the minimum number. Otherwise, the server give a predication of the range of the data, but this can lead to the wrong result. The exception of the above protocols based on additionally homomorphic encryption, the protocols presented in [10] are on the strength of the secure bitwise XOR computations and more efficient with comparison to the above three protocols.

With the difference of the above protocols on the strength of additionally homomorphic encryption or the secure bitwise XOR computations, our protocols are based on fully homomorphic encryption. The notion of fully homomorphic encryption is first introduced by Rivest, Adleman and Dertouzous in 1978 [30] and untill 2009, Gentry [24] first presents a fully homomorphic encryption scheme though its constructuion is very complex and its efficiency is low so that the scheme cannot apply in practical. Afterwards, the number of works make improvement on the design and efficiency of fully homomorphic encryption. The scheme proposed in [26] can homomorphically perform a bit computations and refresh the final output in just about half a second. Hence, fully homomorphic encryption will be more efficient. Our proposed protocols are based on the fully homomorphic encryption [25], and their efficiency is dependent on the efficiency of fully homomorphic encryption. In addition, the used FHE scheme in our protocols is a lattice-based cryptosystem that can resist some quantum attackers who can excute quantum computations on classical queries. The scheme [25] is based on learning with errors [31] that has been proved that it is the same hard as the lattice intractable problems [31], [32]. Thus, our protocols are more secure than others which are based on secure arithmetic sum computations or secure bitwise XOR computations. Moreover, The round and communication complexity of privacy-preserving min or k-th min computing protocols is imporved, which is introduced detailedlly in

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

section 5. Finally, we also present the formal security analysis of our protocols in section 5.

Next, we simply present our privacy-preserving min or k-th min computing protocols. In our privacy-preserving min computing protocol, the server dertermines the minimum value bit by bit form the most significant bit to the lowest significant bit. Of course, the server always makes the computations on the ciphertexts and obtains the ciphertext of the minimum value bit by bit. How the server decide the j-th most significant bit of the ciphertext of the minimum value? We apply the fully homomorphic encryption scheme to encrypt each user's message. So, the server can perform some operations on j-th most significant bit of each ciphertext and then obtains its wanted result. In our privacy-preserving k-th min computing protocol, the server also dertermines the number of "efficitive" ciphertext, and then shrinks the range including the minimum value.

The rest of the paper is organized as follows. In section 2, we survey the related work.In section 3, we present some necessary preliminaries. In section 4, we introduce our protocols of privacy-preserving min or k-th min computation, and the security analysis is presented in section 5. We conclude this paper in section 6.

II. Related Work

Recently, most previous works [6], [7], [8], [9], [11], [12], [13], [14] that the researchers study the problem of data aggregation for mobile sensing without revealing anything more about the data, have emerged in literature. However, many of them [9], [11], [12], [13], [14] only study how to securely compute the sum of all users' data. Only the authors of [6], [7], [8], propose the min computation protocol based on their sum computation protocol.

Shi et al.[6] , to find the minimum vaue, propose the protocol that is based on additionally homomorphic encryption and binary searches. In [7], [8], all users' data is limited in an interval, and then the server traverses the entire interval from the smallest to the largest to find out a number that is owned by at least one user. When the range of the data is very large, the server have to waste more time for computing the minimum number. Otherwise, the server give a predication of the range of the data, but this can lead to the wrong result. The except of the above protocols based on additionally homomorphic encryption, the protocols presented in [10] are on the strength of the secure bitwise XOR computations and more efficient with comparison to the above three protocols. Unlike additionally homomorphic encryption-based and secure bitwise XOR-based protocols ,our protocols can resist some adveraries who can perform quantum computation between classical queries on account that our protocols are on the strength of lattice-based cryptosystem.

The main difference between our protocols and similar works of wireless sensor networks is application scenarios. These protocols [15], [16], [17], [18], [19] rely on communication and cooperation among senor nodes while there is no interaction between each users in our min computation protocol. But our the k-th min computation protocol needs some communication among every smartphone users, because we apply an additionally homomorphic encryption to compute the "efficitive" ciphertexts. Anothor relevant work[20] is based on homomorphic bitwise XOR operations to find the minimum value.

There are some outstanding techniques, which can be applied to solve the problem of the privacy-preserving min or k-th min calculation, such as order-preserving encryptions(OPE) [21], [22] in the database field. OPE can perform the comparison operations on ciphertexts directly, and so it can obtains the order of the plaintexts to find the minimum value. However, we require that the server should know nothing extra about the data, include the order information.Our protocols only can leak the information of the minimum value or the k-th minimum value to the server.

III. Preliminaries

We present two protocols of securely computing the minimum value or the k-th minimum value in this paper. Detailedly speaking, We study a problem how a server calculates the min or the k-th min number of all users' data while learning nothing additional from the data. Our protocols to fix this problem are based on fully homomorphic encryption, and we also provide the security proof of our protocols under the semi-honest model. Thus, we first introduce the semi-honest model and fully homomorphic encryption in this section.

A. Semi-honest Model

Our discussed security model is that we assume each party is honest but curious. That is, all parties abide by the protocol and cannot abort its excution. Nevertheless, each party is willing to learn others' data and they can perform the computations on data received from others. We require that the protocol is secure under this model. Namely, Every party cannot obtain extra knowledge about others' data in the process of the protocol's excution. On the assumption of the semi-honest model, we prove that our protocols are secure, and the server learns no more information about all users' data and vice versa.

The more knowledge of the semi-honest model is introduced in [23]. We give the formal definition of the semi-honest model as follows.

Definition 1: The protocol $\mathcal{M}$ is said to perfectly privacy-preserving compute the minimum value against party P_i if it reveals nothing additional than the final result to P_i in the excution of $\mathcal{M}$. That is, given all inputs $\{x_0, \ldots, x_n\}$, there exists a polynomial time simulator S_i that can simulate the party P_i. We denote $\{S_i(x_i, \mathcal{M}(x_0, \ldots, x_n), K_i)\}_{\{x_0,\ldots,x_n\}}$ to S_i's view and $\{VIEW_i(x_0, \ldots, x_n)\}_{\{x_0,\ldots,x_n\}}$ represents the party P_i's view in $\mathcal{M}$'s excution. We say $\mathcal{M}$ is secure in the semi-honest model, if S_i's view is computationally indistinguishingable with P_i's view and $K_i = \varnothing$, where K_i represents the party P_i's the extra infomation about others' data. Namely,

$$\{S_i(x_i, \mathcal{M}(x_0, \ldots, x_n))\}_{\{x_0,\ldots,x_n\}}$$

$$\stackrel{c}{\equiv} \{VIEW_i(x_0, \ldots, x_n)\}_{\{x_0,\ldots,x_n\}}$$

If $K_i \neq \emptyset$, we say $\mathcal{M}$ is weakly privacy-preserving against P_i, in the sense that P_i learns no more information than K_i about others' private data in the excution of $\mathcal{M}$.

We give the above formal definition of the semi-honest model that is slightly different with the standard definition introduced in [23]. The standard definition is described from the ideal model and the real model, and it is more rigorous. But the above definition is easily understanding and equivalent to the standard definition. Besides, here we consider the cases that only the server is entitled to get the final output of the protocol.

B. Homomorphic Encryption

A homomorphic encryption scheme is a primary cryptosystem that supports one can make some operations on ciphertexts and then gets a ciphertext of the result that one performs the same computations on the corresponding plaintexts. Homomorphic encryption is simply classified as additionally homomorphic encrypton and multiplicatively homomorphic encrypton. If one can do arbitrary computations on the encrypted data and obtain a valid ciphertext of the ouput that the same operations are performed on the corresponding messages, We say that scheme is fully homomorphic. We now present the formal definition of homomorphic encryption and fully homomorphic encryption, as follows, that mostly taken from [25], [27], [28], [29].

1) Homomorphic Encryption: A homomorphic encryption scheme consists of four probabilistic polynomial time algorithms $(Gen, Enc, DEC, Eval)$, where $Eval$ is the algorithm that supports homomorphic computations on ciphertexts.

Gen(1^κ)**:** $(pk, evk, sk) \leftarrow Gen(1^\kappa)$. Choose the security parameter κ. Generate a public encryption key pk,a public evaluation key evk and a private key sk.

Enc(pk, μ)**:** $c \leftarrow Enc(pk, \mu)$. Encrypt a message $\mu \in \{0, 1\}$ and output the ciphertext c.

Dec(sk, c)**:** $\mu^* \leftarrow Dec(sk, c)$. To decrypt a ciphertext c and obtains a corresponding plaintext $\mu^* \in \{0, 1\}$.

Eval$(evk, f, c_1, \ldots, c_l)$**:** $c_f \leftarrow Eval(evk, f, c_1, \ldots, c_l)$. To homomorphically evaluate the function f and the inputs $c_1, \ldots, c_l$, and output c_f that is the ciphertext of $f(\mu_1, \ldots, \mu_l)$, where c_i is a valid ciphertext of μ_i, $i = 1, \ldots, l$.

We can see that homomorphic evaluation ability of a homomorphic encryption scheme is limited in that it only sustains some operations on the encrypted messages. We give the following definition that describes detailedly the homomorphic evaluation ability of a homomorphic encryption scheme.

Definition 2 (C-homomorphic): Denoted a class of functions $C = \{C_\kappa\}_{\kappa \in \mathbb{N}}$. We say a scheme $\Pi = (Gen, Enc, Dec, Eval)$ is C-homomorphic, if for any arbitrary functions $f_\kappa \in C_\kappa$, and inputs $c_1, \ldots, c_l$ such that $c_i \leftarrow Enc(pk, \mu_i)$, $i = 1, \ldots, l$, it satisfies that the probability of incorrectly decrypting the homomorphically evaluated result is negligible, that is

$$Pr[Dec(sk, Eval(evk, f_\kappa, c_1, \ldots, c_l)) \neq f_\kappa(\mu_1, \ldots, \mu_l)] = neg(\kappa)$$

where $(pk, evk, sk) \leftarrow Gen(1^\kappa)$.

Obviously, though a homomorphic encryption scheme can satisfy correctly decrypting the homomorphically evaluated outputs, the output length of $Eval$ may be relative to the numbers l, even the output may contain some knowledge about the function f_κ or messages $\mu_1, \ldots, \mu_l$, . If so, the scheme can be easy to destroy. Therefore, that result of $Eval$ has to reveal nothing about the function f_κ and messages $\mu_1, \ldots, \mu_l$. We now make some requires on the outputs of $Eval$.

Definition 3 (compactness): Let κ be the security parameter. If there exists a polynomial $p = p(\kappa)$ holding that the output length of $Eval$ is at most $p(\kappa)$ bits long without relation to the function f or the numbers of inputs, we say the homomorphic encryption scheme $\Pi = (Gen, Enc, Dec, Eval)$ is compact.

We have introduced the fundamental and necessary knowledge of homomorphic encryption. That is beneficial to understand the notion of fully homomorphic encryption as well as our proposed the privacy-preserving min or k-th min computing protocols. Next, we introduce some necessary information about fully homomorphic encryption.

2) Fully Homomorphic Encryption: The notion of fully homomorphic encryption is first introduced by Rivest, Adleman and Dertouzous [30] in 1978 and untill 2009, Gentry [24] first presents a fully homomorphic encryption scheme though its constructuion is very complex and its efficiency is low so that the scheme cannot apply in practical. Afterwards, the number of works make improvement on the design and efficiency of fully homomorphic encryption. The scheme proposed in [26] can homomorphically perform a bit computations and refresh the final output in just about half a second. Hence, fully homomorphic encryption will be more efficient. Simply, fully homomorphic encryption is a special homomorphic encryption that supports that one can perform arbitrary computations on ciphertext and obtain a valid and "wanted" ciphertext. The formal definition is as follows.

Definition 4 (fully homomorphic encryption): A scheme $\Pi = (Gen, Enc, Dec, Eval)$ is fully homomorphic, if Π is compact and it can homomorphically evaluate arbitrary circuit taken from the class of all arithmetic circuits over $GF(2)$.

Our proposed protocols are based on the fully homomorphic encryption [25], and their efficiency is dependent on the efficiency of fully homomorphic encryption. In addition, the used FHE scheme in our protocols is a lattice-based cryptosystem that can resist some quantum attackers who can excute quantum computations on classical queries.

Though fully homomorphic encryption is a powerful technique, there are some troubles directly applying it in our scenario. For example, the server is able to perform some computations on ciphertexts on the condition that the ciphertexts are encrypted by the same public key. Besides, each user has negligible probability to decrypt others' data and the server cannot decrypt any users' data. So, the corresponding secret key must not be saved by any party alonely. Next, We introduce a special fully homomorphic encryption to fix the above problem, called threshold fully homomorphic encryption(TFHE). TFHE is basically a fully homomorphic

encryption scheme with the deference that the Gen and Dec are now N-party probabilistic polynomial time protocols.

Definition 5 (threshold Fully Homomorphic Encryption):

A threshold fully homomorphic encryption is composed of four probabilistic ploynomial time algorithms $(Gen, Enc, Dec, Eval)$, where Enc and $Eval$ has no difference with that of fully homomorphic encryption. To put it simply, we here refer the algorithms Gen and Dec.

Gen(1^κ)**:** The algorithm ensure that every party P_i, for $i \in [N]$, obtains a common public encryption key pk, a common public evaluation key evk, and a private share sk_i of the secret key sk.

Dec$(sk_1, \ldots, sk_n, c)$**:** each party P_i use their owned secret share sk_i to decrypt partially the ciphertext c, and then do some specific operations on these partially decrypted ciphertexts to get the final plaintext μ.

Threshold fully homomorphic encryption can be applied to design our privacy-preserving min computing protocols, and the detail of our secure computing the minimum value protocol is introduced in the section 4. Next, we present the concrete threshold fully homomorphic encryption scheme [25], which is based on learning with error. We first introduce the basic LWE-based encryption scheme $E = (Setup, Gen, Enc, Dec)$.

Basic LWE-based Encryption scheme: Set the security parameter κ.

- Setup(κ): Output the following parameters, an odd modulus $q(\kappa)$, two dimensions $m(\kappa), n(\kappa)$ and two distributions $\varphi(\kappa), \chi(\kappa)$ over $\mathbb{Z}_q$. We denoted these parameters $params = (\kappa, q, m, n, \varphi, \chi)$.
- Gen(params): Generate the key pairs (pk, s) such that $s \xleftarrow{R} \varphi^n$, $pk := (A, p)$ where $A \leftarrow \mathbb{Z}_q^{m \times n}$, $p := A \cdot s + 2 \cdot e$, $e \leftarrow \chi^m$.
- Enc(pk, μ): Set $c = (a, b)$ as the ciphertext of a message $\mu \in \{0, 1\}$, where $a := r^T \cdot A$, $b :=< r, p > +\mu$, $r \leftarrow \{0, 1\}^m$ uniformally chosed.
- Dec(s, c): To decrypt the ciphertext c, output $b - < a, s >$ mod q mod 2.

The above scheme is a simple LWE-based encryption system, and the following threshold fully homomorphic encryption scheme is designed on fundamental of it. The TFHE scheme's key generation algorithm is two round protocols that can ensure that the scheme's secret key is saved by each users, that means every party only obtains a secret share of the private key. Thus, in our sence, it is beneficial to construct our privacy-preserving min or k-th min computing protocols.

Threshold Fully homomorphic encryption scheme: We here introduce the threshold homomorphic encryption scheme [25], which consists of four probabilistic polynomial time algorithms $(Gen, Enc, Dec, Eval)$.

Gen(κ)**:** The algorithm Gen is complex two-round protocols. The detail of Gen is introduced in [25], and we here only present its final output.

1) Public Evaluation Key evk.
2) Public Encryption Key pk.
3) A share of secret key s_i is obtained by the party P_i.

Enc(pk, μ): Everyone can encryt the message μ with the public encryption key pk, that is, set $(a, b) \leftarrow Enc(pk, \mu)$. This is the same as the encryption algorithm of basic LWE-based encryption scheme. Except that, one should choose an extra noise e from an interval that is generated from Gen. Thus, the final ciphertext $c = ((a, b + 2e), 0)$ with the "leval" 0.

Eval$(evk, f, c_1, \ldots, c_t)$: The evaluation algorithm is exactly the same as that of corresponding fully homomorphic encryption scheme with the evaluation key evk.

Dec(c): The decryption algorithm requires cooperaton of each party, because each party P_i owns a share s_i such that the secret key $s = \sum_{i=1}^{N} s_i$. To decrypt the ciphertext $c = (a, b, d)$ with the "leval" d, Dec runs as follows.

- Each party P_i broadcasts $b^i =< a, s_i > +2e_i$, where e_i is taken from an interval generated in Gen algorithm.
- Output the plaintext $\mu = [b - \sum_{i=1}^{N} b^i]$ mod q_d mod 2, where q_d is an odd modulus in "leval" d.

Threshold fully homomorphic encryption scheme is basically a fully homomorphic encryption scheme, with the difference that each party only has a secret key share. Thus, in our protocols,each user can't decrypt a ciphertxt alonely.

Fully homomorphic encryption is a powerful technique that one can homomorphically evaluate any functions. However, the recent research of fully homomorphic encryption is in general problems such that its constructions and efficiency. We have not found that applying fully homomorphic encryption to solve privacy-preserving min computing so far. Besides, there are some troubles that solving the secure min computing problem directly with fully homomorphic encryption.

IV. Privay-Preserving Min or k-th Min Computation

In this section, we introduce our privacy-preserving min or k-th min computing protocols with threshold homomorphic encryption. The main contribution of our work is that we apply the threshold fully homomorphic encryption to solve the secure minimum value or k-th minimum value computing problem. Originally, the reseach of fully homomorphic encryption is main in its complexity or others' apllication setting. So far, we have not found that some work about utilizing fully homomorphic encryption in our scenario. Now, the fully homomorphic encryption scheme is not efficient in that its homomorphic evaluation algorithm runs slowly. Directly using fully homomorphic encryption in our scene is very unwise. Our work is how to decrease the homomorphic evaluation computations as much as possible in our protocols.

We suppose that there exists a truested authority who can assist the server and the users to establish a key system for once. Suppose the space of the users' data is $[0, M-1](M \geq 2)$ .Let $l = \lceil log_2^M \rceil$ and $m = \lceil log_2^l \rceil$.

A. Privacy-Preserving Min Computation

The privacy-preserving min computation is that a server wants to calculate the minimum value of all users' data while it is kept unknown anything additional about the data in the

protocol's excution. Each user also has negligible probability to know others' private information. In previous works, the protocols are based on secure arithmetic sum computations or secure bitwise XOR computations. They cannot resist some quantum attackers who can make quantum computations on classical queries. So, we, from this point, design a privacy-preserving min computing protocol based on lattice-based cryptosystem that can withstand the above adversaries. The detail of our protocol is as follows and also can be found in Algorithm 1.

Our protocol is based on threshold fully homomorphic encryption scheme introduced in section 3.2.Specifically,our protocol lets all users encrypt their data via TFHE and send the ciphertexts to the server.The server determines the minimum number bit by bit, from the most significant bit(MSB) to the least significant bit(LSB),then sends the minimum number's ciphertext to each user for decrypting the minimum number.We can "see" that each user only do once encryption ,decrytpion and interaction with the server.Next,we give an algorithm how the server determine the minimum number.

We know that fully homomorphic encryption is a special form of encryption where one can perform any algebraic operation on the plaintext by applying the same operation on the ciphertext.So, for conveniently describing and understanding the algorithm, we replace the operation on ciphertext in the algorithm with that on plaintext.After receiving each user's data(actually it gets the ciphertexts),the server start to determine the MSB of the minimum number by doing AND operation on each data's MSB.If every number's MSB is 1,then the server knows the MSB of the minimum number is 1,and is 0 otherwise.Then the server checks whether the bit is equivlent to each data's MSB.If different, it knows this number is not the minimum number ,and the remaining bits are ineffictive to determine the remaining bits of the minimum number,thus we can replace the remaining bits with 1s.The server repeats the above procedure similarly for the remaining bits.

And how the server know whether two bits are equivalence in the condition that it gets the ciphertexts? We can denote each data by the encryption of 0 or 1.If this data is effictive , mark it by the encryption of 0,and by the ciphertext of 1 otherwise.After the server obtains the ciphertext of the MSB of the minimum number, the server adds the result and the ciphertext of each data's MSB.If they are different,the server gets the encryption of 1 ,and the encryption of 0.Then the server does the XOR computation on the result and each data's mark to obtain its new mark which reprensents whether it is effictive. Afterwards,the server uses the new mark to change the number's next MSB.

A formal description of the entire protocol is in Algorithm 1.

Algorithm 1 Algorithm 1:Privacy-perserving Min Computing Protocol

Require:

A group of n users;

User i ($i = 1, \ldots, n$) has the public key pk and the evaluation key evk of TFHE and a share secret key s_D^i;

User i has a number x_i(its binary representation is $x_{i1}, \ldots, x_{il}$);

Ensure:

The server outputs the minimum number in $\{x_i\}_{i=1,\ldots,n}$;

1: user i encrypts his \ her data $x_i : c_{ij} := TFHE.Enc_{sk}(x_{ij})$;

2: user i sends the ciphertext c_i to the server;

3: the server sets each ciphertext's status as "effictive":$s_i := TFHE.Enc_{sk}(0)$;

4: **for** $j = 1, \ldots, l$ **do**

5: the server set $c_{ij} = s_i \oplus c_{ij}$.If the i-th ciphertext is "ineffictive",that is s_i is the encryption of 1,thus c_{ij} becomes the ciphertext of 1 that don't affect the j-th-MSB of the minimum value;c_{ij} is unchanged otherwise;

6: the server determine the j-th-MSB $d_j$of the minimum number via $\prod_{i=1}^{n} c_{ij}$;

7: the server re-sets each ciphertext's status:$s_i = s_i \oplus d_j \oplus c_{ij}$;

8: **end for**

9: the server sends the $d_1, \ldots, d_l$ to each user for decrypting them:$m_j = TFHE.ENC_{sk}(d_j)$;

10: Return the minimum number $d = \sum_{j=0}^{l} d_j \times 2^j$;

B. Privacy-preserving k-th Min Computation

In this section, we present a protocol that the server compute the k-th min value of all users' data. The trouble of designing privacy-preserving k-th min computing protocol is that the server is necessary to judge the j-th MSB of the k-th minimum value in each step. Namely, after the server performs a round computations on the ciphertexts, it can shrink the range including the minimum value. We, next, present the detail of our privacy-preserving k-th min computing protocol, and also can see Algorithm 2 and Algorithm 3.

Similar to the algorithm 1, the server first determine the MSB of the minimum number. Then, the server computes how many "effictive" numbers with the help of all users(please see the algorithm 3).Finally,the server can determine the bit of the k-th minimum number.If the number of "effictive" data is greater than k,the k-th min value is in "effictive" data,and in the "ineffictive" data otherwise. The server repeat the above procedure to determine the minimum number.

We formally present our privacy-preserving k-th min computational protocol in Algorithm 2 and the protocol that the server computes the number of "effictive" ciphertext in Algorithm 3.

V. Analysis

We present the privacy-preserving min or k-th min computing protocols in the above section. Sequentially, we here perform rigious analysis on the protocols' complexity, correctness and security. First, in two privacy-preserving min or k-th min computing protocols, each user only does the following operations, encrypting and sending their data, finally

Algorithm 2 Algorithm 2:Privacy-preserving k-th Min Computation

Require:

A group of n users;

the server decrypts c:$tot = AHE.Dec_{sk_{AHE}}(c)$; User i has the public key pk ,evaluation key evk and a secret key share s_D^i of the TFHE;

User i has the private number $x_i \in [0, 2^l - 1]$(its binary representation is $x_{i1} \dots x_{il}$);

A preset number $K \in \mathbb{N}$;

Ensure:

the server outputs x,the k-th minimum number in $\{x_1, \dots, x_n\}$;

1: user i encrypts his or her number :$c_{ij} = TFHE.Enc_{sk}(x_{ij}), c_i = (c_{i1}, \dots, c_{il})$;
2: user i sends the ciphertext c_i to the server;
3: the server sets each ciphertext's status as "effictive" :$s_i = TFHE.Enc_{sk}(0)$;
4: **for** $j = 1, \dots, l$ **do**
5: the server sets $c_{ij} = s_i \oplus c_{ij}$ .If the i-th ciphertext is "ineffictive",that is s_i is the encryption of 1,thus c_{ij} becomes the ciphertext of 1 that don't affect the j-th MSB of the minimum number,and it keeps c_{ij} unchanged otherwise.
6: the server computes $d_j = \prod_{i=1}^{n} c_{ij}$;
7: all users help the server to compute tot,the number of "effictive" ciphertext.If $tot >= k$,then the j-th MSB of the k-th minimum value is decryption of d_j,and it sets $s_i = s_i \oplus d_j \oplus c_{ij}, k = tot - k$;otherwise,the j-th MSB of the k-th minimum number is the plaintext of $d_j + TFHE.Enc_{sk}(1)$,and it sets $s_i = s_i \oplus (d_j + TFHE.Enc_{sk}(1)) \oplus c_{ij}, k = k - tot$;
8: **end for**
9: the server sends the $d_1, \dots, d_l$ to each user for decrypting them;

Algorithm 3 Algorithm 3:The Secure counting Protocol

Require:

A group of n users;

User i has a number x_i;

User i has the public key pk_{TFHE},evaluation key evk_{TFHE} and a secret key share s_D^i of TFHE;

User i has the public key pk_{AHE} of additionally homomorphicencryption scheme;

The server has the public key pk_{TFHE},evaluation key evk_{TFHE} of TFHE,and the public key pk_{AHE} and secret key sk_{AHE} of the additionally homomorphic encryption scheme;

Ensure:

the server outputs $tot \in \mathbb{N}$,the number of "effictive" ciphertext ;

1: the server decrypts d_j with the help of all users:$m_j = TFHE.Dec_{s_D}(d_j)$;
2: the server sends m_j to each user,and sends c to user 1,where $c = AHE.Enc_{pk_{AHE}}(0)$;
3: **for** $i = 1, \dots, n-1$ **do**
4: user i sets $c = c + AHE.Enc_{pk_{AHE}}(1)$ if $x_{ij} = m_j$,and sets $c = c + AHE.Enc_{pk_{AHE}}(0)$ otherwise;
5: user i sends c to user i+1;
6: **end for**
7: user n does the same as S4, and sends c to the server;
8: the server decrypts c:$tot = AHE.Dec_{sk_{AHE}}(c)$;

assisting the server to decrypt the ciphertext of the minimum value or the k-th minimum value. So, the round complexity of our protocols is $O(1)$. the communication bits is $3N \cdot cl$,where cl is the ciphertext length of the threshold fully homomorphic encryption scheme in the privacy-preserving min computing protocol, and that of the privacy-preserving k-th min computing protocol is more than the above protocol's in that the server has to compute the number of the "efficitive" ciphertext. The communication bits of each time computing the number of the "efficitive" is $(N+1) \cdot acl$, where acl is the ciphertext length of the additional homomorphic encryption scheme. Thus, privacy-preserving k-th min computing protocol's communication bits is $3N \cdot cl + (N+1) \cdot acl \cdot cl$.

Afterwards, our protocols' efficiency is dependent on the efficiency of the threshold fully homomorphic encryption scheme. TFHE's efficiency is not too high in that the greatest breakthough is made by Gentry [24] recently in the area of fully homomorphic encryption, and fully homomorphic encryption is studied extensively in cryptographic. So, the efficiency of fully homomorphic encryption will be improved, at least that now homomorphically computing a bit operatons can be made in just half a second running on a personal computer, which is faster than the previous fully homomorphic encryption scheme.

Next, we discuss the protocols' correctness and security as summarized in Proposition 6 and Theorem 1.

Proposition 6: The accuracy of our privacy-preserving min or k-th computation protocols is exponentially approximate to 1 if all users and the server follow the protocol on the assumption of semi-honest model.

Proof: We can see that the protocol's accuracy is determined by users' decryption, encryption, communication with the server, and the server's homomorphic evaluation computations. The failure of the encryption ,decryption and evaluation of the FHE scheme is negligible. On the condition of semi-honest model, all users and the server follow the excution of protocol. Though they are curious to others' private information, they cannot abort the process of the protocols' excution or send wrong information in the excution. So, the accuracy is exponently approximate to 1. ■

Theorem 1: Our privacy-preserving min or k-th Min computation protocols are perfectly privacy-preserving against all users and the server in the semi-honest model.

Proof: In our protocols,every user only receives the ciphertext of the minimum value,and each one only has a secret key share of the fully homomorphic encryption scheme.Thus,everyone can't decrypt the ciphertext

alonely.Therefore,our protocol reveal no more information to the users.

Similarly,the server only receives the encryptions of all users' data,and perform homomorphic evaluation on the ciphertexts.If the fully homomorphic encryption scheme and the additionally homomorphic encryption scheme are secure,the server can't learn extra knowledge about all users' data. ■

We present formal proof of the protocols' security on the assumption of the semi-honest model. Here, we only discuss the honest-but-curious attackers who can make some quantum computations on classical queries. The previous protocols of privacy-preserving computing the minimum value of all users' data are based on secure arithmetic sum computations or secure bitwise XOR computations, which cannot withstand the above adversaries with some quantum computing ability.

VI. Conclusion

In this paper, we present two protocols that the server compute the minimum value or the k-th minimum value of all users'data without knowing them. The protocol is based on threshold fully homomorphic encryption scheme and our using threshold fully homomorphic encryption scheme is a lattice-based cryptosystem which can resist the quantum attackers, who can do quantum computations on the receiving data. So, our protocol is more secure with comparison to all existing secure computing the minimum value protocol, which are based on secure arithmetic sum computations or secure bitwise XOR computations. We only analyse the complexity, correctness and security of our protocols theoretically, without some experimental data, in that the efficiency of fully homomorphic encryption is limited. However, the technique of fully homomorphic encryption is so powerful that the round and communication complexity of privacy-preserving min or k-th min computing protocol is improved. Besides, The construction of the protocols is also simpler. Therefore, we apply fully homomorphic encryption to solve the privacy-preserving min or k-th min computing problem.

References

[1] R.Herring,A.Hofleitner,D.WOrk,O.Tossavainen, and A.Bayen, *Mobile Millennium-Participstory Traffic Estimation Using Mobile Phones*, 2009.

[2] A.Thiagarajan,L.Ravindranath,K.LaCurts,S.Madden,H.Balakrishman,S.Toledo,and J.Eriksson, "Vtrack:accurate,energy-aware road traffic dely estimation using mobile phones",in *SenSys*, pp. 85-98.

[3] T.Das,P.Mohan,V.N.Padmanabhan,R.Ramjee, and A.Sharma, "Prism:platform for remote sensing using smartphones", in *MobiSys*, S.Banerjee S.Keshav,and A.Wolman,Eds.ACM,2010,pp.63-76.

[4] R.Rana, C.Chou, S.Kanhere, N.Bulusu, and W. Hu, "Earphone:an end-to-end participatory urban noise mapping system,"in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Senor Networks*. ACM, 2010, pp.105-116.

[5] X.Bao and R.R.Choudhury, "Movi:mobile phone based video highlights via collaborative sensing", in *MobiSys*, S.Banerjee, S.Keshav, and A.Wolman,Eds.ACM,2010,pp.357-370.

[6] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense:Privacy-preserving data aggregation in people-centric urban sensing systems", in *INFOCOM*. IEEE,2010,pp.758-766.

[7] Q.Li,and G.Cao,"Efficient and privacy-aware data aggregation in mobile sensing", in *ICNP*. IEEE,2012,pp.1-10.

[8] Q.Li,G.Cao,and T.F.L.Porta,"Efficient and privacy-aware data aggregation in mobile sensing", *IEEE Trans. Dependable Sec.Comput.*, vol.11,no.2,pp.115-129,2014.

[9] V.Rastogi and S.Nath,"Differentially private aggregation of distributed times-series with transformation and encryption ", in *SIGMOD Conference*, A.K.Elmagarmid and D.Agrawal, Eds.ACM,2010,pp.735-746.

[10] Y.Zhang, Q.Chen, S.Zhong, "Efficient and Privacy-preserving Min and K-th Min Computations in Mobile Sensing Systems" ,in *Dependable and Secure Computing ,IEEE Transactions on* ,Vol:PP Issue:99,2015.

[11] G.Ács and C. Castelluccia,"I have a dream! (differently private smart metering)," in *Information Hiding*,ser. Lecture Notes in Computer Science ,T. Filler,T. Pevný,S.Craver,and A. D. Ker,Eds., vol. 6985.Springer,2011,pp.118-132.

[12] E. G. Rieffel, J. T. Biehl, W. van Melle, and A. j. Lee,"Secured histories:computing group statistics on encrypted data while preserving individual privacy,"*CoRR*,vol. abs/1012.2152,2010.

[13] E. Shi,T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song,"Privacy-preserving aggragation of time-series data," in *NDSS*. The Internet Society,2011.

[14] T.-H. H. Chan, E.Shi, and D. Song,"Privacy-preserving stream aggregation with fault tolerance", in *Financial Cryptogrsphy*,ser.Lecture Notes in Computer Science, A. D. Keromytis, Ed.,vol.7397. Springer,2012,pp.200-214.

[15] M. M. Groat,W. He, and S. Forrest,"Kipda:K-indistinguishable privacy-preserving data aggregation in wireless sensor networks," in *INFOCOM,2011 Proceedings IEEE*. IEEE, 2011, pp.2024-2032.

[16] R.Lu, X.Liang, X.Li, X.Lin, and X.Shen, "Eppa:An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *Parallel and Distributed Systems,IEEE Transactions on,* vol.23,no.9,pp.1621-1631,2012.

[17] Y. Yang, X. Wang, S. Zhu, and G. Cao,"Sdap:a secure hop-by-hop data aggregation protocol for sensor networks," in *Proceedings of the 7th ACM international symposium on mobile ad hoc networking and computing*. ACM, 2006,pp.356-357.

[18] W. He, X. Liu, H.Nguyen, K. Nahrstedt, and T. F. Abdelzaher,"Pda:Privacy-preserving data aggregation in wireless sensor networks," in *INFOCOM*. IEEE,2007,pp.2045-2053.

[19] C.Wang, G.Wang, W.Zhang, and T.Feng, "Reconciling privacy preservation and intrusion detection in sensory data aggregation," in *INFOCOM,2011 Proceedings IEEE*. IEEE, 2011,pp.336-340.

[20] B.K.Samanthula, W. Jiang, and S.Madria, "A probabilistic encryption based MIN/MAX computation in wireless sensor networks," in *2013 IEEE 14th International Conference on Mobile Data Management,Milan,Italy,June 3-6,2013 - Volume 1*,2013,pp.77-86.[Online]. Available:http://dx.doi.org/10.1109/MDM.2013.18

[21] R.Agrawal, J.Kiernan, R.Srikant, and Y.Xu, "Order-preserving encryption for numeric data," in *SIGMOD Conference*,G.Weikum, A.C.K önig, and S. Dessloch, Eds. ACM,2004,pp.563-574.

[22] A. Boldyreva, N.Chenette,Y.Lee,and A.O'Neill, "Order-preserving symmetric encryption", in *Adavances in Cryptology-EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. proceedings, 2009*, pp.224-241.

[23] O.Goldreich,*Foundations of cryptography:Basic applications.*Cambridge Univ Press,2004,vol.2.

[24] C. Gentry,"Fully homomrphic encryption using ideal lattices," in *STOC*,pp.169-178,2009.

[25] G.Asharov, A.Jain, A.L.-Alt, E.Tromer, V.Vaikuntanathan, and D.Wichs, "Multiparty Computation with Low Communicaton ,Computation and Interaction via Threshold FHE", In *EUROCRYPT 2012*, LNCS 7237, pp. 483-501, 2012.

[26] L. Ducas, and D. Micciancio, "FHEW:Bootstrapping Homomorphic Encryption in Less Than a Second," in *EUROCRYPT 2015,Part I,LNCS 9056,* pp. 617-640, 2015.

[27] Z.Brakerski, C.Gentry, V.Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping", in *ITCS, 2012*, http://eprint.iacr.org/2011/277

[28] Z.Brakerski, V.Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE",In *SIAM journal on Computing* vol.43,no.2,pp.831-871,2014.

[29] Z.Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP", in *CRYPTO 2012*, LNCS 7417,pp.868-886,2012.

[30] R.Rivest, L.Adleman, and M.Dertouzous, “On data banks and privacy homomorphisms”, in *Foundations of Secure Computatuion*, pp.169-180, 1978.
[31] O.Regev, “On lattices, learning with errors, random linear codes, and cryptography”, in *JACM 2009*, vol.56(6),2009.
[32] C.Peikert, “Public-key cryptosystems from the worst-case shortest vector problem”, in *STOC 2009*, pp.333-342, 2009.