SCIENCE CHINA
Information Sciences

• HIGHLIGHT •

May 2018, Vol. 61 058103:1–058103:3
doi: 10.1007/s11432-017-9205-0

Securely min and k-th min computations with fully homomorphic encryption

Bingbing JIANG & Yuan ZHANG*

Department of Computer Science and Technology, Nanjing University, Nanjing 210046*, China*

Received 17 March 2017/Revised 16 May 2017/Accepted 21 June 2017/Published online 17 November 2017

Citation Jiang B B, Zhang Y. Securely min and k-th min computations with fully homomorphic encryption. Sci China Inf Sci, 2018, 61(5): 058103, doi: 10.1007/s11432-017-9205-0

Due to the rapid development of smart phones that have decent computing and sensing abilities, it is now possible to outsource a sensing job to many smartphone users. Researchers can then perform computations on the data collected by the users. However, the users may reveal their private information in the process of sending the collected data to the server. So, the users may transmit the encrypted data to the server, meaning the server is not able to compute the extra knowledge about the plain data in the execution of the protocol.

In this article, we study how the server securely computes the minimum and the k-th minimum value of the users' data. In mobile sensing jobs, the researchers usually compute the minimum and the k-th minimum value. The investigators could, for example, calculate the minimum temperature or the minimum air quality indexes of a city. There are several studies [1–4] investigating the secure computation of the minimum and the k-th minimum value in the mobile sensing systems. The previous literature either uses additional homomorphic encryption or secure bitwise XOR computation protocols.

Unlike the above protocols, our protocols are based on fully homomorphic encryption. In 2009, Gentry [5] first presented a fully homomorphic encryption scheme. Our protocols utilize a variant of the GSW13 scheme [6] based on learning with errors. We know that the public key of the GSW13 scheme is a matrix, denoted $\boldsymbol{A}$, the secret key is a vector, denoted $\boldsymbol{v}$ and a ciphertext $\boldsymbol{C}$ and the corresponding plaintext μ satisfies $\boldsymbol{C}\cdot\boldsymbol{v}=\mu\cdot\boldsymbol{v}+\boldsymbol{e}$ where $\boldsymbol{e}$ is some small vector. Now, we can let a trusted third party generate a pair $(\boldsymbol{A},\boldsymbol{v})$ and allocate the secret key into n parts $(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n)$ to n participants. So, when the server requires all participants to decrypt a ciphertext $\boldsymbol{C}$, it only sends a part of the ciphertext $\boldsymbol{C}_i$ which consists of part columns of $\boldsymbol{C}$ to a relevant participant, u_i. The user u_i computes $\boldsymbol{C}_i\cdot\boldsymbol{v}_i$ and sends it to the server, and finally the server runs the decryption algorithm of the GSW13 scheme to obtain the plaintext. In addition, each user can generate their own encryption and decryption keys independently. The encryption key of the protocol is then composed of each user's public key.

To find the minimum value, Shi et al. [1], proposed a protocol that is based on additional homomorphic encryption and binary searches. In [2, 3], all users' data is limited to an interval, and then a server traverses the entire interval from the smallest to the largest to find a number that is owned by at least one user. When the range of the data is very large, the server has to waste more time to compute the minimum number. Otherwise, the server predicates the range of the data, but this can lead to an incorrect result. With the exception of the protocols based on additional homomorphic encryption, the protocols presented in [4]

* Corresponding author (email: zhangyuan05@gmail.com)
The authors declare that they have no conflict of interest.

are stronger and more efficient compared with the above three protocols, due to the secure bitwise XOR computation. In [7], the protocol requires a threshold fully homomorphic encryption scheme and an additional homomorphic encryption. Unlike the above protocols, our protocol can not only resist quantum computer attack, but also all participants can generate their own encryption and decryption keys without running a key-allocation protocol through a server or other participants.

A variant of the GSW13 Scheme. we provide a variant of the GSW13 scheme, called varGSW, which has N_0 parties in evaluation. A scheme Π has six probabilistic polynomial time algorithms (**Setup**, **Gen**, **Enc**, **Dec**, **Add**, **Multi**):

Setup$(1^\lambda, 1^L)$. Run GSW13.**Setup**$(1^\lambda, 1^L)$ to obtain the parameters params$=(q, n, \chi, m)$. Let $l = \lfloor \log q \rfloor + 1$ and $N = n \times l$.

Gen(params). A trusted third party runs GSW13.**Gen**(params) to obtain the public key A and the secret key $\boldsymbol{s}$, and parts $\boldsymbol{s}$ averagely into $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{N_0})$. $\boldsymbol{s}_i$ and A are sent to the party P_i, and the public key A is also sent to the server S. Let $\boldsymbol{v}_i$=Powerof2$(\boldsymbol{s}_i)$.

Enc(params, pk, μ). Choose randomly a matrix $\boldsymbol{R} \leftarrow \{0,1\}^{N\times m}$. Then encrypt the message μ as follows:

$$\boldsymbol{C} = \text{Flatten}(\mu \cdot \boldsymbol{I}_N + \text{BitDecomp}(\boldsymbol{R} \cdot \boldsymbol{A})) \in \mathbb{Z}_q^{N\times N}.$$

Output the ciphertext $\boldsymbol{C}$.

Dec(params, $sk, \boldsymbol{C}$). The server S then divides the ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{N\times N}$ into $(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{N_0})$ where $\boldsymbol{C}_i \in \mathbb{Z}_q^{N\times N/N_0}$. Then, each $\boldsymbol{C}_i$ is sent to each party P_i. P_i computes $\mathbf{TC}_i = \boldsymbol{C}_i \cdot \boldsymbol{v}_i$ and sends $\mathbf{TC}_i$ to S. Finally, S computes $\mathbf{TC} = \Sigma_{i=0}^{N_0} \mathbf{TC}_i$ and outputs $\lfloor \mathbf{TC}_{2^{l-1}} / 2^{l-1} \rceil$.

Add(params, $pk, \boldsymbol{C}_1, \boldsymbol{C}_2$). To add two ciphertexts $\boldsymbol{C}_1, \boldsymbol{C}_2 \in \mathbb{Z}_q^{N\times N}$. Output Flatten$(\boldsymbol{C}_1 + \boldsymbol{C}_2)$.

Multi(params, $pk, \boldsymbol{C}_1, \boldsymbol{C}_2$). To multiply two ciphertexts $\boldsymbol{C}_1, \boldsymbol{C}_2 \in \mathbb{Z}_q^{N\times N}$. Output Flatten$(\boldsymbol{C}_1 \cdot \boldsymbol{C}_2)$.

Three operations Powerof2(), Flatten() and BitDecomp() are defined in [6].

Theorem 1 (Informal)**.** For the parameters in original GSW13 scheme, if there exists a probabilistic polynomial time adversary who can break a varGSW scheme, then there exists an efficient algorithm to break the GSW13 scheme.

We here introduce our privacy-preserving minimum or k-th minimum computing protocol with the varGSW scheme. We suppose that there exists a trusted authority that can assist the server and the users to establish a key system, allowing one or all users to generate their public and secret keys and then publish their public key. Suppose the space of the users' data is $[0, 2^{l-1}]$. We assume there are N_0 mobile phone users and everyone P_i has collected a private data x_i. In our second protocol, k denotes the k-th minimum value of all data and k-th is the minimum value of effective data at each loop.

Privacy-preserving min computing protocol.

S1: User P_i encrypts the input x_i : $\boldsymbol{C}_{i,j} :=$ varGSW.Enc$_{pk}(x_{i,j})$, to denote $\boldsymbol{C}_i = (\boldsymbol{C}_{i,1}, \ldots, \boldsymbol{C}_{i,l})$.

S2: User P_i sends the ciphertext $\boldsymbol{C}_i$ to the server.

S3: The server sets each data's status as effective: $\boldsymbol{s}_i :=$ varGSW.Enc$_{pk}(0)$.

S4: The server sets $j = 1$.

S5: If $j \leqslant l$, then go to S6; otherwise, go to S8.

S6: The server sets $\boldsymbol{c}_{ij} = \boldsymbol{s}_i$ or $\boldsymbol{C}_{i,j}$; the server determines the j-th-MSB $\min_j$ of the minimum number via $\Pi_{i=1}^{N_0} \boldsymbol{C}_{i,j}$; the server then resets each data's status: $\boldsymbol{s}_i = \boldsymbol{s}_i$ or $(\mathbf{min}_j + \boldsymbol{C}_{i,j})$.

S7: $j = j + 1$; go to S5.

S8: The server sends the $\mathbf{min}_1, \ldots, \mathbf{min}_l$ to each user for decryption: $m_j =$ varGSW.Dec$(sk, \mathbf{min}_j)$; they then return the minimum number min$= \Sigma_{j=0}^{l} m_j \times 2^j$.

Privacy-preserving $\boldsymbol{k}$-th min computing protocol.

S1: User P_i encrypts the input: $\boldsymbol{C}_{i,j} =$ varGSW.Enc$_{pk}(x_{i,j}), \boldsymbol{c}_i = (\boldsymbol{c}_{i,1}, \ldots, \boldsymbol{c}_{i,l})$; then user P_i sends the ciphertext $\boldsymbol{c}_i$ to the server.

S2: The server sets each data's status as effective: $\boldsymbol{s}_i =$ varGSW.Enc$_{pk}(0)$.

S3: The server sets $j = 1$.

S4: If $j \leqslant l$, then go to S5; otherwise, go to S7.

S5: The server sets $\boldsymbol{c}_{i,j} = \boldsymbol{s}_i$ or $\boldsymbol{C}_{i,j}$. The server computes $\mathbf{min}_j = \Pi_{i=1}^{N_0} \boldsymbol{C}_{i,j}$. The server runs the secure count protocol once to judge whether the j-th MSB of the k-th minimum value of all data is equal to the plaintext of $\min_j$. If equal, set $\mathbf{kmin}_j = \mathbf{min}_j$. Otherwise, $\mathbf{kmin}_j = \mathbf{min}_j +$ varGSW.Enc$(pk, 1)$. The server re-sets each data's status:$\boldsymbol{s}_i = \boldsymbol{s}_i$ or $(\mathbf{kmin}_j + \boldsymbol{C}_{i,j})$.

S6: $j = j + 1$; go to S4.

S7: The server sends the $\mathbf{kmin}_1, \ldots, \mathbf{kmin}_l$ to each user for decryption and receives km_j; Then output the k-th minimum value $k\min = \Sigma_{j=0}^{l} m_j \times 2^j$.

The secure count protocol. The server judges whether the j-th MSB of the k-th minimum value of all data is equal to that of the minimum value of the effective data and the index of the k-th minimum value of all data in the ascending order of the effective data.

S1: The server sends $\mathbf{min}_j + \boldsymbol{C}_{i,j}$ to each user P_i and sends **count**=varGSW.Enc$(pk, 0)$ to the first

party P_1.

S2: The server sets $j = 1$.

S3: If $j \leqslant N_0$, then go to S4; otherwise, go to S6.

S4: P_i collaborates with other users to decrypt $\mathbf{min}_j + \boldsymbol{C}_{i,j}$; if the plaintext is 0, P_i does **count**=**count**+varGSW.Enc$(pk, 1)$ otherwise, **count**=**count**+varGSW.Enc$(pk, 0)$. P_i sends count to next party P_{i+1} if $i < N_0$. Otherwise, P_i sends count to the server.

S5: $j = j + 1$; go to S3.

S6: The server decrypts count with the assistance of all users. If $k <$count, it means that the j-th MSB of k-th minimum value is equal to the plaintext of $\min_j$ and the k-th minimum value of all data is also the k-th minimum value of the effective data. Otherwise, the j-th MSB of k-th minimum value is equal to the plaintext of $\mathbf{min}_j +$ varGSW.Enc$(pk, 1)$ and the k-th minimum value of all data is also the $(k -$ **count**$)$-th minimum value of the effective data.

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

Theorem 2. Our privacy-preserving minimum or k-th minimum computation protocols are perfectly privacy-preserving against all users and the server in the semi-honest model.

In this article, we first provide a variant of GSW13 scheme for our privacy-preserving minimum or k-th minimum value computations protocols. Then, we present two anti-quantum computer attacking protocols that the server securely computes the minimum value or the k-th minimum value of all users' data.

Supporting information Appendixes A and B. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Shi J, Zhang R, Liu Y, et al. Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In: Proceedings of the 29th Conference on Information Communications, San Diego, 2010. 758–766

2 Li Q, Cao G. Efficient and privacy-aware data aggregation in mobile sensing. In: Proceedings of the 20th IEEE International Conference on Network Protocols, Austin, 2012. 1–10

3 Li Q, Cao G, Porta T F L. Efficient and privacy-aware data aggregation in mobile sensing. IEEE Trans Dependable Sec Comput, 2014, 11: 115–129

4 Zhang Y, Chen Q, Zhong S. Efficient and privacy-preserving min and K-th min computations in mobile sensing systems. IEEE Trans Dependable Sec Comput, 2015, 14: 9–21

5 Gentry C. Fully homomrphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, New York, 2009. 169–178

6 Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology — CRYPTO 2013. Berlin: Springer, 2013. 8042: 75–92

7 Jiang B, Zhang Y. Privacy-preserving min and k-th min computatoins with fully homomorphic encryption. In: Proceedings of 34th IEEE International Performance Computing and Communicatons Conference, Nanjing, 2015. 1–8