Ding, Wenxiu; Yan, Zheng; Qian, X.R.; Deng, R.H.

# Computing Maximum and Minimum with Privacy Preservation and Flexible Access Control

*Published in:*
IEEE Global Communications Conference

*DOI:*
10.1109/GLOBECOM38437.2019.9013937

Published: 01/01/2019

*Document Version*
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

*Please cite the original version:*
Ding, W., Yan, Z., Qian, X. R., & Deng, R. H. (2019). Computing Maximum and Minimum with Privacy Preservation and Flexible Access Control. In *IEEE Global Communications Conference* Article 9013937 (IEEE Global Communications Conference). IEEE. https://doi.org/10.1109/GLOBECOM38437.2019.9013937

# Computing Maximum and Minimum with Privacy Preservation and Flexible Access Control

Wenxiu Ding*, Zheng Yan*†, Xinren Qian*, and Robert H. Deng‡
*School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, China
†Department of Communications and Networking, Aalto University, Espoo, Finland
‡School of Information System, Singapore Management University, Singapore
Emails: {wxding, zyan}@xidian.edu.cn, xinrenqian@gmail.com, robertdeng@smu.edu.sg

***Abstract*—With the fast development of Internet of Things, huge volume of data is being collected from various sensors and devices, aggregated at gateways, and processed in the cloud. Due to privacy concern, data are usually encrypted before being outsourced to the cloud. However, encryption seriously impedes both computation over the data and sharing of the computation results. Computing maximum and minimum among a data set are two of the most basic operations in machine learning and data mining algorithms. In this paper, we study how to compute maximum and minimum over encrypted data and control the access to the computation result in a privacy-preserving manner. We present four schemes to realize privacy-preserving maximum and minimum computations with flexible access control that can adapt to various application scenarios. We further analyze their security and show their efficiency through extensive evaluations and comparisons with existing work.**

***Index Terms*—maximum & minimum, privacy preservation, access control, homomorphic encryption, attribute-based encryption**

## I. Introduction

With the fast growth of Internet of Things (IoT), huge volume of data is being generated and then outsourced to the cloud for analysis and processing, which greatly releases the storage and computation burdens of cloud users. However, outsourcing data to the semi-trusted cloud incurs security and privacy risks to data owners. In order to overcome this issue while benefiting from cloud computing, a commonly accepted approach is to outsource encrypted data to the cloud. But it incurs several new issues as described below.

First, encryption seriously complicates data computations. Fully homomorphic encryption (FHE) in principle enables arbitrary computations to be performed on encrypted data. Though significant research progress has been made since Gentry's breakthrough work on FHE [2], FHE is still far from being practical in data analytics applications. There are a number of efforts in recent years realizing computations over ciphertexts based either on partially homomorphic encryption (PHE) [3], [5] or secure multi-party computation (SMC) [6]. Compared with FHE, these schemes are much more efficient but require several rounds of interactions among multiple servers instead of using a single server in FHE.

Second, another important issue associated with secure computation is how to share the computation results among multiple data users. Homomorphic encryption schemes are always a single-user system, while those realizing flexible access control on outsourced encrypted data are not designed to handle access control of the computation results from ciphertexts but merely outsourced data. However, in this big data era, data processing or analysis results always need to be shared among authorized users. For example, outcome of a clinical decision based on Naïve Bayesian Classifier [5] needs to be shared with patients, doctors in charge and other caregivers according to some policy and procedure.

Specifically, most of the multi-server computation schemes mentioned above support basic arithmetic operations such as addition, subtraction, and comparison, but ignore maximum or minimum which are two of the most basic computations in data analysis (e.g., data classification and clustering [7]). Though the schemes in [1], [5] support maximum computation, they are not efficient when the data set is large – the maximum computation in [1] requires too many interactions between servers while the one in [5] introduces a very high computation overhead. Although Chameleon [10] can support privacy-preserving maximum computation in machine learning, it ignores the access control issue. Bost et al. [7] also designed some computations including maximum that underlie classification protocols. However, it needs two entities to first obtain a comparison result through interaction before calculating the maximum value, which increases computation rounds, and thus has a high communication overhead. CryptDB [21] can easily solve the order issue and get the maximum/minimum of a column by order-preserving encryption (OPE), but it applies different encryptions to realize other SQL operations (such as SUM, JOIN) and is not efficient for complex computations.

Generally speaking, most aforementioned works mainly focus on encrypted data computations, but ignore the requirements on the flexible access control over computation results, which is highly requested in many application scenarios. Our

This work is sponsored by the National Natural Science Foundation of China (Grants 61802293, 61672410 and U1536202), National Postdoctoral Program for Innovative Talents (grant BX20180238), the National Key Research and Development Program of China (Grant 2016YFB0800700), the Academy of Finland (Grant 308087 and 314203), the Project funded by China Postdoctoral Science Foundation (grant 2018M633461), the Fundamental Research Funds for the Central Universities (grant JB191504), the open grant of the Tactical Data Link Lab of the 20th Research Insti-tute of China Electronics Technology Group Corporation, P.R. China (grant No. CLDL-20182119), the Key Lab of Information Network Security, Ministry of Public Security (Grant C18614), and the 111 project (Grants B16037).

previous work [8] is one of the few efforts in the literature dealing with the problem of access control over results of basic computations over encrypted data (such as addition, multiplication, and comparison), but maximum and minimum is ignored.

In this paper, we explore the challenging issue of how to compute maximum and minimum over ciphertexts together with access control over the computation results in a privacy-preserving manner, which can extend and complement our previous work. Our concrete contributions are as follows:

- We extend our previous work [8] and enhance the scalability of our previous system to realize maximum and minimum computations. Hence, the system can be more general and flexible.
- We apply secret share to realize the homomorphism of Attribute-Based Encryption (ABE) for fine-grained access control over the computation results, which guarantees that the results are not disclosed to any unauthorized entities including the servers Computation Party (CP) and Data Service Provider (DSP) and hence that a fully-trusted cloud server is not needed in our system.
- We propose four schemes for maximum and minimum computation to support a variety of application scenarios when dealing with different number of data providers and requesters.
- We further analyze the security of the four schemes and show their efficiency through extensive evaluations and detailed comparison with existing work, especially in the case that multiple users access the computation results.

The rest of this paper is organized as follows. In Section II, we briefly review related work about HE-based secure data processing and secure data access control. Section III presents the notations and preliminaries, followed by the system model and our proposed schemes in Section IV. Security analysis and comparisons as well as performance evaluation are given in Section V. Finally, conclusion remarks are summarized in the last section.

## II. Related Work

### A. Homomorphic Encryption based Secure Data Processing

HE is commonly applied to support data analysis over ciphertexts, but most HE-based schemes are designed to achieve data aggregation [3] and cannot support multi-user access. The evidence aggregation in [9] enables multi-party access to computation result, but it ignores other computations. In order to flexibly support more application scenarios, more computations [10]–[12] over encrypted data were designed. FHE algorithms [13], [14] aim to support arbitrary computations over ciphertexts, but they are not satisfactory in practical applications owing to their heavy computation and storage burdens.

ABY [15] combines arithmetic sharing, garbled circuits and Boolean sharing to support several standard operations, such as addition, multiplication and comparison, etc. Further, Chameleon [10] revised ABY to precompute in an offline phase and help reduce its heavy computation cost. Liu et al. [4] also proposed a framework to support privacy-preserving and efficient computations over securely outsourced data, which realizes maximum and minimum operations. But their framework cannot flexibly issue the access rights of data processing results to any number of eligible parties. Though the privacy-preserving classification scheme in [5] realizes maximum computation over multiple inputs, it needs the cloud users to be involved in the computation and the computation over multiple provided data incurs complex interactions among users and servers.

The work in [1] enables privacy-preserving maximum computation based on PHE, which applies the ciphertext transformation in [3] to provide the capability of multi-user access. However, this scheme incurs too many interactions, which aggravates computational overhead, as shown in our experimental tests in Section V.

### B. Secure Data Access Control

ABE can realize fine-grained access control with high flexibility, which has been widely applied in cloud for storage management [16]–[18]. It enables multiple attributes for access judgement and enhances cloud data security. However, few existing works investigate flexible access control over encrypted data computation results in cloud computing. Our previous work [8] realized an access control scheme over seven basic computation results based on PHE and ABE, but did not consider the maximum and minimum computations.

TABLE I
NOTATION DESCRIPTION

| Symbols | Description |
|---|---|
| $g$ | The public system generator; |
| $n$ | The basic parameter for generating modulus; |
| $(sk_i, pk_i)$ | The key pair of entity $i$; |
| $PK = pk_{DSP}^{sk_{CP}} = PK_{CP}^{sk_{DSP}}$ | The public key for data outsourcing; |
| $[m]$ | The ciphertext of $m$ encrypted with $PK$; |
| $[m]_{pk_i}$ | The ciphertext of $m$ encrypted with $pk_i$; |
| $\mathcal{L}(*)$ | The bit length of input data; |
| $r$ | The random number; |
| $N$ | The number of data involved in computation; |
| $(ck_i, pk_{ck_i})$ | The key pair chosen by servers for access control; |
| $CK'$ | The ABE encryption result; |
| $MSK'$ | The master key in ABE for issuing keys; |
| $(SK', PK')$ | The key pair for ABE encryption and decryption; |
| $PK'$ | The public key for ABE encryption. |

## III. Notations And Preliminaries

*1) Notations:* For easy presentation and understanding, Table I summarizes the notations used throughout this paper.

*2) Preliminaries:* To extend our previous work [8] to support maximum and minimum computation with flexible access control, we also apply our previous Homomorphic Re-Encryption System (HRES) [19] as theoretical basis. A brief introduction is as below, refer to [19] for details.

- **System Setup**: $KeyGen$ generates basic key pairs $(sk_i, pk_i) = (sk_i, g^{sk_i})$ for users and servers. Further two servers (i.e., DSP and CP in our system) negotiate and publish the Diffie-Hellman key $PK = pk_{DSP}^{sk_{CP}} = pk_{CP}^{sk_{DSP}}$ for data outsourcing.

- **Encryption**: $Enc(m, pk_i) \rightarrow [m]_{pk_i}$
- **Decryption**: $Dec([m]_{pk_i}, sk_i) \rightarrow m$
- **Data Outsourcing**: $EncTK(m, PK) \rightarrow [m] = [m]_{PK}$
- **Partial Decryption**: $PDec1([m], sk_{DSP}) \rightarrow [m]_{pk_{CP}}$
- **Final Decryption**: $PDec2([m]_{pk_{CP}}, sk_{CP}) \rightarrow m$

In addition, HRES has the following features:

1) $([m]_{pk_i})^t = [t * m]_{pk_i}$;
2) $([m]_{pk_i})^{n-1} = [-m]_{pk_i}$;
3) $([m]_{pk_i})^{1,t} = \left\{((1 + m * n)pk_i^r)^t, g^r\right\} \mod n^2$.

Besides HRES, we adopt KP-ABE [20] for access control. Here we just list the main functions:

- **System Setup**: $Setup^{ABE}(\gamma, U) \rightarrow (PK', MSK')$
- **Data Encryption**: $Enc^{ABE}(M, \gamma, PK') \rightarrow CK'$
- **Key Issue**: $KeyGen^{ABE}(\mathcal{T}, MSK') \rightarrow SK'$
- **Data Decryption**: $Dec^{ABE}(CK', PK', SK') \rightarrow M$
- **Multiplicative Homomorphism of KP-ABE**: $HE^{ABE}$=$Enc^{ABE}(M_1 * M_2, \gamma, PK') = Enc^{ABE}(M_1, \gamma, PK') * Enc^{ABE}(M_2, \gamma, PK')$.

## IV. Proposed Schemes

This section describes the system and security model, followed by the design details of our schemes.

### A. System and Security Model

*1) System Model:* Our schemes can be applied into a system consisting of five different types of entities as shown in Fig. 1. The cloud servers (DSP and CP) cooperate with each other to provide services for cloud users (DRs and DPs) through the following steps:

1) DPs perceive or produce data, and then outsource them for further processing; 2) DSP as a public server takes the responsibility of data storage and computation; CP is mainly responsible for the computation and access control of computation results. DSP and CP cooperate to process encrypted middle result and deliver final processing results to DR; 3) DRs that are interested in the data computation results acquires analysis results from cloud servers according to their own demands; 4) Authority controls the access and issues keys to the authorized DRs.

*2) Security Model:* In our system model, we have the following assumptions. Authority is regarded as a fully trusted party that never colludes with others. Other entities including DSP and CP are semi-trusted and curious about others' private data but follow the design of system protocols strictly. Moreover, owing to interest conflict (e.g., user resources) and legal responsibilities, DSP and CP would never collude with each other for gaining their own reputation and market division.

*3) Design Goal:* **Data Confidentiality** that any unauthorized user cannot get the data provided by each DP or the computation results.

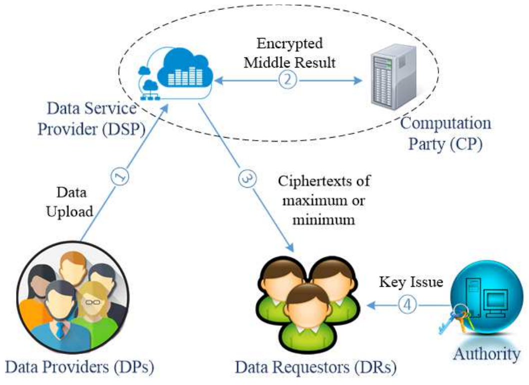

Fig. 1. System model

### B. Proposed Schemes

In this subsection, we present four schemes for privacy-preserving maximum and minimum computations over encrypted data that can adapt to four different application scenarios. First, all schemes need to setup system and collect data in the same procedure as follows:

**Step 1 (@ All Entities)**: The system calls $KeyGen$ to complete the setup of HRES, while the Authority should call $Setup^{ABE}(\gamma, U)$ to generate the basic parameters $PK'$ and $MSK'$ of the ABE algorithm. Then it publishes $PK'$ to its service users.

**Step 2 (@ DPs)**: DP encrypts their personal data $m_i$ by directly invoking $EncTK(m_i, PK)$ and then outsources it to DSP: (Unless otherwise specified, $\mathcal{L}(m_i) < \mathcal{L}(n)/4$ where $\mathcal{L}(*)$ indicates the bit length of input): $[m_i] = \left\{T_i, T_i^{'}\right\} = \{(1 + m_i * n) * PK^{r_i}, g^{r_i}\} \mod n^2$.

Then four schemes are further designed for four scenarios. The basic design idea is to split secret key into two parts, which are respectively handled by two non-colluding servers and encryted with ABE. And then the secret key can be recovered by making use of the homomorphism of ABE.

*1) Maximum and minimum computations over two inputs for a single DR :* The first scheme named Two-to-One (T2O) aims to obtain the maximum and minimum values from two encrypted data for a specified DR. Given two ciphertexts $[m_1]$ and $[m_2]$, T2O can provide the sorting results $[max]_{pk_{DR}}$ and $[min]_{pk_{DR}}$, the ciphertexts of the maximum and the minimum data, to the targeted DR. The detailed interaction procedure between DSP and CP is described as below:

**Step 3 (@ DSP)**: First, DSP randomly selects some numbers $R_1$, $R_2$ and $R_3$ where $\mathcal{L}(R_1) < \mathcal{L}(n)/4$ and then computes:

1) $[1] = \left\{(1 + n) * PK^{r'}, g^{r'}\right\}$
2) $[m_-] = [m_1 - m_2] = [m_1] * [m_2]^{n-1}$
3) $[R_2 * m_+ + R_3] = ([m_1 + m_2])^{R_2} * [R_3]$
4) $[R_2 * m_-] = (T_-, T_-^{'}) = [m_1 - m_2]^{R_2}$
5) $[2 * m_- + 1] = (T, T') = [m_-]^2 * [1]$

where $m_- = m_1 - m_2$ and $m_+ = m_1 + m_2$.

Then it flips a coin $s$. If $s = -1$, then compute $(T_1^{(1)}, T_1'^{(1)}) = \left\{T^{n-R_1}, (T')^{sk_{DSP}*(n-R_1)}\right\} = [-R_1 * (2 * m_- + 1)]_{pk_{CP}}$ and $(T_2, T_2^{'}) = [-R_2 * m_-]_{pk_{CP}} = \left\{T_-^{n-1}, (T_-^{'})^{sk_{DSP}*(n-1)}\right\}$. Otherwise, $(s = 1)$, it computes: $(T_1^{(1)}, T_1'^{(1)}) = \left\{T^{R_1}, T'^{sk_{DSP}*R_1}\right\} = [R_1*(2*m_-+1)]_{pk_{CP}}$ and $(T_2, T_2^{'}) = [R_2 * m_-]_{pk_{CP}} = \left\{T_-, (T_-^{'})^{sk_{DSP}}\right\}$.

It further calls $PDec1([R_2 * m_+ + R_3], sk_{DSP})$ to get $[R_2 * m_+ + R_3]_{pk_{CP}}$. Finally, it forwards CP the data packet $\left\{(T_1^{(1)}, T_1'^{(1)}), [R_2 * m_+ + R_3]_{pk_{CP}}, (T_2, T_2^{'})\right\}$ .

**Step 4 (@ CP)**: CP further processes the data packet from the DSP. It first decrypts $(T_1^{(1)}, T_1'^{(1)})$ and $(T_2, T_2^{'})$ with $PDec2(*, sk_{CP})$ to obtain raw data $\hat{m} = R_1 * (2 * m_- + 1) \mod n$, $\hat{m_-} = (R_2 m_-) \mod n$ if $s = 1$ or $\hat{m} = -R_1 * (2 * m_- + 1) \mod n$, $\hat{m_-} = (-R_2 m_-) \mod n$ if $s = -1$.

Then the CP needs to compare $\mathcal{L}(\hat{m})$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(\hat{m}) < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$. The CP further encrypts the raw data $u * \hat{m_-}$ with the public key of the targeted DR as: $[u * \hat{m_-})]_{pk_{DR}} = (\bar{T}, \bar{T'}) = \{(1 + u\hat{m_-} * n) pk_{DR}^r, g^r\}$ .

Decrypt $[R_2*m_++R_3]_{pk_{CP}}$ and then encrypt it with $pk_{DR}$ to get $[R_2*m_++R_3]_{pk_{DR}}$. Finally, the CP forwards the data packet to DSP: $\{[u*\hat{m_-}]_{pk_{DR}}, [R_2 * m_+ + R_3]_{pk_{DR}}\}$.

**Step 5 (@ DSP)**: The DSP first removes the mask $R_3$ by computing $[R_2*m_+]_{pk_{DR}} = [R_2*m_++R_3]_{pk_{DR}}*[-R_3]_{pk_{DR}}$. Then it can get the maximum and minimum with $r = (2R_2)^{-1} \mod n$:

$$[max]_{pk_{DR}} = ([u * \hat{m_-}]_{pk_{DR}} * [R_2 m_+]_{pk_{DR}})^r$$
$$[min]_{pk_{DR}} = \left([u * \hat{m_-}]_{pk_{DR}}^{(n-1)} * [R_2 m_+]_{pk_{DR}}\right)^r.$$

**Step 6 (@ DR)**: The DR with the corresponding secret key can decrypt the ciphertexts ($[max]_{pk_{DR}}$ and $[min]_{pk_{DR}}$) to obtain the maximum and minimum values.

*2) Maximum and minimum computations over more than two inputs for a single DR :* The second scheme named Multiple-to-One (M2O) aims to obtain the maximum and minimum values from more than two pieces of encrypted data for a specified DR. Given an example of $n$ pieces of ciphertexts $([m_1], [m_2], \cdots, [m_i], \cdots, [m_n])$ , it can get the maximum and minimum results $[max]_{pk_{DR}}$ and $[min]_{pk_{DR}}$ for the targeted DR without revealing the raw data. Note that the T2O scheme can provide the maximum and minimum values from $[m_1]$ and $[m_2]$ for DR. If we use the $PK$ to replace the public key of DR ($pk_{DR}$) in T2O, we can get the ciphertexts $[max]$ and $[min]$.

In order to get the final maximum (as an example) from more than two ciphertexts, the computation of M2O follows the tree structure in Fig. 2. It divides the data into many groups and each group has no more than two pieces of data. Then T2O is executed over every two ciphertexts with $PK$ to get the ciphertext $[max]$. Until the last two pieces of data are obtained in the layer $\lceil lb(n) \rceil - 1$ , DSP and CP execute T2O with $pk_{DR}$ to get the final ciphertext $[max]_{pk_{DR}}$.

*3) Maximum and minimum computations over two inputs for a group of DRs with flexible access control:* The third scheme named Two-to-Multiple (T2M) aims to enable fine-

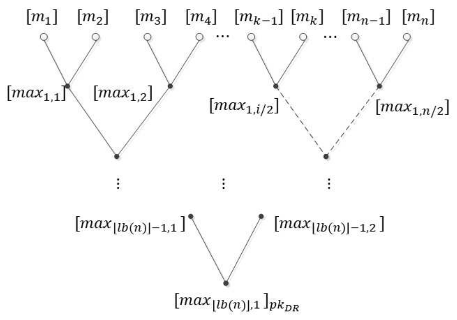

Fig. 2. The procedure of maximum computation over $n$ ciphertexts for a targeted DR

grained and flexible access control over the maximum and minimum computation results for a group of DRs. Given two ciphertexts $[m_1]$ and $[m_2]$, this scheme can provide the sorting results $[max]_{pk_{ck}}$ and $[min]_{pk_{ck}}$, which indicates the ciphertexts of maximum and the minimum results under the public key $pk_{ck}$. Moreover, the corresponding secret key $ck$ of $pk_{ck}$ is encrypted with ABE, which guarantees the fine-grained access control over computation results and supports a group of DRs to access the results.

In order to improve the security of each processed result, two non-colluding servers DSP and CP randomly select secret share to generate their Diffie-Hellman key $pk_{ck}$ for each computation task. The ciphertext of corresponding secret key $ck$ is obtained by combining two ABE ciphertexts, which finally protects the processed data from any unauthorized users including servers. The detailed procedure is presented as below.

**Step 3 (@ DSP)**: DSP randomly selects four numbers: $R_1, R_2, R_3, ck_1$ which satisfies $R_1 = R_2 * ck_1 \mod n^2$ and $\mathcal{L}(R_1) < \mathcal{L}(n)/4$ and then preprocesses the data from DPs as follows:

1) $[1] = \left\{(1 + n) * PK^{r'}, g^{r'}\right\}$
2) $[m_-] = [m_1 - m_2] = [m_1] * [m_2]^{n-1}$
3) $[R_2 * m_+ + R_3] = ([m_1 + m_2])^{R_2} * [R_3]$
4) $[R_2 * m_-] = (T_-, T_-^{'}) = [m_1 - m_2]^{R_2}$
5) $[2 * m_- + 1] = (T, T') = [m_-]^2 * [1]$

By taking the same operations as T2O, DSP also generates the data packet $\left\{(T_1^{(1)}, T_1'^{(1)}), [R_2 * m_+ + R_3]_{pk_{CP}}, (T_2, T_2^{'})\right\}$ .

**Step 4 (@ CP)**: CP decrypts $(T_1^{(1)}, T_1'^{(1)})$ and $(T_2, T_2^{'})$ from DSP to obtain raw data $m' = R_1*(2*m_-+1) \mod n$, $\hat{m_-} = (R_2 m_-) \mod n$ if $s = 1$ or $m' = -R_1*(2*m_-+1) \mod n$ , $\hat{m_-} = (-R_2 m_-) \mod n$ if $s = -1$.

The CP checks the sign of $m'$ by comparing $\mathcal{L}(m')$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(m') < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$. And it further encrypts the raw data $u * \hat{m_-}$ with a randomly chosen key pair $(ck_2, pk_{ck_2} = g^{ck_2})$: $[u * \hat{m_-}]_{pk_{ck_2}} = (\bar{T}, \bar{T'}) = \left\{(1 + u\hat{m_-} * n) g^{ck_2 * r}, g^r\right\}$.

Decrypt $[R_2 * m_+ + R_3]_{pk_{CP}}$ to get $R_2 * m_+ + R_3$ and re-encrypt it as $[R_2 * m_+ + R_3]_{pk_{ck_2}}$. Moreover, it needs to encrypt $ck_2$ with ABE to get $CK_1^{'} = Enc^{ABE}(ck_2, \gamma, PK')$. Finally, the CP forwards the data packet to DSP: $\left\{[u * \hat{m_-}]_{pk_{ck_2}}, [R_2 * m_+ + R_3]_{pk_{ck_2}}, CK_1^{'}\right\}$.

**Step 5 (@ DSP)**: First, the DSP sets $\{\bar{T}, \bar{T'}\} = [R_2 * m_+ + R_3]_{pk_{ck_2}}$, and computes $[R_2 * m_+]_{pk_{ck_2}} = \{\bar{T} * (1 - R_3 * n), \bar{T'}\}$. Then, the DSP computes $r = (2R_1)^{-1} \bmod n$ and finally obtains the encrypted maximum and minimum: $[max]_{pk_{ck}} = \left(\left([u * \hat{m_-}]_{pk_{ck_2}} * [R_2 * m_+]_{pk_{ck_2}}\right)^{1, ck_1}\right)^r$, $[min]_{pk_{ck}} = \left(\left(([u * \hat{m_-}]_{pk_{ck_2}})^{n-1} * [R_2 * m_+]_{pk_{ck_2}}\right)^{1, ck_1}\right)^r$, where $ck = ck_1 * ck_2$ and $pk_{ck} = (pk_{ck_2})^{ck_1} = (pk_{ck_1})^{ck_2} = g^{ck_1 * ck_2}$. The DSP calls $HE^{ABE}$ to obtain $CK = CK_1^{'} * Enc^{ABE}(ck_1, \gamma, PK^{'}) = Enc^{ABE}(ck_1 * ck_2, \gamma, PK^{'})$.

**Step 6 (@ DR)**: DR can decrypt and access the computation results if it satisfies the access policy.

*4) Maximum and minimum computations over more than two inputs for a group of DRs with flexible access control:* The fourth scheme named Multiple-to-Multiple (M2M) aims to enable flexible access control over the computation results from more than two pieces of data. The first $\lceil lb(n) \rceil - 1$ rounds of operations in M2M is the same as that in M2O. But in the last round of computation over $[max_{\lceil lb(n) \rceil - 1, 1}]$ and $[max_{\lceil lb(n) \rceil - 1, 2}]$, DSP and CP invoke the T2M rather than the T2O to obtain the final result $[max_{\lceil lb(n) \rceil, 1}]_{pk_{ck}}$. Due to paper length limitation, we skip the details of above process.

## V. Security Analysis and Performance Evaluation

In this section, we prove the security of the proposed schemes, and show their advanced functionalities and performance through comprehensive simulation and comparison with existing work.

### A. Security Analysis

The security of the four schemes inherits from the semantic security of HRES [19] and ABE. HRES realizes secure data outsourcing with the Diffie-Hellman key of two non-colluding servers. In order to prove their security, we apply the security model for achieving scheme functionalities with the presence of semi-honest (non-colluding) adversaries, which include four kinds of entities except the Authority in our system. Then, we construct four simulators $Sim = (Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$ to fight against their corresponding adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$ that corrupt DP, DSP, CP and DR, respectively.

**Theorem 1**. T2O can securely obtain the maximum and minimum values through computations over encrypted data in the existence of semi-honest (non-colluding) adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$.

**Proof**. Here we construct the four independent simulators $Sim = (Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$.

DP only needs to invoke the $EncTK(m_i, PK)$ to outsource its personal data, hence its security can directly inherit from the original HRES. $Sim_{DP}$ receives the two inputs of $m_1$ and $m_2$, and then it simulates $\mathcal{A}_{DP}$ and encrypts $m_i$ as $[m_i](i = 1, 2)$. Finally, it returns $[m_1]$ and $[m_2]$ to $\mathcal{A}_{DP}$, which is the entire view of $\mathcal{A}_{DP}$. The view of $\mathcal{A}_{DP}$ are indistinguishable owing to the security of HRES.

$Sim_{DSP}$ simulates $\mathcal{A}_{DSP}$ as follows: first $Sim_{DSP}$ calls $EncTK(*, PK)$ to encrypt random messages $\tilde{m_1}$ and $\tilde{m_2}$; then it chooses some random numbers and flips a coin $s$ to obtain $[sR_2\tilde{m_-}]_{pk_{CP}}$, $[sR_1(2\tilde{m_-} + 1)]_{pk_{CP}}$ and $[R_2 * \tilde{m_+} + R_3]_{pk_{CP}}$ by calling $PDec1(*, sk_{DSP})$. By accessing $Sim_{CP}$, it receives $[u * \hat{\tilde{m_-}}]_{pk_{DR}}$, $[R_2 * \tilde{m_+} + R_3]_{pk_{DR}}$, which is encrypted under the public key of targeted DR. Then it applies the additive homomorphism to obtain $[\tilde{max}]_{pk_{DR}}$ and $[\tilde{min}]_{pk_{DR}}$. Finally, $Sim_{DSP}$ outputs $[sR_2\tilde{m_-}]_{pk_{CP}}$, $[sR_1 * (2 * \tilde{m_-} + 1)]_{pk_{CP}}$ and $[R_2 * \tilde{m_+} + R_3]_{pk_{CP}}$, $[\tilde{max}]_{pk_{DR}}$ and $[\tilde{min}]_{pk_{DR}}$ to $\mathcal{A}_{DSP}$. If $\mathcal{A}_{DSP}$ replies with $\perp$, $Sim_{DSP}$ returns $\perp$.

The view of $\mathcal{A}_{DSP}$ only includes the ciphertexts under the public keys of DR and CP. Owing to intrinsic security of HRES and the honesty of challenged cloud users, the outputs to $\mathcal{A}_{DSP}$ in the real and ideal executions are the same. As DSP chooses random numbers in operations, $\mathcal{A}_{DSP}$ still cannot obtain any other information by analyzing the results obtained from several challenges. Thus, the views of $\mathcal{A}_{DSP}$ are also indistinguishable.

$Sim_{CP}$ simulates $\mathcal{A}_{CP}$ as follows: $Sim_{CP}$ first chooses two random numbers $\hat{m_-}$ and $\hat{m_+}$ , and calls $Enc(*, pk_{DR})$ to encrypt them with the public key of DR to obtain $[u * \hat{m_-}]_{pk_{DR}}$, $[R_2 * \hat{m_+} + R_3]_{pk_{DR}}$. Then it further sends them to $\mathcal{A}_{CP}$. If $\mathcal{A}_{CP}$ replies with $\perp$, $Sim_{CP}$ returns $\perp$. $\mathcal{A}_{CP}$ receives the ciphertexts under the public of targeted DR. The security can be ensured by the semantic security of HRES.

$Sim_{DR}$ simulates $\mathcal{A}_{DR}$ as follows: Besides the challenged data, any two random ciphertexts $[\hat{m_1}]_{pk_{DR}}$ and $[\hat{m_2}]_{pk_{DR}}$ are chosen and decrypted to obtain $\hat{m_1}$ and $\hat{m_2}$. Then they are sent to $\mathcal{A}_{DP}$. If $\mathcal{A}_{DP}$ replies with $\perp$, $Sim_{DR}$ returns $\perp$. The view of $\mathcal{A}_{DP}$ is the decrypted data. The semantic security of HRES guarantees the indistinguishability of two views from real and ideal executions. Moreover, the randomly chosen numbers in the challenges have no relation to the originally challenged data but make it difficult to execute exhaustive attacks.

The security proofs of other schemes (i.e., T2M, M2O, M2M) are similar to that of the T2O. The main difference of T2M and M2M from T2O and M2O is that DSP and CP apply ABE to encrypt shares of secret key respectively and then combine the ciphertexts together, which can be guaranteed by the security of ABE [20] and HRES.

### B. Experimental Results

As discussed in Section I, most existing works did not consider multi-user access, while the work [1] can be extended to support multi-user access based on the technique in [3]. Hence, we selected it as a benchmark and further compared it with our work through simulations to show the superiority of our designs in the following subsections. We implemented our four schemes and exisiting work [1] to check the aforementioned theoretical analysis and compared them with each other to show the efficiency and feasibility of our

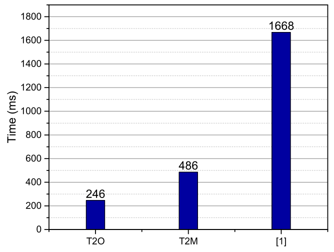

Fig. 3. Comparison of T2O, T2M and [1]

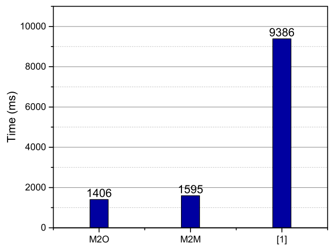

Fig. 4. Comparison of M2O, M2M and [1]

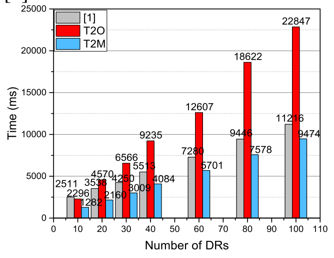

Fig. 5. Performance of T2O, T2M, and [1] with different numbers of DRs

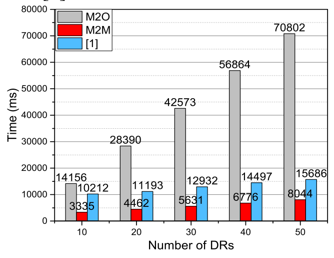

Fig. 6. Performance of M2O, M2M, and [1] with different numbers of DRs

work. We performed our simulations on a laptop with Intel Core i5-5200U CPU @ 2.20GHz and 8 GB SDRAM and applied the Java Paring-Based Cryptography (jPBC) library (http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html#TypeA).

In our tests, we choose the TYPE A curve in jPBC and set $\mathcal{L}(ck_1) = \mathcal{L}(ck_2) = 250$ bits, which should satisfy that $\mathcal{L}(ck_1) + \mathcal{L}(ck_2) < 512$ bits. The curve of TYPE E can also be applied, which can support longer keys of $\mathcal{L}(ck_1) + \mathcal{L}(ck_2) < 1024$ bits and provide higher security. Unless particularly specified, we set the parameters as default values: $N = N_R = 10$, $|U| = |\gamma| = 5$, $\vartheta = 1$, $\mathcal{L}(n) = 1024$, $\mathcal{L}(p) = \mathcal{L}(q) = 512$ bits, $\mathcal{L}(m) = 255$ bits, $\mathcal{L}(sk_i) = 500$ bits and $\mathcal{L}(r) < \mathcal{L}(n)/4$ bits. Hence, the modulus $n^2$ in HRES achieves 2048 bits and guarantees the security. We calculated the average of the simulation results by performing each algorithm for 100 times to improve the test accuracy. In our simulation environment, it takes about 9.4 milliseconds (ms) to complete one bilinear pairing.

*Test 1: Performance comparison of T2O, T2M, and [1]*

In this test, we tested and compared the performance of three schemes: T2O, T2M, and [1]. As the servers in [1] need complex interactions and the tests are executed for 100 times to calculate its average, it is difficult to separate the computation time of two servers. In addition, we note DP and DR only need to do one encryption and one decryption for data outsourcing and access to both maximum and minimum, respectively. Hence, we directly give the total computation time of the whole procedure in each scheme including the data outsourcing time of one DP and the decryption time of one DR in Fig. 3. From the simulation result, we can observe that T2O is much more efficient than [1] and T2M. But T2M can support multi-user access.

*Test 2: Performance comparison of M2O, M2M, and [1]*

Different from Test 1, we only tested the performance of

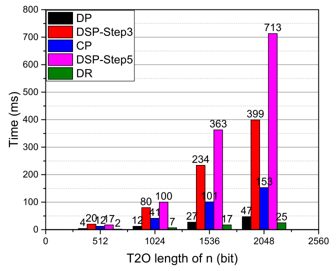

Fig. 7. Efficiency of T2O when the bit length of $n$ increases

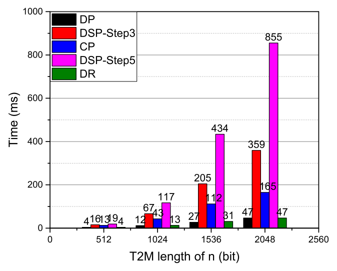

Fig. 8. Efficiency of T2M when the bit length of $n$ increases

maximum computation in M2O, M2M and [1] when dealing with multiple pieces of provided data in all related tests. In this test, we implemented the schemes M2O and M2M and compared them with [1] when there are 10 pieces of encrypted data input ($N = 10$) and one DR for data access. The original work in [1] is designed for two DPs with different keys $(pk_{s1}, pk_{s2})$ and its output ciphertext is under public key $pk_{s1+s2}$. Hence, the output can also be regarded as a new input for further computation with other data, which can provide the capability of dealing with multiple inputs.

Due to the same reason as described in Test 1, we also only give the total computation time of one data outsourcing, data processing and one data access in Fig. 4. We can observe that M2O and M2M do not vary too much but they are superior to [1] since they have smaller number of interaction rounds.

*Test 3: Performance test of T2O, T2M, and [1] with different numbers of DRs*

We further tested and compared the performance of our proposed schemes and [1] with two inputs and a changing number of DRs. We execute T2O for each DR to support multi-user access. Fig. 5 presents the total running time of DSP, CP, Authority and all involved DRs. We can find that T2O is time-consuming to support multi-user access, which takes much longer time than other two schemes. Compared with T2O and [1], T2M performs very efficiently and can flexibly support the access of multiple DRs. Moreover, T2M embeds the access policy into data encryption, which is more secure than the ciphertext transformation [3]. We can see that T2M is very scalable to support many DRs.

*Test 4: Performance test of M2O, M2M, and [1] with different numbers of DRs*

In this test, we compared the performance of M2O, M2M and [1] when dealing with multi-user access to the results of maximum and minimum computations. Fig. 6 shows the performance of three schemes with ten pieces of input data and a varying number of DRs. We can observe that M2M is much more efficient than [1] and M2O, while M2O is not suitable for supporting multiple DRs.

*Test 5: Efficiency test of T2O, T2M, M2O, and M2M when the bit length of $n$ changes*

Furthermore, we demonstrated the efficiency of T2O, T2M, M2O, and M2M with different length of modulus $n$. With $|U| = |\gamma| = 5$ and $\vartheta = 1$, ABE encryption in DSP and

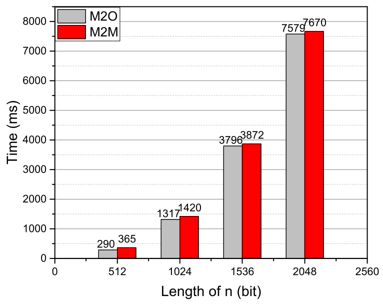

Fig. 9. Efficiency of M2O and M2M when the bit length of $n$ increases

CP takes about 75 ms, and key issue by Authority needs 72 ms. ABE decryption executed by DRs is efficient and takes only 7 ms. As ABE algorithm for encrypting key shares is not related to the length of modulus $n$, thus we exclude the computation time of ABE algorithms and give a clear relationship of modulus $n$ and encrypted data computation procedures. Figs. 7 and 8 show the computation time of all involved entities in each system entity, while Fig. 9 merely gives the total computation time of M2O and M2M due to the complex interactions between DSP and CP.

DPs have the same operations in all proposed schemes. DRs have the same computation in both T2O and M2O, while they also have the same computation in both T2M and M2M although different from that in T2O and M2O. From Fig. 9, we can find that M2M does not increase the computation cost seriously. Moreover, DP and DR only need to do one-time computation. Hence, from the three figures we can find that most computation overhead is moved to the cloud servers, especially the DP and DR have little computation cost, which is suitable for resource-constrained user devices.

Based on the above tests, we can see that each scheme shows great advantages on operation performance when being applied into its specific application scenario. We suggest executing them adaptively according to concrete application demands.

## Conclusion

In this paper, we extended our previous work to support maximum and minimum computation with privacy preservation, which can be selectively applied into various application scenarios. We realized maximum/minimum computation over two or more than two pieces of encrypted data with flexible and fine-grained access control. Security analysis, performance evaluation and comparison further demonstrated that our schemes are secure, efficient and scalable in their specific application scenarios. In the future, we are going to extend the current schemes to support other types of data (not only integers, but also fractions and decimals) and apply them into complex data analytics.

## References

[1] X. Liu, R. H. Deng, K. K. R. Choo, and J. Weng, "An Efficient Privacy-Preserving Outsourced Calculation Toolkit With Multiple Keys," IEEE Transactions on Information Forensics and Security, vol. 11, no. 11, pp. 2401-2414, 2016.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," 47th ACM Symposium on Theory of Computing (STOC). pp. 169-178, 2009.

[3] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," IEEE Transactions on Information Forensics and Security (TIFS), vol. 8, no. 12, pp. 2046-2058, 2013.

[4] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 15, no. 1, pp. 27-39, 2018.

[5] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive Bayesian classification," IEEE journal of biomedical and health informatics, vol. 20, no. 2, pp. 655-668, 2016.

[6] T. Krips, and J. Willemson, "Hybrid model of fixed and floating point numbers in secure multiparty computations," International Conference on Information Security. pp. 179-197, 2014.

[7] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine Learning Classification over Encrypted Data," NDSS, vol. 4324, 2015.

[8] W. Ding, Z. Yan, and R. Deng, "Privacy-Preserving Data Processing with Flexible Access Control," IEEE Transactions on Dependable and Secure Computing, 2017, doi:10.1109/TDSC.2017.2786247.

[9] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," Future Generation Computer Systems, vol. 62, pp. 175–189, 2015.

[10] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications," Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 2018, pp. 707-721.

[11] E. M. Songhori, S. U. Hussain, A. R. Sadeghi, T. Schneider, and F. Koushanfar, "TinyGarble: Highly Compressed and Scalable Sequential Garbled Circuits," 2015 IEEE Symposium on Security and Privacy. pp. 411-428, 2015.

[12] K. Zhou, M. H. Afifi, and J. Ren, "ExpSOS: Secure and Verifiable Outsourcing of Exponentiation Operations for Mobile Cloud Computing," IEEE Transactions on Information Forensics and Security, vol. 12, no. 11, pp. 2518-2531, 2017.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309-325, 2012.

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

[15] D. Demmler, T. Schneider, and M. Zohner, "ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation," NDSS, 2015.

[16] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 1, pp. 131-143, 2013.

[17] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, "On the practicality of cryptographically enforcing dynamic access control policies in the cloud," 2016 IEEE Symposium on Security and Privacy, pp. 819-838, 2016.

[18] H. Tang, Y. Cui, C. Guan, J. Wu, J. Weng, and K. Ren, "Enabling Ciphertext Deduplication for Secure Cloud Storage and Access Control," Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, pp. 59-70, 2016.

[19] W. Ding, Z. Yan, and R. H. Deng, "Encrypted Data Processing with Homomorphic Re-Encryption," Information Sciences, vol. 409–410, pp. 35-55, 2017.

[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," 13th ACM conference on Computer and communications security. pp. 89-98, 2006.

[21] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, pp. 85-100, 2011.