9-2017

An efficient privacy-preserving outsourced computation over public data

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Baodong QIN
Southwest University of Science and Technology

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

DOI: https://doi.org/10.1109/TSC.2015.2511008

Citation

LIU, Ximeng; QIN, Baodong; DENG, Robert H.; and Yingjiu LI. An efficient privacy-preserving outsourced computation over public data. (2017). *IEEE Transactions on Services Computing*. 10, (5), 756-770. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3384

Published in IEEE Transactions on Services Computing,
2017 September, Volume 10, Issue 5, Pages 756 - 770
http://doi.org/10.1109/TSC.2015.2511008

An Efficient Privacy-Preserving Outsourced Computation over Public Data

Ximeng Liu, *Member, IEEE*, Baodong Qin, Robert H. Deng, *Fellow, IEEE*, and Yingjiu Li, *Senior Member, IEEE*

Abstract—In this paper, we propose a new efficient privacy-preserving outsourced computation framework over public data, called EPOC. EPOC allows a user to outsource the computation of a function over multi-dimensional public data to the cloud while protecting the privacy of the function and its output. Specifically, we introduce three types of EPOC in order to tradeoff different levels of privacy protection and performance. We present a new cryptosystem called Switchable Homomorphic Encryption with Partially Decryption (SHED) as the core cryptographic primitive for EPOC. We introduce two coding techniques, called message pre-coding technique and message extending and coding technique respectively, for messages encrypted under a composite order group. Furthermore, we propose a Secure Exponent Calculation Protocol with Public Base (SEPB), which serves as the core sub-protocol in EPOC. Detailed security analysis shows that the proposed EPOC achieves the goal of outsourcing computation of a private function over public data without privacy leakage to unauthorized parties. In addition, performance evaluations via extensive simulations demonstrate that EPOC is efficient in both computation and communications.

Index Terms—Data privacy, encryption, outsourced computation, function privacy

1 Introduction

Cloud computing are gaining momentum in many areas such as Internet of Things (IoT) [1], e-commerce [2], and scientific research [3], [4]. Hybrid cloud is a cloud computing environment which uses a mix of on-premises, private and public clouds to perform distinct functions within an organization. By balancing the use of internal assets and external services, hybrid cloud has drawn much attention not only from the industry, but also from the academic. For example, Microsoft's hybrid cloud [5] aims to take advantage of storage, backup, and recovery options with increased efficiency and reduced cost. MIT Media Lab [6] explores the use of cloud and hybrid cloud to enjoy such use benefits as IT speed, agility and robust, and three-tier disaster recovery (DR).

Huge amount of data stored in the public cloud, such as news, twitters, maps and genomes, are publicly accessible. One of the most commonly used type of publicly accessible data is called the multi-dimensional public data and its analytics requires data processing across multiple dimensions and is receiving tremendous attention in the business world today. For example, let us consider a stock brokerage firm which selects preferred stocks or bonds for its clients based on a proprietary prediction function with multi-dimensional inputs. Computation of the prediction function is time-consuming and hence is outsourced to the cloud, where servers gather data from various stock markets as input, evaluate the function, and return the results to the firm. Based on the results of the computation, the firm provides recommendations of financial securities to its clients. Although outsourcing computation of functions can certainly benefit users who have limited storage and computational capabilities, its flourish still hinges on understanding and managing its flexibility, data security and privacy challenges. In the brokerage firm example, as the prediction function is proprietary to the firm, without appropriate privacy protection, the firm would be very reluctant to delegate the computation of the function to cloud service providers. Furthermore, the results of the functional computation contains valuable and sensitive information, and transmitting such data without adequate protection is clearly not in the interest of the firm and its clients. In short, both the firm's function and its output must be protected throughout the entire outsourcing process.

In this paper, to address the above-mentioned privacy issue in outsourcing functional computation over public multi-dimensional dataset, we propose an **E**fficient **P**rivacy-preserving **O**utsourced **C**omputation Framework over Public Data, called EPOC, which protects privacy of both the function and its output. Specifically, the main contributions of this paper are fourfold.

- We propose a generic EPOC framework which allows privacy-preserving outsourced computation over publicly accessible multi-dimensional data according to a function defined by user. With EPOC, user's private function and the final computed results will not be leaked to other parties. Three concrete constructions of EPOC, called basic EPOC, enhanced EPOC, and full EPOC, are proposed in order to balance security and performance.

- *X. Liu, R.H. Deng, and Y. Li are with the School of Information Systems, Singapore Management University, Singapore. E-mail: snbnix@gmail.com, {robertdeng, yjli}@smu.edu.sg.*
- *B. Qin is with the Southwest University of Science and Technology, China E-mail: qinbaodong@swust.edu.cn.*

TABLE 1
Definitions and Notations in EPOC

Symbol	Definition
pk^+, sk^+	Additive homomorphic (ADD) public & private key
$pk^\times$, $sk^\times$	Multiplicative homomorphic (MUL) public & private key
$E_{pk^+}(\cdot)$	ADD encryption algorithm with ADD public key
$E_{pk^\times}(\cdot)$	MUL encryption algorithm with MUL public key
$(a\|p)$	Legendre symbol (odd prime p and an integer a)
$(a\|n)$	Jacobi symbol (two integer a and n)
$\mathcal{F}, \mathcal{F}', \mathcal{F}''$	The outsourced function
$\vec{\mathbf{C}}$	A plaintext vector $\vec{\mathbf{C}} = (C_1, \ldots, C_n)$
$\vec{\mathbf{C}}^{E_+}$	An ADD encrypted vector $\vec{\mathbf{C}}^{E_+} = (E_{pk^+}(C_1), \ldots, E_{pk^+}(C_n))$
$\vec{\mathbf{C}}^{E_\times}$	A MUL encrypted vector $\vec{\mathbf{C}}^{E_\times} = (E_{pk^\times}(C_1), \ldots, E_{pk^\times}(C_n))$
$a \cdot b$	Multiplication between a and b over cyclic group

- As the core cryptographic primitive of realizing EPOC, we present a new cryptosystem called Switchable Homomorphic Encryption with Partially Decryption (SHED). In SHED, an additive homomorphic ciphertext is transformed into a multiplicative homomorphic ciphertext and the additive homomorphic private key can be randomly separated into two shares for distributively decryption.
- SHED is constructed over a composite group. We introduce two coding methods, called Message Precoding Technique (MPT), and Message Extending and Coding technique (MEC), respectively, for converting messages into the input domain of SHED. We also present a Secure Exponent Calculation Protocol with Public Base (*SEPB*) for securely computing exponential functions with public base and encrypted exponent.
- To assess performance of the proposed EPOC, we develop a custom simulator in Java. Extensive simulation results show that our EPOC is efficient in both computation and communications.

Applications. Our EPOC framework can protect user's computation privacy and its final results with public input which contains multitudinous applications, for example, the National Center for Health Statistics (NCHS) provide public-use data [7] in the cloud server. A commercial company can compute a function (e.g., a prediction model) over the public dataset in order to obtain forecast on-the-fly while without compromising the privacy of computation and the final results. Moreover, our EPOC framework focuses on protecting computational privacy of outsourced polynomial functions. Polynomial functions [8] are considered as one of the most important kernel in Support Vector Machines (SVMs), which find numerous applications, such as prediction of foreign currency exchange rates [9], prediction of bankruptcy [10], and protein sub-cellular localization [11]. Protecting polynomial function is the key to protect polynomial kernel SVM. The protected model can be used for privacy-preserving classification and prediction.

2 Preliminary

In this section, we review partially homomorphic cryptosystems and switchable homomorphic encryption in the literature, which will serve as the basis of the proposed EPOC. For ease of reference, Table 1 lists the main notations used throughout the paper.

2.1 Partially Homomorphic Cryptosystems

Partially homomorphic cryptosystems can be categorized into additive homomorphic (ADD) cryptosystems and multiplicative homomorphic (MUL) cryptosystems. Suppose $E_{pk^+}(m_1)$ and $E_{pk^+}(m_2)$ are two additive homomorphic ciphertexts under the same ADD public key pk^+, additive homomorphic cryptosystems has the **additive homomorphism** property:

$$D_{sk^+}(E_{pk^+}(m_1) \cdot E_{pk^+}(m_2)) = m_1 + m_2.$$

Suppose $E_{pk^\times}(m_1)$ and $E_{pk^\times}(m_2)$ are two multiplicative homomorphic ciphertexts under the same MUL public key $pk^\times$, multiplicative homomorphic cryptosystems has the **multiplicative homomorphism** property:

$$D_{sk^\times}(E_{pk^\times}(m_1) \cdot E_{pk^\times}(m_2)) = m_1 \cdot m_2.$$

2.2 Switchable Homomorphic Encryption

The notion of Switchable Homomorphic Encryption (SHE) is proposed in [12] which allows a server and a proxy to transform multiplicative homomorphic ciphertext into additive homomorphic ciphertext. The SHE scheme works as follows:

KeyGen: Given a security parameter k and two large prime numbers p, q, where $p = 2p' + 1$ and $q = 2q' + 1$, $|p| = |q| = k$, compute $N = pq$ and $\lambda = lcm(p-1, q-1)/2$. Define a function $L(x) = \frac{x-1}{N}$, choose a generator g, such that the order of g is $2p'q'$. Then choose two random odd numbers $\theta_1, \theta_2 \in \mathbb{Z}_N$, where $|\theta_1| \approx |\theta_2| < \frac{1}{2}|N|$. Set $\theta = \theta_1\theta_2$, calculate $h = g^\theta \bmod N$, and output the following public-private key pairs:

$$\{pk^+; sk^+\} := \{N; (\lambda, p, q)\}, \ \{pk^\times; sk^\times\} := \{(N, g, h); \theta\}.$$

AddEnc: The algorithm takes as input an ADD public key $pk^+ = N$ and a message $m \in \mathbb{Z}_N$ It chooses a random $r' \in \mathbb{Z}_N$ and outputs the ADD ciphertext as

$$E_{pk^+}(m) = (1+N)^m \cdot {r'}^N \bmod N^2.$$

AddDec: An ADD ciphertext $E_{pk^+}(m)$ can be decrypted using the ADD private key $sk^+ = \lambda$ by first calculating,

$$T_1 = E_{pk^+}(m)^\lambda \mod N^2 = {r'}^{\lambda \cdot N}(1 + mN\lambda) \mod N^2$$
$$= (1 + mN\lambda).$$

Then, due to $gcd(\lambda, N) = 1$,[1] m can be recovered as:

$$L(T_1 \mod N^2)\lambda^{-1} \mod N = m.$$

1. $gcd(x, y)$ is greatest common divisor between x and y.

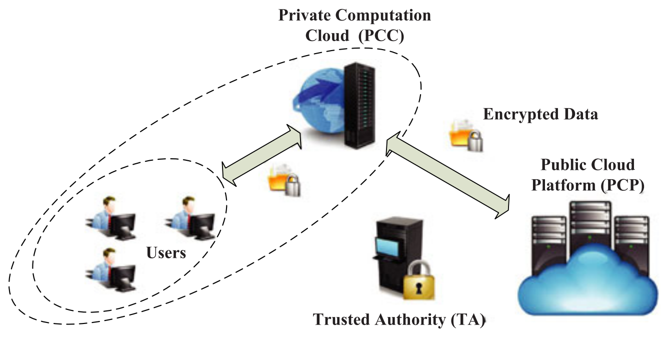


Fig. 1. System model under consideration.

MulEnc: The algorithm takes as input a MUL public key $pk^{\times} = (N, g, h)$ and a message $m \in \mathbb{Z}_N$, such that the Jacobi symbol of m is 1. It chooses a random $r \in \mathbb{Z}_{2p'q'}$ and outputs the MUL ciphertext[2] as

$$E_{pk^{\times}}(m) = \{C_1, C_2\} = \{m \cdot h^r \bmod N, g^r \bmod N\}.$$

MulDec: The MUL ciphertext $E_{pk^{\times}}(m)$ can be decrypted using the MUL private key $sk^{\times} = \theta$ by calculating:

$$C_1/C_2^{\theta} = m \cdot h^r/(g^r)^{\theta} = m.$$

AddToMix: This algorithm is run locally by the server. Given a ADD ciphertext $E_{pk^+}(m)$ and MUL public key $pk^{\times}$. It chooses a random $r \in \mathbb{Z}_{2p'q'}$ and outputs the ciphertext as:

$$E_{pk^+,pk^{\times}}(m) = \{(1+N)^{mh^r} \cdot r'^N \bmod N^2, g^r \bmod N\}.$$

MixToAdd: This algorithm is jointly run by the server and the proxy. Given a mixed ciphertext $E_{pk^+,pk^{\times}}(m) = \{T_1, T_2\}$, the server chooses a random number $s \in \mathbb{Z}_{2p'q'}$, and compute $c_1 = (T_2 \cdot g^s)^{\theta_1} = g^{(r+s)\theta_1} \bmod N$ and $c_2 = g^s \bmod N$, then sends $\{T_1, c_1, c_2\}$ to proxy.

Once received $\{T_1, c_1, c_2\}$, the proxy computes $(c_1)^{\theta_2} = h^{r+s}$ and its inverse $(h^{r+s})^{-1} \bmod N$. Then, it computes

$$T_1' = T_1^{(h^{r+s})^{-1}} \bmod N^2 = E_{pk^+}(mh^{-s}). \tag{1}$$

and $c_2' = (c_2)^{\theta_2} = g^{s\theta_2} \bmod N$, and sends $\{T_1', c_2'\}$ to the server.

The server uses $\{T_1', c_2'\}$ to compute $(c_2')^{\theta_1} = h^s \bmod N$. Then, the ADD ciphertext can be recovered by calculating

$$(T_1')^{h^s} = E_{pk^+}(mh^{-s} \cdot h^s) = E_{pk^+}(m).$$

3 System & Privacy Models

In this section, we formalize the EPOC system and privacy models.

3.1 System Model

Our system consists of four types of generic entities: Trusted Authority (TA), User, Public Cloud Platform (PCP) and Private Computation Cloud (PCC)—see Fig. 1.

2. As we choose composite order of ElGamal encryption, the ciphertext will leak the Jacobi symbol of m. To solve the problem, we only encrypt m with Jacobi symbol 1.

3.1.1 Trusted Authority (TA)

TA is assumed to be trusted by all the other entities to distribute and manage all the private & public keys in the system.

3.1.2 User

The goal of a user is to get computed results over public database according to a function of his choice. The user sends a query to PCP, specifying the function in a privacy-preserving manner. The PCP computes the results and returns them back to the user. Note that the results can only be decrypted with the help of PCC, i.e., the user cannot get the finally results without the authorization of PCC.

3.1.3 Private Computation Cloud

PCC provides computation service to the user, e.g., it can process encrypted data such as multiplication over the encrypted data. Also, PCC has the ability to partially decrypt ciphertexts which are sent from PCP, do some calculations over the plaintexts, and then re-encrypt the data with corresponding user's public key.

3.1.4 Public Cloud Platform

PCP contains unlimited data storage space which stores and manages all the public data. PCP stores all the intermediate and final results in encrypted form. Furthermore, PCP has some computation power to perform certain calculations over encrypted data.

Next, we present the dataset and the function used in our EPOC. Suppose a data space D defined by a set of γ dimensions $\{x_1, \ldots, x_{\gamma}\}$ and a dataset S on D with cardinality γ. A transaction $\vec{\mathbf{a}}_i \in S$ can be represented as $\vec{\mathbf{a}}_i = (a_{i,1}, \ldots, a_{i,\gamma})$, where $a_{i,k} \in \mathbb{G}_{2p'q'}$ is a value on dimension d_k $(i = 1, \ldots, \tau, k = 1, \ldots, \gamma)$. All the transactions are stored as plaintext form in PCP and the dataset S can be accessed by any party in the system (include the adversary $\mathcal{A}^*$ defined in 3.2). A user α's goal is to obtain the output of the function

$$\mathcal{F} : o = \sum_{j=1}^{\kappa} C_j x_1^{t_{j,1}} \ldots x_{\gamma}^{t_{j,\gamma}},^{3}$$

over the public data stored in PCP, where $C_j \in \mathbb{G}_{2p'q'}$ and $t_{j,k} \in \mathbb{Z}_N$. This kind of functions can be used for statistical analysis. For example, the user can calculate the arithmetic mean $\overline{x} = (\sum_{i=1}^{\gamma} x_i)/\gamma$ across different dimensions. The standard deviation $\sigma = \sqrt{\frac{1}{\gamma}\sum_i^{\gamma}(x_i - \overline{x})^2}$ can also be calculated (note that $X = \frac{1}{\gamma}\sum_i^{\gamma}(x_i - \overline{x})^2$, a special case of $\mathcal{F}$, can

3. All the functions used in EPOC are defined over ring of integers $\mathbb{Z}$, thus the function must satisfy the following restriction: 1) the size of output of the function must less than N, i.e. $|o| < N$. 2) the size of every monomials must less N, i.e. $|C_j x_1^{t_{j,1}} \ldots x_{\gamma}^{t_{j,\gamma}}| < N$ for every j.

If the function is defined over $\mathbb{Z}_N$, the function which used in EPOC has no restriction, i.e., the function is formalized as

$$\mathcal{F} : o = \sum_{j=1}^{\kappa} C_j x_1^{t_{j,1}} \ldots x_{\gamma}^{t_{j,\gamma}} \bmod N.$$

be computed on server side, the user can calculate $\sqrt{X}$ by himself to get the standard deviation). Moreover, for polynomial kernel SVM classification, the classifier is to map the data from the input space X to a feature space F. In the space F, the discriminant function is $f(\mathbf{x}) = \sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i) + b$, where $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^d$ is a degree-d polynomial kernel.

3.2 Privacy Model

In our privacy model, PCP and PCC are *curious-but-honest* parties in that they strictly follow the protocol, but are interested in user's private information, e.g., both PCP and PCC are curious about user's private outsourced function and calculated results. EPOC should meet the following privacy requirements.

- *User Privacy*: protect privacy of user's function and the output of the function from PCP, PCC and any external adversary. That is, PCP, PCC and any external adversary cannot learn anything about user's function in the query as well as anything about the output of the function.
- *Privilege Separation*: It can prevent the user to the abuse of privilege without the permission of the PCC and allow PCC to efficiently revoke the user's private key.
- *Security of Data-in-transit*: guarantee confidentiality of data during transmission.

To satisfy these privacy requirements, we assume that there exist an active adversary $\mathcal{A}^*$ in our system. The goal of $\mathcal{A}^*$ is to compromise user's privacy[4] with the following attacking abilities: $\mathcal{A}^*$ may eavesdrop on all the communication links to get the challenge user's encrypted queries, encrypted middle results and encrypted final results. Also, $\mathcal{A}^*$ may compromise PCP, PCC, and users simultaneously, but subject to the following restrictions: 1) $\mathcal{A}^*$ cannot compromise PCP and PCC at the same time. 2) $\mathcal{A}^*$ cannot compromise the challenge user.

For efficiency concern, we achieve the aforementioned privacy requirements by defining the following three privacy levels.

Definition 1 (Level-I Privacy). *Upon completion of an EPOC protocol, $\mathcal{A}^*$ cannot learn the coefficients $C_j (j = 1, \ldots, \kappa)$ of the outsourced function $\mathcal{F}$ and the final outputs.*

Definition 2 (Level-II Privacy). *Upon completion of an EPOC protocol, $\mathcal{A}^*$ cannot learn the coefficients $C_j (j = 1, \ldots, \kappa)$ and degrees $t_{j,1} \ldots, t_{j,\gamma}$ of the outsourced function $\mathcal{F}$ and the final outputs.*

Definition 3 (Level-III Privacy). *Upon completion of an EPOC protocol, $\mathcal{A}^*$ cannot learn the coefficients $C_j (j = 1, \ldots, \kappa)$, degrees $t_{j,1} \ldots, t_{j,\gamma}$, the size κ, and the format of the outsourced function $\mathcal{F}$ and the final outputs.*

Note that both the PCC and PCP can know which dimensions in the dataset S have been used for outsourced function calculation in the level-I and level-II privacy. This kind of information is fully protected in level-III privacy EPOC, i.e., PCC and PCP cannot know which data in S are used for calculation (see Section 4.5 for detailed construction).

4 Proposed EPOC Framework

In this section, we will mainly introduce three versions of EPOC, called basic EPOC, enhanced EPOC, and full EPOC, respectively, to achieve different privacy levels defined in Section 3.2.

4.1 Switchable Homomorphic Encryption with Partially Decryption

In order to realize EPOC, we design a new cryptosystem called Switchable Homomorphic Encryption with Partially Decryption based on Switchable Homomorphic Encryption (SHE) [12]. The **AddEnc**, **AddDec**, **MulEnc**, **MulDec**, **AddToMix**, **MixToAdd** of SHED is the same as the corresponding algorithm in SHE. The SHED works as follows:

KeyGen: Given the security parameter k, choose two large safe prime numbers p', q', compute $p = 2p' + 1$ and $q = 2q' + 1$, such that $p \equiv -1 \bmod 8$ and $q \equiv -1 \bmod 8$, where $|p| = |q| = k$. Then, compute $N = pq$ and $\lambda = lcm(p - 1, q - 1)/2$. Define a function $L(x) = \frac{x-1}{N}$, choose a generator g, s.t., the order of g is $2p'q'$, then choose two random odd numbers $\theta_1, \theta_2 \in \mathbb{Z}_{(N-1)/2}$.[5] It sets $\theta = \theta_1\theta_2$ and calculate $h = g^\theta \bmod N$,. randomly separated λ into two parts and denote the partial ADD private key as λ_1 and λ_2, s.t., $\lambda_1 + \lambda_2 \equiv 0 \bmod \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \bmod N^2$ hold at the same time (the selection of p, q and the existence of the partial ADD private key splitting can be found in Section 5.1). The algorithm output the following ADD and MUL public-private key pairs:

$$\{pk^+; sk^+\} := \{N; (\lambda_1, \lambda_2, p, q)\},$$

$$\{pk^\times; sk^\times\} := \{(N, g, h); (\theta_1, \theta_2)\}.$$

The λ_1 and θ_1 are given to server while λ_2 and θ_2 are given to proxy.

MulToMix: This algorithm is run locally by the server. Given a MUL ciphertext $E_{pk^\times}(m) = \{m \cdot h^r, g^r \bmod N\}$ and public key pk^+ and $pk^\times$, choose a random $r_1 \in \mathbb{Z}_{2p'q'}$ to re-randomize the MUL ciphertext as $E_{pk^\times}(m) = \{m \cdot h^{r''}, g^{r''} \bmod N\}$, where $r'' = r + r_1$. Then, choose a random $r' \in \mathbb{Z}_{2p'q'}$ and output the mixed ciphertext as:

$$E_{pk^+, pk^\times}(m) = \{(1 + N)^{mh^{r''}} \cdot r'^N \bmod N^2, g^{r''} \bmod N\}.$$

AddPDec1: Upon receiving the ADD ciphertext $E_{pk^+}(m)$, uses the partial ADD private key λ_1 to generate the partially decrypted ciphertext $E_{pk^+}(m)$ as:

$$CT(m) = (E_{pk^+}(m))^{\lambda_1} = r'^{N\lambda_1}(1 + mN\lambda_1) \bmod N^2.$$

4. Before attacking a user, the adversary $\mathcal{A}^*$ should issue a target and try to guess the plaintext value of the target user's ciphertext with some existing resources. We call the target user as the challenge user.

5. From [13], the distribution of a uniform random variable over $\mathbb{Z}_{(N-1)/2}$ is statistically indistinguishable from the one over $\mathbb{Z}_{2p'q'}$.

AddPDec2: Upon receiving both the original ciphertext $E_{pk^+}(m)$ and the partially decrypted ciphertext $CT(m)$, recover the original message by using the partial ADD private key λ_2:

$$\begin{aligned} T'' &= CT(m) \cdot (E_{pk^+}(m))^{\lambda_2} \\ &= r'^{N(\lambda_1+\lambda_2)}(1 + mN(\lambda_1 + \lambda_2)) \bmod N^2. \end{aligned}$$

Then calculate $L(T'') = m$ to recover m.

4.2 Key Distribution in EPOC

Before executing EPOC, TA should generate all the private keys and distribute them to the corresponding parties. Suppose there are one PCC, one PCP and β users involved in the EPOC.

TA firstly runs **KeyGen** algorithm to generate β ADD public key $pk_i^+ = N_i$ and corresponding ADD private key $sk_i^+ := (\lambda_{i,1}, \lambda_{i,2}, p_i, q_i)(i = 1, \dots, \beta)$, and MUL private keys $sk_i^\times = \theta_i \bmod N_i$ under N_i. TA uses θ_i to generate $h_i = g^{\theta_i} \bmod N_i$ for each $i = 1, \dots, \beta$, and denotes the $sk_i^\times$'s corresponding public key as $pk_i^\times = (N_i, g_i, h_i)$. The ADD and MUL public key pair $(pk_i^+, pk_i^\times)_{i=1,\dots,\beta}$ are sent to the corresponding user i. Furthermore, the θ_i are needed to generate partial MUL private keys $(\theta_{i,1}, \theta_{i,2})$ and $\theta_{i,1}$ should be sent to PCP while $\theta_{i,2}$ should be sent to PCC for storage respectively. The partial ADD private key $\lambda_{i,1}$ are send to PCC while $\lambda_{i,2}$ are sent to the user i for storage respectively. Notice that no party in the system holds sk_i^+ and $sk_i^\times$, which guarantee no one can directly decrypt encrypted results. Furthermore, pk_i^+ is sent to PCP while $pk_i^\times (i = 1, \dots, \beta)$ are sent to PCC for storage. After distributing all the private keys, EPOC will be executed. Next, we gives detailed construction of EPOC in different privacy levels.

4.3 Basic EPOC for Achieving Level-I Privacy

In our basic EPOC, the outsourced function's coefficients are stored as encrypted form. Once a user α want to retrieve some data from the dataset, both the encrypted coefficients $E_{pk_\alpha^+}(C_i)$ and function's patten $x_1^{t_{j,1}} \dots x_\gamma^{t_{j,\gamma}} (j = 1, \dots, \kappa)$ should be outsourced to PCP.

1. *Secure monomials calculation stage (Stage One)*. In this stage, PCP needs to calculate the monomials over the public dataset which is formalized as $C_j x_1^{t_{j,1}} \dots x_\gamma^{t_{j,\gamma}}$. As all the data $\vec{\mathbf{a}}_i = (a_{i,1}, \dots, a_{i,\gamma})$ are stored in plaintext form in PCP, $a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}}$ $(i = 1, \dots, \tau; j = 1, \dots, \kappa)$ can be calculated by PCP according to the function's patten $x_1^{t_{j,1}} \dots x_\gamma^{t_{j,\gamma}}$. Then, the function's monomials can be easily calculated as:

$$E_{pk_\alpha^+}(C_j)^{a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}}} = E_{pk_\alpha^+}(C_j a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}}),$$

which is stored for further processing.

2. *Secure outsourced function calculation stage (Stage Two)*. Once the secure monomials calculation phase is finished, PCP needs to use $E_{pk_\alpha^+}(C_j a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}})$ to compute the final encrypted results $E_{pk_\alpha^+}(o_i)$ for every i according to the outsourced function $\mathcal{F}$. Due to the additive homomorphism property of SHED, additive operation over the plaintext can be easily achieved by multiplication of the ciphertext as follows:

$$\begin{aligned} E_{pk_\alpha^+}(o_i) &= E_{pk_\alpha^+}\left(\sum_{j=1}^{\kappa} C_j a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}}\right) \\ &= \prod_{j=1}^{\kappa} E_{pk_\alpha^+}(C_j a_{i,1}^{t_{j,1}} \dots a_{i,\gamma}^{t_{j,\gamma}}). \end{aligned}$$

After that, all the encrypted results $E_{pk_\alpha^+}(o_i)$ need to be decrypted by user α. Firstly, all the $E_{pk_\alpha^+}(o_i)$ are send to PCC and are partially decrypted by calling **AddPDec1** with $\lambda_{\alpha,1}$. Then, these partially decrypted ciphertexts are sent to the user α and are decrypted to get the results $o_i (i = 1, \dots, \tau)$ by calling **AddPDec2** with user α's partially private key $\lambda_{\alpha,2}$. That is, the user can only decrypt the final results with the help of PCC.

In the basic EPOC, only the coefficients of the outsourced function are protected. If the user requires a better protection, i.e., to protect not only the coefficients of the function, but also the exponent of every monomials in the outsourced function, the enhanced EPOC will be presented below should be used.

4.4 Enhanced EPOC for Achieving Level-II Privacy

In order to protect coefficient C_j and the exponent of monomials $t_{j,k}$ at the same time, the user should encrypt all these queries with $pk_\alpha^\times$ and send them to PCP. However, there exist two main problems which need to be solved. 1) Because all the MUL ciphertext is computed over N, the message should be chosen over the sub-group $2p'q'$ to guarantee the security of MUL ciphertext, i.e., the plaintext must be chosen such that its Jacobi symbol is 1. Thus, as $t_{j,k}$ are encrypted using **MulEnc** with $pk_\alpha^\times$, the plaintext must satisfy that its Jacobi symbol is 1 and the message is not equal to 0. 2) to securely compute the outsourced function, the PCP must compute the monomials formalized as $C_j x_1^{t_{j,1}} \dots x_\gamma^{t_{j,\gamma}}$. As $t_{j,k}$ is stored in encrypted form, it is impossible for PCP to directly compute $E_{pk_\alpha^\times}(x_k^{t_{j,k}})$ using MUL ciphertext.

In order to solve these two problems, we first present a technique called message pre-coding to solve the MUL encryption problem, and then present a protocol called Secure Exponent Calculation Protocol with Public Base to solve the secure exponent calculation problem. Finally, we use the *SEPB* to construct the enhanced EPOC.

4.4.1 Message Pre-Coding Technique

The exponent $t_{j,k}$ are arbitrary positive integer number, however, the plaintext in the MUL ciphertext can only have Jacobi symbol 1 and must not equal to 0, as discussed above. In order to successful design the EPOC system, Message Pre-coding Technique is adopted to transform all the non-zero data into the message with Jacobi symbol 1. Notice that the Legendre symbol $(-1|p) = (-1)^{(p-1)/2} = -1$, $(-1|q) = (-1)^{(q-1)/2} = -1$, $(1|p) = 1$, and $(1|q) = 1$, the Jacobi symbol $(-1|N) = (-1|pq) = (-1|p)(-1|q) = 1$ and $(1|N) = (1|pq) = (1|p)(1|q) = 1$.

We first show that the Jacobi symbol $(2|N) = 1$. As p', q' are two safe primes, $p = 2p' + 1$ and $p \equiv -1 \bmod 8$, and $q = 2q' + 1$ and $q \equiv -1 \bmod 8$. The Legendre symbol $(2|p) = (-1)^{(p^2-1)/8} = 1$ and $(2|q) = (-1)^{(q^2-1)/8} = 1$. As p and q are primes, the Jacobi symbol $(2|N) = (2|pq) = (2|p)(2|q) = 1$. So the Jacobi symbol $(2^k|N) = \prod_k (2|N) = 1$.

Next, we show that given an arbitrary positive integer A, it can be pre-coding as the following form:

$$A = 2^k \sum_{i=0}^{\phi-1} b_i 2^i,$$

where $b_i = \{+1, -1\}$, k is the largest non-negative integer so that $2^k|A$. For x^A, it can be represented as

$$x^A = \prod_{i=0}^{\phi-1} x^{b_i 2^{k+i}} = x^{t_\phi} \dots x^{t_1},$$

where $t_{i+1} = b_i 2^{k+i}$. Notice that $(t_{i+1}|N) = (b_i 2^{k+i}|N) = (b_i|N)(2^{k+i}|N) = (b_i|N)\prod_{k+i}(2|N) = 1$.

Algorithm 1. Message Pre-Coding Technique

```
Input: Positive integer number $A$ and large number $N$.
Output: Vector $(t_\phi, \dots, t_1)$, such that for each $t_i$, $(t_i|N) = 1$.
1 The number $A$ should be first converted into binary number,
  such that $A = \sum_j a_i 2^j$.;
2 Extract $2^k$ from $A$, denote as $A = 2^k B$, such that $B$ is odd inte-
  ger number, $k$ is the largest non-negative integer.;
3 Denote $B = \sum_{i=0}^{c} b'_i 2^i$. If $c < \phi$, $\phi - c - 1$ elements should be
  added into $B$, such that $B' = \sum_{i=0}^{\phi} b''_i 2^i$, when $i \le c$, $b''_i = b'_i$;
  when $c < i \le \phi$, $b''_i = 0$.;
4 for $i \leftarrow 0$ to $\phi - 1$ do
5     if $b''_i = 1$ and $b''_{i+1} = 0$ then
6         let $b_i = -1$ and $b_{i+1} = 1$.
7     if $b''_i = 1$ and $b''_{i+1} = 1$ then
8         let $b_i = 1$ and $b_{i+1} = 1$.
9 generate $(t_\phi, \dots, t_1)$, where $t_{i+1} = b_i 2^{k+i}$ $(0 \le i \le \phi - 1)$.
```

The Jacobi symbol of t_i is equal to $(t_{i+1}|N) = (b_i 2^{k+i}) = (b_i|N)(2^{k+i}|N) = (b_i|N)(2^{k+i}|N) = 1$. Therefore, an arbitrary integer A can be can transformed into $(t_\phi, \dots, t_1)$, where $(t_i|N) = 1$. The construction can be found in Algorithm 1.

Example 1. We use an example to illustrate our Message Pre-coding Technique. Let $p' = 3$ and $q' = 11$ and $\phi = 3$, compute $p = 2p' + 1 = 7$ and $q = 2q' + 1 = 23$. It can be easily verified that $p \bmod 8 = q \bmod 8 = -1$. The Legendre symbol $(3|p) = -1$ and $(3|q) = 1$. Suppose there exists two integers $A = 6$ and $B = 2$, the Jacobi symbol $(A|N) = (3|N)(2|N) = -1$ and $(B|N) = 1$. The integer A can be represented as $A = 2^1 \cdot \sum_{i=0}^{2} a_i 2^i$, where $(a_2, a_1, a_0) = (1, -1, 1)$. The integer B can be represented as $B = 2^0 \cdot \sum_{i=0}^{2} b_i 2^i$, where $(b_2, b_1, b_0) = (1, -1, -1)$.

4.4.2 Secure Exponent Calculation Protocol with Public Base

The *SEPB* protocol involves two parties, PCP and PCC. Consider PCP holds a public base x, an encrypted exponent $E_{pk_\alpha^\times}(y)$ and a partial MUL private key $\theta_{\alpha,1}$. PCC contains a partial MUL private key $\theta_{\alpha,2}$ and public key $pk_\alpha^\times$. The goal of *SEPB* protocol is to calculate $E_{pk_\alpha^\times}(x^y)$ without leaking y to PCC. The overall steps of *SEPB* protocol are shown as follows:

Step 1: PCP selects two random numbers $R_1, R_2 \in \mathbb{Z}_{2p'q'}$ with Jacobi symbol of 1, s.t., $gcd(R_1, R_2) = 1$, calculates $X_1 = E_{pk_\alpha^\times}(y) \cdot E_{pk_\alpha^\times}(R_1) = \{yR_1 h^{r_1 + r'_1} \bmod N,\ g^{r_1 + r'_1} \bmod N\}$ and $X_2 = E_{pk_\alpha^\times}(y) \cdot E_{pk_\alpha^\times}(R_2) = \{yR_2 h^{r_2 + r'_2} \bmod N, g^{r_2 + r'_2} \bmod N\}$. Then, calculates $C_1 = (g^{r_1 + r'_1})^{\theta_{\alpha,1}}$, $C_2 = (g^{r_2 + r'_2})^{\theta_{\alpha,1}}$, and sends $X'_1 = \{yR_1 h^{r_1 + r'_1}, C_1\}$, $X'_2 = \{yR_2 h^{r_2 + r'_2}, C_2\}$ and x to PCC.

Step 2: PCC firstly calculates $C'_1 = C_1^{\theta_{\alpha,2}} = h^{r_1 + r'_1} \bmod N$ and $C'_2 = C_2^{\theta_{\alpha,2}} = h^{r_2 + r'_2} \bmod N$, computes $T_1 = yR_1 h^{r_1 + r'_1} / C'_1 \bmod N = yR_1$, $T_2 = yR_2 h^{r_2 + r'_2} / C'_2 \bmod N = yR_2$, and $h_1 = x^{T_1}$ and $h_2 = x^{T_2}$. Then, PCC encrypts h_1 and h_2 by using $pk_\alpha^\times$ and denotes it as $H_1 = E_{pk_\alpha^\times}(h_1) = \{x^{yR_1} h^{r''_1} \bmod N, g^{r''_1} \bmod N\}$ and $H_2 = E_{pk_\alpha^\times}(h_2) = \{x^{yR_2} h^{r''_2} \bmod N, g^{r''_2} \bmod N\}$, and sends H_1 and H_2 to PCP.

Step 3: Since $gcd(R_1, R_2) = 1$, there exists integers a, b, s.t, $aR_1 + bR_2 = 1$. Once H_1 and H_2 are received, PCP can calculate the following formula to gain the encrypted x^y:

$$\begin{aligned} {H_1}^a \cdot {H_2}^b &= \{x^{yR_1 a} h^{r''_1 a} \cdot x^{yR_2 b} h^{r''_2 b}, g^{r''_1 a} \cdot g^{r''_2 b}\}, \\ &= E_{pk_\alpha^\times}(x^{yR_1 a} \cdot x^{yR_2 b}) = E_{pk_\alpha^\times}(x^y). \end{aligned}$$

4.4.3 Construction of Enhanced EPOC

In our enhanced EPOC, only the needed coefficients and exponents are encrypted with MUL public key, i.e., no zero elements are needed to encrypt under the **MulEnc** algorithm. Notice that the **MulEnc** can only encrypt the message with the Jacobi symbol of 1 in order to guarantee the security of the ciphertext. Before encryption, all exponent of every monomials $t_{j,k}$ should be transformed into the $t'_{j,k,l}$ $(l = 1, \dots, \phi)$ with Jacobi symbol of 1 by using message precoding technique. The value of all the coefficient in function $\mathcal{F}$ should be selected by user, such that the Jacobi symbol of C_j $(j = 1, \dots, K)$ is 1.

After transformation, the outsourced function $\mathcal{F}$ used in basic EPOC should be transformed into the new function

$$\mathcal{F}' : o = \sum_{j=1}^{\kappa} C_j \left(\prod_{k=1}^{\gamma} \prod_{l=1}^{\phi} x_k^{t'_{j,k,\phi}} \right).$$

Once the encrypted user α's queries $E_{pk_\alpha^\times}(C_j)$ and $E_{pk_\alpha^\times}(t'_{j,k,l})$ $(j = 1, \dots, \kappa; k = 1, \dots, \gamma; l = 1, \dots, \phi)$ are received, PCP needs to use all these queries in conjunction with the public dataset S to compute the results privately according to the user's outsourced function $\mathcal{F}$. The calculation can be described as following stages:

1. *Secure monomials calculation stage (Stage One)*. In this stage, PCP needs to calculate the monomials over the public dataset which formalized as $C_j \cdot \prod_{k=1}^{\gamma} x_k^{t'_{j,k,1}} \dots x_k^{t'_{j,k,\phi}}$. As all the $t'_{j,k,1}, \dots, t'_{j,k,\phi}$ are stored as encrypted form, the monomials cannot be computed directly. For every i, j, k, l $(i = 1, \dots, \tau; j = 1, \dots, \kappa; k = 1, \dots, \gamma; l = 1, \dots, \phi)$, PCP first computes $E_{pk_\alpha^\times}(a_{i,k}^{t'_{j,k,l}})$ by calling:

$$E_{pk_\alpha^\times}\left(a_{i,k}^{t'_{j,k,l}}\right) = SEPB(a_{i,k}, E_{pk_\alpha^\times}(t'_{j,k,l})). \quad (2)$$

Then, PCP uses $E_{pk_\alpha^\times}(C_j)$ and $E_{pk_\alpha^\times}(a_{i,k}^{t'_{j,k,l}})(l = 1,\dots,\phi)$ to calculate

$$\begin{aligned} E_{pk_\alpha^\times}\left(C_j \cdot \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}}\right) \\ = E_{pk_\alpha^\times}(C_j) \cdot \prod_{k=1}^{\gamma}\prod_{l=1}^{\phi} E_{pk_\alpha^\times}(a_{i,k}^{t_{j,k}}). \end{aligned}$$

and the results can be used for further processing.

2. *Secure outsourced function calculation stage (Stage Two).* Once the secure monomials calculation phase is finished, PCP needs to use $E_{pk_\alpha^\times}(C_j \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}})$ to compute the final encrypted results $E_{pk_\alpha^+}(o_i)$ for every $i(i = 1,\dots,\tau)$ according to the outsourced function $\mathcal{F}$. Firstly, $E_{pk_\alpha^\times}(C_j \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}})$ should be transferred into $E_{pk_\alpha^+}(C_j \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}})$ by calling **MulToMix** and **MixToAdd** of the SHED. After that, additive operation over the plaintexts can be easily achieved by multiplication of the ADD ciphertexts as follows:

$$\begin{aligned} E_{pk_\alpha^+}(o_i) &= E_{pk_\alpha^+}\left(\sum_{j=1}^{\kappa} C_j \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}}\right) \\ &= \prod_{j=1}^{\kappa} E_{pk_\alpha^+}(C_j \prod_{k=1}^{\gamma} a_{i,k}^{t'_{j,k,1}} \dots a_{i,k}^{t'_{j,k,\phi}}). \end{aligned}$$

After that, the user α can decrypt the results o_i same procedure as basic EPOC.

4.5 Full EPOC for Achieving Level-III Privacy

In the enhanced EPOC, only the needed dimension are involved in the calculation. Indeed, it will expose the adversary $\mathcal{A}^*$ some extra information about which dimension is used in the function. Moreover, the number of monomials in the function also expose some extra information. In our full EPOC, all the dimension should be used in the function. If the user does not need the dimension in the function, he simply denote the value of exponent of redundancy dimension as 0. Furthermore, some redundancy monomials should be added to the original function $\mathcal{F}$, and these redundancy monomials should not change the output value of the original function $\mathcal{F}$. The goal of the function transformation is to avoid $\mathcal{A}^*$ to guess how many monomials contained in $\mathcal{F}$.

In order to expose nothing about the outsourced functions (even the patten of the function), full PPOC should be designed in this section. Before executing the full EPOC, some redundancy monomials should be added in the function. The function used in full EPOC contains fix K monomials, i.e., $K - \kappa$ redundancy monomials are added to the original function $\mathcal{F}$. The goal of the function transformation is to avoid adversary to guess how many monomials contained in the original outsourced function. Moreover, as our SHED cannot encrypt message "0" under $pk_\alpha^\times$, all the coefficient of the redundancy monomials should be ($C_j = +1$ and $C_{j+1} = -1$) or ($C_j = -1$, $C_{j+1} = -1$ and $C_{j+2} = 2$).

In full EPOC, all the dimensions will be involved in the calculation, and the value of exponent of redundancy dimension should be 0. It prevents PCC and PCP to know how many dimensions are used in each monomials. As SHED cannot encrypt message '0' under $pk_\alpha^\times$, a new technique called Message Extending and Coding Technique are proposed in Algorithm 2 to solve this problem.

Algorithm 2. MESSAGE EXTENDING AND CODING TECHNIQUE

```
Input: Integer number $A$ and large number $N$.
Output: Vector $(t_\phi,\dots,t_1)$ and $(t'_\phi,\dots,t'_1)$, such that for each
        $t_i$ and $t'_i$, $(t_i|N) = 1$ and $(t'_i|N) = 1$
1 The number $A$ should be converted into two number $A_1$ and
  $A_2$, such that: if $A \neq 1$, then $A_1 = A - 1$ and $A_2 = 1$; if $A = 1$,
  then $A_1 = A + 1 = 2$ and $A_2 = -1$.;
2 if $A_1 = -1$ then
3     coding $A_1$ as $(\delta_\phi,\dots,\delta_1)$ such that $\delta_\phi = -2^{\phi-1}$,
      $\delta_{j+1} = 2^j (0 \leq j \leq \phi - 1)$.
4 else
5     coding $A_1$ as $(\delta_\phi,\dots,\delta_1)$ using message pre-coding
      technique.
6 if $A_2 = -1$ then
7     coding $A_2$ as $(\delta'_\phi,\dots,\delta'_1)$ such that $\delta'_\phi = -2^{\phi-1}$,
      $\delta'_{j+1} = 2^j (0 \leq j \leq \phi - 1)$.
8 else
9     coding $A_2$ as $(\delta'_\phi,\dots,\delta'_1)$ using message pre-coding
      technique.
```

Before the **MulEnc** encryption, all exponent of monomials $t_{j,k}$ should be transformed into the $(\delta_{j,k,\phi}\dots\delta_{j,k,1})$ and $(\delta'_{j,k,\phi}\dots\delta'_{j,k,1})$ by using message extending and coding technique. Notice that $t_{j,k} = \sum_{l=1}^{\phi}(\delta_{j,k,l} + \delta'_{j,k,l})$.

After transformation, the outsourced function $\mathcal{F}$ used in basic EPOC can be transformed into the new function $\mathcal{F}''$ as follows:

$$\mathcal{F}'' : o = \sum_{j=1}^{K} C_j \cdot \left(\prod_{k=1}^{\gamma}\prod_{l=1}^{\phi} x_k^{\delta_{j,k,l}} \cdot x_k^{\delta'_{j,k,l}}\right).$$

It is worth noting that the final results calculated by $\mathcal{F}$ and $\mathcal{F}''$ are the same. The construction of full EPOC is similar to that of the enhanced EPOC. The difference is that we use the outsourced function $\mathcal{F}''$ instead of $\mathcal{F}$ in the enhanced version (see Section 4.4.3).

Example 2. We use the following example to illustrate the correctness of the full EPOC. Denote $p' = 3$ $q' = 11, K = 2$, and $\phi = 3$, compute $p = 2p' + 1 = 7$ and $q = 2q' + 1 = 23$. Suppose given a public data $\vec{\mathbf{a}} = (a_1, a_2) = (2, 3)$, the user wants to calculate the function $\mathcal{F} : o = a_1^3$. Before uploading the coefficients and expoents, $\mathcal{F}$ must be transformed into $\mathcal{F}''$ due to the security concern. Firstly, one $0a_1^0a_2^0$ should pad to $\mathcal{F}$ such that $o = a_1^3a_2^0 + 0a_1^0a_2^0 = a_1^3a_2^0 + a_1^0a_2^0 - a_1^0a_2^0$. After that, all the exponent should be coded using message extending and coding technique, i.e., 3 should be transformed into two integers 2 and 1, and then coding them as $(2^3, -2^2, -2^1)$ and $(2^2, -2^1, -2^0)$ respectively. The integer

0 should be transformed into -1 and 1, and then coding them as $(-2^2, 2^1, 2^0)$ and $(2^2, -2^1, -2^0)$ respectively. We can easily verified that $(2^3 - 2^2 - 2^1)+$ $(2^2 - 2^1 - 2^0) = 2 + 1 = 3$ and $(-2^2 + 2^1 + 2^0) + (2^2 - 2^1 - 2^0) = -1 + 1 = 0$. Finally, $\mathcal{F}_2$ can be transformed into $\mathcal{F}'' : o' = a_1^{2^3} a_1^{-2^2} a_1^{-2^1} a_2^{-2^2} a_2^{2^1} a_2^{2^0} + a_1^{-2^2} a_1^{2^1} a_1^{2^0} a_2^{-2^2} a_2^{2^1} a_2^{2^0} - a_1^{-2^2} a_1^{2^1} a_1^{2^0} a_2^{-2^2} a_2^{2^1} a_2^{2^0}$. All the coefficients and exponents should be encrypted as MUL ciphertext and send to PCP for calculation. The final output of $\mathcal{F}$ and $\mathcal{F}''$ are the same, i.e., $o' = o = 8$.

Remark 1. If the user only wants to get the top-K values from the computed results, privacy-preserving top-K protocol [14] can be used between PCP and PCC to select K encrypted value from the τ encrypted values. Only the selected K encrypted values are needed to be sent back to the user which can reduce both communication cost and computational overhead.

Remark 2. In order to prevent a users privilege from being abused, our private key revocation solution is easy and simple: once the user needs to be revoked, the PCC does not provide the **AddPDec1** service to the revoked user. As only holding the partial ADD private key λ_2, the revoked user cannot decrypt the encrypted final results.

5 Security Analysis

In this section, we will analyze the security of SHED and then show that our EPOCs meet the three privacy levels defined in 3.2.

5.1 The Analysis of SHED

5.1.1 The Existence of Large Prime Number Selection

In the **KeyGen** algorithm, we need to select two large safe prime numbers p', q', compute $p = 2p' + 1$ and $q = 2q' + 1$, such that $p \equiv -1 \bmod 8$ and $q \equiv -1 \bmod 8$. We call a prime number p' a Sophie Germain prime if $2p' + 1$ is also prime. Here, our selection can be guaranteed by the following theorem.

Theorem 1. *Let $p' \equiv 3 \bmod 4$ be prime. $2p' + 1$ is also prime if and only if $2p' + 1$ divides $M_{p'}$, where $M_{p'}$ is Mersenne prime.*

Proof. The theorem was stated by Euler in 1750 and proved by Lagrange in 1775. The proof can be found in [15]. □

Notice if $p' \equiv 3 \bmod 4$, then $p \equiv 2p' + 1 = -1 \bmod 8$. So we can use the theorem 1 directly to select p' and q' which can satisfy the prime constrictions in our **KeyGen** algorithm.

5.1.2 The Existence of ADD Private Key Splitting

Here, we show how to randomly separate the ADD private key into two parts. Due to $gcd(\lambda, N^2) = 1$, thus $\exists$ s, s.t. $s \equiv 0 \bmod \lambda$ and $s \equiv 1 \bmod N^2$ hold at the same time (thanks to Chinese remainder theorem [16], $s = \lambda \cdot$ $(\lambda^{-1} \bmod N^2)$ $\bmod \lambda N^2$). Thus, we only need to randomly choose λ_1 and λ_2, s.t., $\lambda_1 + \lambda_2 = s$.

5.1.3 Security of SHED

The security of our SHED can be guaranteed by the following theorem.

Theorem 2. *The SHED scheme is semantic secure under Decisional Diffie-Hellman (DDH) problem [12].*

Proof. The newly proposed SHED is based on a switchable homomorphic encryption mechanism and the original scheme is secure under the standard chosen-plaintext attack (CPA) model which based on the Decisional Diffie-Hellman (DDH) problem [12]. In order to support partially decryption, $(2, 2)$-secret sharing method will be used to randomly separate sk^+ into two parts. Due to the character of the Shamir secret sharing techniques [17], it is hard for adversary to recover the original secret (the ADD private key) even the adversary gets the one share (the adversary can also select the share all by himself). □

Next, we will show that our SHED can resist the following four kinds of analyses.

Against Original Ciphertext Analysis: The orignial (Paillier) ADD ciphertext $E_{pk^+}(m)$ and (ElGamal) MUL ciphertext $E_{pk^\times}(m)$ are secure under the CPA model [18], [19]. Also, the ciphertext $E_{pk^+, pk^\times}(m)$ in SHE is also secure under the CPA model [12].

Against Partially Decrypted Ciphertext Analysis: Given $CT(m) = r'^{N\lambda_1}(1 + mN\lambda_1) \bmod N^2$ to adversary for analysis, it is still hard for adversary to get m or λ_1 due to the hardness of Partial Discrete Logarithm (PDL) problem [18].

Against Partially Decrypted Ciphertext Relevant Analysis: Given $CT(m) = r'^{N\lambda_1}(1 + mN\lambda_1) \bmod N^2$ and $CT'(m) = r''^{N\lambda_1}(1 + mN\lambda_1) \bmod N^2$ to adversary for analysis, it is still hard for adversary to distinguish these two partially decrypted ciphertexts. It is because that r' and r'' are randomly chosen from a same distribution (r' and r'' are indistinguishable) and both the ciphertexts are encrypted under the same encryption function.

5.2 Security Model Definition

Here we recall the security model for securely realizing an ideal functionality in the presence of non-colluding semi-honest adversaries. For simplicity, we do it for the specific scenario of our functionality, which involve three parties, the message which need to be protected from the challenge user (a.k.a. "D_U"), and both PCP (a.k.a. "S_1") and PCC (a.k.a."S_2"). We refer the reader to [20] for the general case definitions.

Let $\mathcal{P} = (D_U, S_1, S_2)$ be the set of all protocol parties. We consider three kinds of adversaries $(\mathcal{A}_{D_U}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ that corrupt D_U, S_1 and S_2, respectively. In the real world, D_U runs on input x and y (with additional auxiliary inputs z_x and z_y), while S_1 and S_2 receive auxiliary inputs z_1 and z_2. Let $H \subset \mathcal{P}$ be the set of honest parties. Then, for every $P \in H$, let out_P be the output of party P, whereas if P is corrupted, i.e. $P \in \mathcal{P} \backslash H$, then out_P denotes the view of P during the protocol Π.

For every $P^* \in \mathcal{P}$, the *partial view* of P^* in a real-world execution of protocol Π in the presence of adversaries $\mathcal{A} = (\mathcal{A}_{D_U}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ is defined as

$$REAL^{P^*}_{\Pi, \mathcal{A}, H, \mathbf{z}}(x, y) = \{out_P : P \in H\} \cup out_{P^*}.$$

In the ideal world, there is an ideal functionality $\mathbf{f}$ for a function f and the parties interact only with $\mathbf{f}$. Here, the challenge user sends x and y to $\mathbf{f}$. If any of x or y is $\perp$, then $\mathbf{f}$ returns $\perp$. Finally, $\mathbf{f}$ returns $f(x,y)$ to the challenge user. As before, let $H \subset \mathcal{P}$ be the set of honest parties. Then, for every $P \in H$, let out_P be the output returned by $\mathbf{f}$ to party P, whereas if P is corrupted, out_P is the same value returned by P.

For every $P^* \in \mathcal{P}$, the partial view of P^* in an ideal-world execution in the presence of independent simulators Sim= $(\mathsf{Sim}_{D_U}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ is defined as

$$IDEAL^{P^*}_{\mathbf{f},\mathsf{Sim},H,\mathbf{z}}(x,y) = \{out_P : P \in H\} \cup out_{P^*}.$$

Informally, a protocol Π is considered secure against non-colluding semi-honest adversaries if it *partially emulates*, in the real world, an execution of $\mathbf{f}$ in the ideal world. More formally,

Definition 4. *Let $\mathbf{f}$ be a deterministic functionality among parties in $\mathcal{P}$. Let $H \subset \mathcal{P}$ be the subset of honest parties in $\mathcal{P}$. We say that Π securely realizes $\mathbf{f}$ if there exists a set Sim= $(\mathsf{Sim}_{D_U}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ of PPT transformations (where $Sim_{D_1} = Sim_{D_1}(\mathcal{A}_{D_1})$ and so on) such that for all semi-honest PPT adversaries $\mathcal{A} = (\mathcal{A}_{D_U}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$, for all inputs x, y and auxiliary inputs **z**, and for all parties $P \in \mathcal{P}$ it holds*

$$\{REAL^{P^*}_{\Pi,\mathcal{A},H,\mathbf{z}}(\lambda,x,y)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{IDEAL^{P^*}_{\mathbf{f},\mathsf{Sim},H,\mathbf{z}}(\lambda,x,y)\}_{\lambda\in\mathbb{N}}$$

where $\overset{c}{\approx}$ denotes computational indistinguishability.

5.3 Security of SEPB

Theorem 3. *The SEPB protocol is securely computes exponent operation with private exponent in the presence of semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{D_U}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. Here, we show how to construct the three independent simulators $\mathsf{Sim}_{D_U}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2}$.

Sim_{D_U} receives y as input and then simulates $\mathcal{A}_{D_U}$ as follows: it generates encryption $E_{pk^\times_\alpha}(y) = \mathbf{MulEnc}$ $(pk^\times_\alpha, y)$ of y. Finally, it returns $E_{pk^\times_\alpha}(y)$ to $\mathcal{A}_{D_U}$ and outputs $\mathcal{A}_{D_U}$'s entire view. The view of $\mathcal{A}_{D_U}$ consists of the encrypted data. The views of $\mathcal{A}_{D_U}$ in the real and the ideal executions are indistinguishable due to the security of MUL ciphertext [19].

Sim_{S_1} simulates $\mathcal{A}_{S_1}$ as follows: First, it generates (fake) encryptions of the inputs $E_{pk^\times_\alpha}(y)$ by running **MulEnc** on randomly chosen $\hat{\theta}_{\alpha,1}$, randomly generates $\hat{R}_1, \hat{R}_2 \in \mathbb{Z}_N$, calculates $\hat{X}'_1$ and $\hat{X}'_2$. Then, Sim_{S_1} sends the encryption $\hat{X}'_1$ and $\hat{X}'_2$ to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} returns $\perp$. The view of $\mathcal{A}_{S_1}$ consists of the encrypted data it creates. In both the real and the ideal execution, he receives the output the encryptions $\hat{X}'_1$ and $\hat{X}'_2$. In the real world this is guaranteed by the fact that the user is honest and the semantically security of SHED. The views of $\mathcal{A}_{S_1}$ in the real and the ideal executions are indistinguishable.

Sim_{S_2} simulates $\mathcal{A}_{S_2}$ as follows: it randomly chooses $\hat{h}_1$ and $\hat{h}_2$, uses the **MulEnc** to get $\hat{H}_1$ and $\hat{H}_2$, and then sends the encryption to $\mathcal{A}_{S_2}$. If $\mathcal{A}_{S_2}$ replies with $\perp$, then Sim_{S_2} returns $\perp$. The view of $\mathcal{A}_{S_2}$ consists of the encrypted data it creates. In both the real and the ideal execution he receives the output the encryptions $\hat{H}_1$ and $\hat{H}_2$. In the real world this is guaranteed by the fact that the user is honest and the semantically security of SHED. The views of $\mathcal{A}_{S_2}$ in the real and the ideal executions are indistinguishable. □

5.4 Security of Basic EPOC

Here, we analyze that our basic EPOC can resist adversary $\mathcal{A}^*$ described in Section 3.2 and achieve Level-I privacy.

If $\mathcal{A}^*$ eavesdrop the transmission link between the challenge user and PCP, the encrypted coefficients $E_{pk^+_\alpha}(C_j)$ and the final results $E_{pk^+_\alpha}(o_i)$ are got by $\mathcal{A}^*$. Moreover, the intermediated ciphertext $E_{pk^+_\alpha}(C_j a_{i,1}^{t_{j,1}} \ldots a_{i,\gamma}^{t_{j,\gamma}})$ and $E_{pk^+_\alpha}(o_i)$ transmitted between PCP and PCC may also be eavesdropped by $\mathcal{A}^*$. Because all the data in the transmission link are transmitted as encrypted form, it is impossible for $\mathcal{A}^*$ to decrypt these ciphertexts without knowing the user's ADD private key due to IND-CPA secure of (Paillier) ADD ciphertext [18]. Next, if PCP is compromised by $\mathcal{A}^*$, $\mathcal{A}^*$ can get all the ADD ciphertexts. Even holding the challenge user's partial ADD private key, $\mathcal{A}^*$ still cannot get all the queries and final results due to the semantically security of SHED. Moreover, suppose PCC is compromised by $\mathcal{A}^*$, $\mathcal{A}^*$ can get all the user's ADD ciphertexts and their partial MUL private keys (all the ADD & MUL private keys are randomly separated). According to the security of SHED, all the user's final results cannot be decrypted by $\mathcal{A}^*$ even getting the corresponding partial ADD private keys. Furthermore, if $\mathcal{A}^*$ compromises some of the users to get some user's partial ADD private keys, he still cannot decrypt the challenge user's ADD ciphertexts. It is because different user's ADD private keys in our system are unrelated (all the system ADD private keys in SHED are selected randomly and independently, and partial ADD private keys are randomly separated). Furthermore, even the $\mathcal{A}^*$ can know which original data are accessed, the coefficients of the outsourced function and the final results can still be protected due to the CPA-secure of the ADD ciphertext. In all, $\mathcal{A}^*$ cannot know the challenge user's sensitive information which satisfies Level-I privacy which defined in Section 3.2.

5.5 Security of Enhanced EPOC

Here, we analyze that our enhanced EPOC can resist the adversary $\mathcal{A}^*$. The analysis is similar to that of basic EPOC list in Section 5.4. The difference is that *MPT* technique are used to transform all the exponents and coefficients of the outsourced function into the data with the Jacobi symbol of 1, and then the transformed queries are encrypted under the MUL public key before sending to PCP. More extra the intermediate ciphertexts (X'_1, X'_2, H_1, H_2 in *SEPB*) are also transmitted between PCP and PCC in the enhanced EPOC. Due to the semantically security of SHED, $\mathcal{A}^*$ cannot decrypt any of the ADD ciphertexts and MUL ciphertexts (IND-CPA secure of El-Gamal ciphertext [19]). Moreover, even $\mathcal{A}^*$ get the user's encrypted query, he cannot still decrypt the results because PCP only hold the partial of the

TABLE 2
The Performance of SHED Cryptosystem (1,000-Times for Average, 80-Bits Security Level)

Algorithm	AddEnc	MulEnc	AddPDec1	AddPDec2	AddDec	MulDec	AddToMix	MulToMix	MixToAdd
Time	7.865 ms	10.122 ms	23.156 ms	22.864 ms	8.209 ms	2.801 ms	10.861 ms	12.459 ms	20.128 ms

MUL private key (the whole MUL private key are randomly divided by running **KeyGen** algorithm). Furthermore, the adversary $\mathcal{A}^*$ can still get nothing from the decryption results even he can partially decrypt to get the intermediate results. Due to the security of *SEPB* protocol (see Section 5.3 for detailed analysis), some random numbers are used to randomize each of the intermediate results. For the challenge user α, $\mathcal{A}^*$ cannot decrypt $E_{pk_\alpha^\times}(C_j)$ and $E_{pk_\alpha^\times}(t'_{j,k,l})$ without knowing domain α's MUL private key (different MUL private keys are unrelated and independence), and cannot decrypt the final results $E_{pk_\alpha^+}(o_i)$ without knowing domain α's ADD private key (different domain's MUL encrypted queries and ADD encrypted results are encrypted under different modular N). In all, $\mathcal{A}^*$ cannot know the challenge user's sensitive information which satisfies Level-II privacy which defined in Section 3.2.

5.6 Security of Full EPOC

The security of full EPOC is similar to that of the enhanced version. The difference is that *MEC* technique are used to transform all the coefficients and exponents into the message with the Jacobi symbol of 1. The information of the MUL encrypted queries will not be leaked to $\mathcal{A}^*$. As all the data (dimensions) are used for calculation, it is impossible for $\mathcal{A}^*$ guess in which dimensions or data the challenge user are interested. Moreover, all outsourced function have fix numbers of monomials (constant-length, with some redundancy monomials added to the original function), it is hard for $\mathcal{A}^*$ to guess the format of the outsourced function. In all, $\mathcal{A}^*$ cannot know the challenge user's sensitive information which satisfies Level-III privacy which defined in Section 3.2.

6 Performance Analysis

In this section, we analyze the performance of the three versions of EPOC in terms of both the computation cost and the communication overhead.

6.1 Experimental Analysis

In this section, we evaluate computation cost and communication overhead of the proposed EPOC by using a custom simulator built in Java. The experiment is run on a PC with 3.6 GHz eight-core processors and 12 GB memory. Both synthetic dataset and real dataset will be used to test the performance of EPOC.

6.1.1 Synthetic Dataset

We use synthetic dataset which is randomly generated to test the three versions of our EPOC. This dataset contains 500 tuples. Each tuple contains 10 attributes. Each attribute of tuple is randomly picked from 1 to 5,000. There are four factors which affect the total running time of the EPOC: i) the number of monomials contained in the outsourced functions (NNF); ii) the number of tuples in the synthetic dataset (NUM); iii) the dimension of the vector existed in the dataset (DIM); iv) the maximum length of the vector expansion in the message pre-coding technique (MLV).

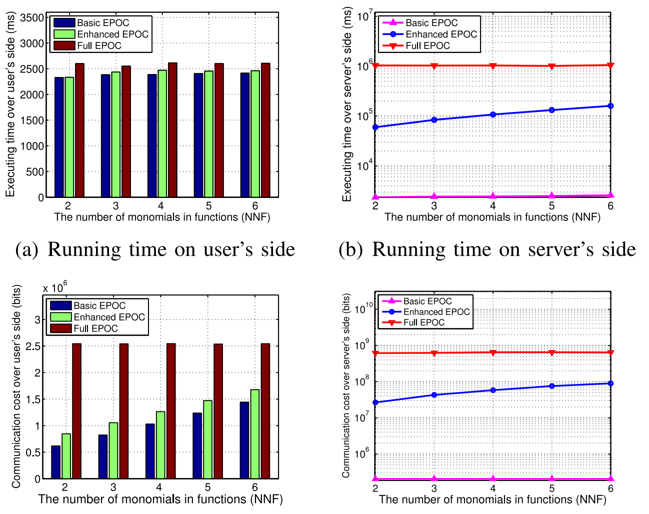
We first test the performance of SHED scheme and *SEPB* protocol, and list the results in Tables 2 and 3 respectively. We denote N as 1,024 bits in SHED cryptosystem to achieve 80-bit security levels [21]. Then, we plot both the computation cost and communication overhead which vary with NNF in Fig. 2. Because more queries (coefficients of the function and the exponent of every monomials of the function) are encrypted and sent to PCP for computation, both the computation cost and communication overhead in user's side and server's side are increase with the NNF in the basic and enhanced EPOC. The enhanced EPOC is much more costly than the basic EPOC. Because *SEPB* protocol in the enhanced EPOC is called while no sub-protocol are called in the basic EPOC. Furthermore, the full EPOC is not vary with NNF because fix number of monomials are involved in this kind of EPOC. In Fig. 3, we plot both the computation cost and communication overhead which vary with NUM. From the figure we can see that both the computation cost and communication overhead in user's side and server's side are increased with the NUM in these three EPOC. It is because more tuples will be processed in PCP and more encrypted results will be transmitted back to user. In Fig. 4, we plot both the computation cost and communication overhead which vary with DIM. From the figure we can see that the computation cost and communication overhead over the user side of the enhanced EPOC are increased with DIM while the basic EPOC is not changed much with DIM. It is because more *SEPB* are needed in the enhanced EPOC while only some exponential calculation over plaintext are involved in the basic EPOC. Moreover, some fix numbers of dimensions are involved in the full EPOC such that the full EPOC is free from DIM. In Fig. 5, we plot both the computation cost and communication overhead which vary with MLV. The basic EPOC is free from MLV because no expansion are needed in the base version. Both the enhanced and full EPOC are increased with MLV.

6.1.2 Real Dataset

Before using real dataset to test the efficiency of the EPOC, we first choose a baseline for comparison: do function

TABLE 3
The Performance of Algorithm & Sub-Protocol (1,000-Times for Average, 80-Bits Security Level)

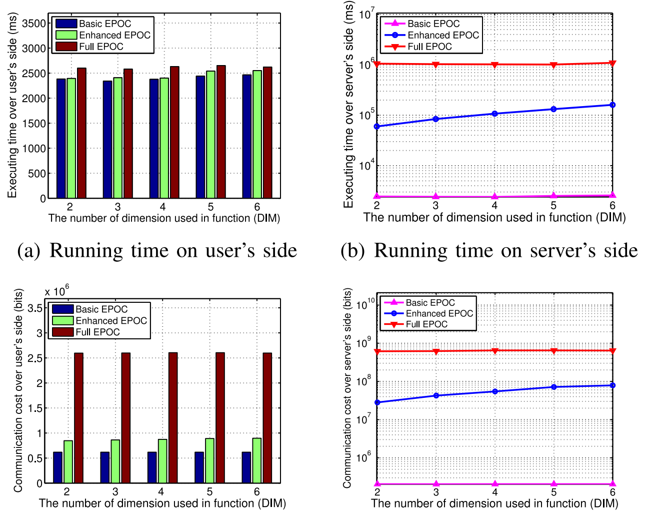
	PCP compute.	PCC compute.	Commu.
MixToAdd (SHED)	8.973 ms	11.199 ms	0.874 KB
SEPB	11.791 ms	12.075 ms	1.497 KB



(a) Running time on user's side (b) Running time on server's side

(c) Communication cost on user's side (d) Communication cost on server's side

Fig. 2. Experiment analysis vary with NNF in the synthetic dataset (NUM = 100, DIM = 2, $\phi = 5$).

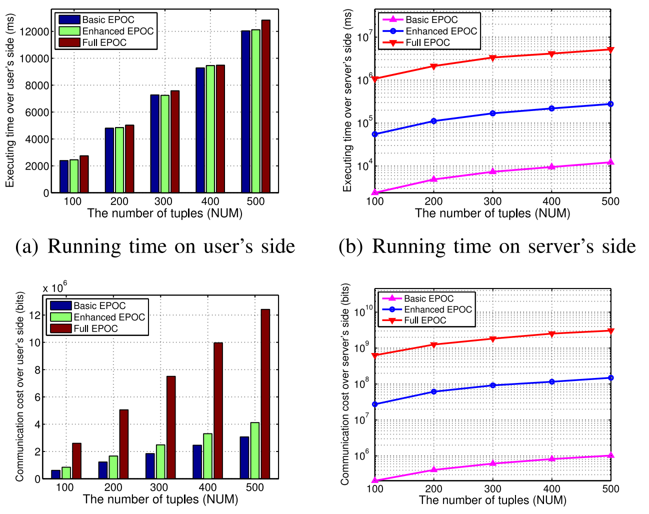


(a) Running time on user's side (b) Running time on server's side

(c) Communication cost on user's side (d) Communication cost on server's side

Fig. 4. Experiment analysis vary with DIM in the synthetic dataset (NUM = 100, NNF = 2, $\phi = 5$).

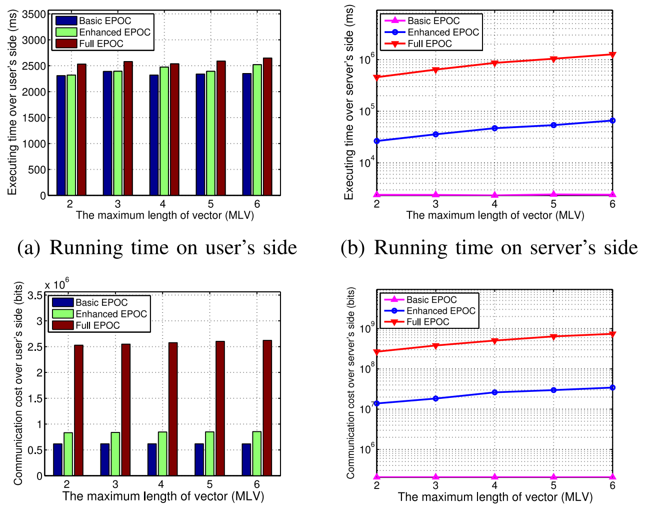
calculation over public database without any privacy protection technique, i.e., the user sends the function to PCP for calculation without any protection, and the calculated results are directly sent back to the user after calculation. Next, we use the ISTANBUL STOCK EXCHANGE Data Set from UCI Machine Learning Repository [22] as the real dataset to test the performance of EPOC. There are 536 tuples includes returns of Istanbul Stock Exchange with seven other international indexs [including S&P 500 Index (New York Stock Exchange); Deutscher Aktien Index (Frankfurt Stock Exchange); FTSE 100 Index (London Stock Exchange); Nikkei Index (Tokyo Stock Exchange); Bovespa Index (Brasil Sao Paulo Stock Exchange); MSCI Europe Index; MSCI Emerging Markets Index] from Jun 5, 2009 to Feb 22, 2011. We use the seven attributes to test the efficiency of EPOC. Because SHED can only encrypt positive integers, all the attributes in the dataset should be normalized into integer before running EPOC. The transformed dataset should be used to test both the computation cost and communication overhead of between the baseline and different EPOC. After that, we list all the comparison results in Table 4. No encryption is needed which cost no computational resource in user's side for the baseline while only the function's coefficients are needed to be encrypted and sent to the PCP in the user side of the basic EPOC. The ciphertext homomorphic calculation in basic version consumes more computational resource than the directly plaintext computation in baseline. In the enhanced EPOC, both the coefficients and the encrypted exponent $t_{i,k}$ of a monomials independent variable are coded and encrypted by user, and then sent to PCP which involves more computation cost to the user. More computation and communication resources are needed between PCP and PCC due to the necessity of executing *SEPB*. In the full version of EPOC, all



(a) Running time on user's side (b) Running time on server's side

(c) Communication cost on user's side (d) Communication cost on server's side

Fig. 3. Experiment analysis vary with NUM in the synthetic dataset (NNF = 2, DIM = 2, $\phi = 5$).



(a) Running time on user's side (b) Running time on server's side

(c) Communication cost on user's side (d) Communication cost on server's side

Fig. 5. Experiment analysis vary with MLV in the synthetic dataset (NUM = 100, NNF = 2, DIM = 2).

TABLE 4
Experiment over Real Dataset (NUM = 536, DIM = 3, $\kappa = 3$, K = 10, $\phi = 5$, 80-Bit Security Level)

	Computation (user)	Computation (server)	Communication (user & PCP)	Communication (PCC & PCP)
Baseline	N.A.	0.283 s	71.922 KB	70.347 KB
Basic EPOC	12.386 s	13.126 s	535.771 KB	133.842 KB
Enhanced EPOC	12.631 s	10.943 min	674.772 KB	41.352 MB
Full EPOC	13.028 s	93.748 min	1.585 MB	386.051 MB

TABLE 5
Theoretical Analysis of EPOC

EPOC	T. Comput. (/mul)	T. Commun. (/bits)	User Comput. (/mul)	User & PCP Commu.(/mul)	User & PCP	PCP & PCC
Basic	$\mathcal{O}(\kappa\tau\|N\|)$	$\mathcal{O}((\kappa+\tau)\|N\|)$	$\mathcal{O}((\kappa+\tau)\|N\|)$	$\mathcal{O}((\kappa+\tau)\|N\|)$	One Round	One Round
Enhanced	$\mathcal{O}(\kappa\gamma\tau\phi\|N\|)$	$\mathcal{O}(\kappa\gamma\tau\phi\|N\|)$	$\mathcal{O}((K\gamma\phi+\tau)\|N\|)$	$\mathcal{O}((\kappa\gamma\phi+\tau)\|N\|)$	One Round	$\mathcal{O}(\kappa\gamma\tau\phi)$ Rounds
Full	$\mathcal{O}(K\gamma\tau\phi\|N\|)$	$\mathcal{O}(K\gamma\tau\|N\|)$	$\mathcal{O}((K\gamma\phi+\tau)\|N\|)$	$\mathcal{O}((K\gamma\phi+\tau)\|N\|)$	One Round	$\mathcal{O}(K\gamma\tau\phi)$ Rounds

the dimensions will be used in the calculation that involves more computation cost and communication overhead. Moreover, because some redundancy monomials are used for calculation, more *SEPB* and multiplications are involved which introduced more computational cost and communication overhead.

6.2 Theoretical Analysis

6.2.1 Computational Cost

In SHED scheme, let us assume that one regular exponentiation operation with an exponent of length $|N|$ requires $1.5|N|$ multiplications [23], e.g, the length of λ is $|N|$ and compute g^{λ} requires $1.5|N|$ multiplications. The length of λ_1 and λ_2 are $3|N|$ and compute g^{λ_1} or g^{λ_2} requires $4.5|N|$ multiplications. Due to exponentiation operation is much costly than the addition and multiplication operation, we ignore some fix numbers of addition and multiplication operations in our analysis. For SHED, it costs $1.5|N|$ multiplications to use **AddEnc**, **AddDec** and **MulDec** algorithm. Moreover, it needs $3|N|$ multiplications to run **MulEnc** algorithm. Furthermore, **AddToMix**, **MulToMix**, **AddPDec1** and **AddPDec2** algorithm needs $4.5|N|$ multiplications while **MulToMix** needs $7.5|N|$ multiplications. Finally, it costs $9|N|$ multiplications to run **MixToAdd**. For the basic subprotocols, it costs $18|N|$ multiplications by executing the *SEPB* protocol. In basic EPOC, it costs the user $\mathcal{O}(\kappa|N|)$ multiplications to encrypted all the coefficients C_j, costs $\mathcal{O}(\kappa\tau|N|)$ multiplications in Stage One, costs $\mathcal{O}(\tau(\kappa+|N|))$ multiplications in Stage Two, and costs the user $\mathcal{O}(\tau|N|)$ multiplications to decrypt all the final results. In enhanced EPOC, it costs the user $\mathcal{O}(\kappa|N|)$ multiplications to encrypted all the coefficients C_j and costs $\mathcal{O}(\kappa\gamma\phi|N|)$ multiplications to encrypt exponent of all monomials. In all, it costs $\mathcal{O}(\kappa\gamma\phi|N|)$ multiplications in Stage One, costs $\mathcal{O}(\kappa\tau(\gamma+|N|))$ multiplications in Stage Two, and costs the user $\mathcal{O}(\tau|N|)$ multiplications to decrypt all the final results. In full EPOC, it costs the user $\mathcal{O}(K|N|)$ multiplications to encrypted all the coefficients C_j and costs $\mathcal{O}(K\gamma\phi|N|)$ multiplications to encrypt exponent of all monomials. In all, it costs $\mathcal{O}(K\gamma\phi|N|)$ multiplications in Stage One, costs $\mathcal{O}(K\tau(\gamma+|N|))$ multiplications in Stage Two, and costs the user $\mathcal{O}(\tau|N|)$ multiplications to decrypt all the final results.

6.2.2 Communication Overhead

In this section, we compare the communication overhead about these three versions of EPOC. In SHED scheme, each ADD ciphertext $E_{pk^+}(x)$ needs $2|N|$ bits to represent while MUL ciphertext $E_{pk^\times}(x) = \{C_1, C_2\}$ needs $2|N|$ bits to transmit (C_1 and C_2 needs $|N|$ bits for storage, respectively). In the basic EPOC, the user needs to send all the encrypted coefficients $E_{pk^+}(C_j)$ to PCP which takes $\mathcal{O}(\kappa|N|)$ bits to communicate in Stage One. Moreover, all the partially decrypted results should be sent back to the user which takes $\mathcal{O}(\tau|N|)$ bits to transmit in Stage Two. So it costs $\mathcal{O}((\kappa+\tau)|N|)$ bits (one round between PCP and the user[6]) to transmit all the encrypted data in the transmission link in basic EPOC. In the enhanced EPOC, the user needs to send all the encrypted coefficients $E_{pk^\times_\alpha}(C_j)$ and the exponent of monomials $E_{pk^\times_\alpha}(t'_{j,k,l})$ to PCP which takes $\mathcal{O}(\kappa\gamma\phi|N|)$ bits to communicate. Also, it takes $\mathcal{O}(\kappa\gamma\tau\phi|N|)$ bits by executing Stage One of the enhanced EPOC ($3|N|$ bits for executing *SEPB* for one time). In Stage Two, all the partially decrypted results cost $\mathcal{O}(\tau|N|)$ bits to transmit. So it costs $\mathcal{O}(\kappa\gamma\tau\phi|N|)$ bits (one round between PCP and the user, $\mathcal{O}(\kappa\tau\gamma\phi)$ rounds between PCP and PCC) to transmit all the encrypted data in the transmission link. The full EPOC is similar to that of the enhance EPOC which costs $\mathcal{O}(K\gamma\phi|N|)$ bits between the user and PCP while costs $\mathcal{O}(K\gamma\tau\phi|N|)$ bits between PCP and PCC (one round between PCP and the user, $\mathcal{O}(K\tau\gamma\phi)$ rounds between PCP and PCC). Both the computational and communication comparison results are shown in Table 5.

7 Related Work

More and more data are generated and outsourced to cloud servers for storage with the development of cloud computing. However, the cloud server is a third-party server. One of the momentous problems in cloud computing is how to perform

6. For a resource limited user, one round communication between the user and the server is optimal.

computation over encrypted data. Homomorphic encryption is the holy grail of the cryptography which allows computations to be carried out on ciphertext which reflect on some operations over the plaintext. Additive homomorphic encryption scheme allows a third party to do some operations over the ciphertext which reflect on some additive calculations over the plaintext (e.g. Paillier cryptosystem [18], Benaloh cryptosystem [24]) Moreover, multiplicative homomorphic encryption permits users to do some operations over the ciphertext which reflect on some multiplication calculations over the plaintext (e.g. Unpadded RSA cryptosystem [25], ElGamal cryptosystem [19]). Many protocols [26], [27] and applications [28], [29] are proposed using these two kinds of homomorphic cryptosystems. However, it has been still a tricky problem to achieve both addition and multiplication in plaintext by calculation over the ciphertexts under one cryptosystem. A cryptosystem that supports arbitrary computations on ciphertexts is known as fully homomorphic encryption (FHE) and is far more powerful. The first fully homomorphic encryption scheme is constructed by Gentry [30] based on lattice-based cryptography which supports both addition and multiplication operations on ciphertexts, followed by many other efforts on fully homomorphic cryptosystems [31], [32]. However, fully homomorphic cryptography is still not efficient enough to be implemented on real systems [33], [34]. Recently, Lim et al. [12] show a switchable homomorphic encryption which can transform a ciphertext between additive homomorphic cryptosystem and multiplicative homomorphic cryptosystem. A limitation of the scheme is that it can only encrypt messages with Jacobi symbol 1 to be secure.

The Partial Homomorphic Cryptosystem (additive homomorphic OR multiplicative homomorphic) is much more efficient for secure outsourced storage [35], [36] and computation [37]. Samanthula et. al. [38] propose an efficient and Secure Data Sharing (SDS) framework using homomorphic encryption prevents the leakage of unauthorized data when a revoked user rejoins the system. Popa et. al. [39] design a system called CryptDB which aims to protect data confidentiality against threats by executing SQL (Structured Query Language) queries (such as, sum, equality, additions, etc.) over encrypted data in a privacy-preserving way. Stephen et. al. [40] present a system called Crypsis that allows execution of mapreduce-style data analysis jobs directly on encrypted big data. Tang et. al. [41] focus on authenticated data-outsourcing problem specifically for continuous updating the multi-version key-value data. Moreover, a lots of protocol are designed for some specific functionality over the outsourced encrypted data, such as: secure outsourced sequence comparison protocol [42], [43], secure outsourced k-means protocol [27], [44], secure outsourced set intersection protocol [45]. All the above the systems focus on protecting confidential of data rather than the computation procedure, i.e., all data are protected using encryption technique before outsourcing, the calculating algorithm/function are known to the cloud (curious cloud may know your are interested add/k-means calculation). Different from the above models, the new privacy issues are occurred when meet with the public dataset: protect the computation procedure and its calculated outputs with public inputs. A new system should be designed for protecting the outsourced computation procedure and calculated results against a curious servers.

8 Conclusion

In this paper, we propose a new and efficient framework for privacy-preserving outsourced functional computation over public data (EPOC). EPOC enables user to efficiently outsource the computation of a function to the cloud without compromising the privacy of the user's function and its output. As key components of EPOC, a new public key cryptosystem presented to support partially decryption, and a new Secure Exponent Calculation Protocol with Public Base is designed. In addition, through extensive performance evaluation, we demonstrate that our EPOC can balance the computation-intensive functional computation and user's privacy protection at different levels. As future work, we will study EPOC which provides verifiability of the outsourced computation over public databases.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under the research grant 14-C220-SMU-006. It is also supported in part by the National Natural Science Foundation of China under grant No. 61402109, No. 61370078 and No. 61502248.

References

[1] (2014). B. Chamberlin. Iot (internet of things) will go nowhere without cloud computing and big data analytics [Online]. Available: http://ibmcai.com/2014/11/20/iot-internet-of-things-will-go-nowhere-without-cloud-computing-and-big-data-analytics/

[2] H. Wang. (2011). Cloud computing in ecommerce [Online]. Available: http://www.comp.leeds.ac.uk/mscproj/reports/1011/wang.pdf

[3] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for IT and scientific research," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 10–13, 2009.

[4] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific cloud computing: Early definition and experience," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun.*, 25–27 Sep. 2008, pp. 825–830.

[5] (2015). Microsoft. Hybrid cloud [Online]. Available: http://www.microsoft.com/en-us/server-cloud/solutions/hybrid-cloud.aspx

[6] D. Gardner. (2014). Mit media lab computing director details the virtues of cloud for agility and disaster recovery [Online]. Available: https://www.linkedin.com/pulse/20141007202429-135530-mit-media-lab-computing-director-details-the-virtues-of-cloud-for-agility-and-disaster-recovery

[7] (2012). Cdc/national center for health statistics [Online]. Available: http://www.cdc.gov/nchs/data_access/ftp_data.htm

[8] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. (2010). A practical guide to support vector classification. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[9] J. Kamruzzaman, R. Sarker, I. Ahmad, et al., "SVM based models for predicting foreign currency exchange rates," in *Proc. 3rd IEEE Int. Conf. Data Mining*, 2003, pp. 557–560.

[10] J. H. Min and Y.-C. Lee, "Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters," *Expert Syst. Appl.*, vol. 28, no. 4, pp. 603–614, 2005.

[11] S. Hua and Z. Sun, "Support vector machine approach for protein subcellular localization prediction," *Bioinformatics*, vol. 17, no. 8, pp. 721–728, 2001.

[12] H. W. Lim, S. Tople, P. Saxena, and E. Chang. (2014). Faster secure arithmetic computation using switchable homomorphic encryption, *IACR Cryptol. ePrint Archive* [Online]. 2014, p. 539. Available: http://eprint.iacr.org/2014/539

[13] B. Qin and S. Liu, "Efficient chosen ciphertext secure public-key encryption under factoring assumption," *Security Commun. Netw.*, vol. 6, no. 3, pp. 351–360, 2013.
[14] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive bayesian classification," in *IEEE J. Biomedical Health Informat.*, 2015, to be published. Available: http://ieeexplore.ieee.org/xpls/abs all.jsp?arnumber=7047716&tag=1
[15] (2012). Proof of a result of euler and lagrange on mersenne divisors [Online]. Available: https://primes.utm.edu/notes/proofs/MerDiv2.html
[16] C. Ding, *Chinese Remainder Theorem*. Singapore: World Scientific, 1996.
[17] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. 17th Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[19] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*. New York, NY, USA: Springer, 1985, pp. 10–18.
[20] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptol. ePrint Archive*, vol. 2011, p. 272, 2011.
[21] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Nist special publication 800-57," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142.
[22] (2013). Istanbul stock exchange data set, uci machine learning repository [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ISTANBUL+STOCK+EXCHANGE
[23] D. E. Knuth, "Seminumerical asgorithm (arithmetic) the art of computer programming vol. 2," Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997.
[24] J. Benaloh, "Dense probabilistic encryption," in *Proc. Workshop Selected Areas Cryptography*, 1994, pp. 120–128.
[25] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[26] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure K-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 664–675.
[27] B. K. Samanthula, F. Rao, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user K-means clustering," *CoRR*, vol. abs/1412.4378, 2014.
[28] Y. Chen, C. Chu, J. Hwang, and J. Yoo, "A privacy-preserving human tracking scheme in centralized cloud based camera networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 10–14, 2014, pp. 793–798.
[29] R. Lu, H. Zhu, X. Liu, J. K. Liu, and S. Jun, "Towards efficient and privacy-preserving computing in big data era," *IEEE Netw. Mag.*, vol. 28, no. 4, pp. 46–50, 2014.
[30] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[31] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. 33rd Annu. Cryptol. Conf. Adv. Cryptol.*, 2013, pp. 75–92.
[32] J. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Proc. 17th Int. Conf. Practice Theory Public-Key Cryptograph*, 2014, pp. 311–328.
[33] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Proc. 13th Int. Conf. Practice Theory Public Key Cryptography*, 2010, pp. 420–443.
[34] L. Morris. (2013). Analysis of partially and fully homomorphic encryption [Online]. Available: http://www.liammorris.com/crypto2/Homomorphic%20Encryption%20Paper.pdf
[35] S. Kamara and K. E. Lauter, "Cryptographic cloud storage," in *Proc. 14th Int. Conf. Financial Cryptograpy Data Security*, 2010, pp. 136–149.
[36] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2263–2268, 2013.
[37] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, 2012.
[38] B. K. Samanthula, G. Howser, Y. Elmehdwi, and S. Madria, "An efficient and secure data sharing framework using homomorphic encryption in the cloud," in *Proc. 1st Int. Workshop Cloud Intell.*, 2012, p. 8.
[39] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Operat. Syst. Principles*, 2011, pp. 85–100.
[40] J. J. Stephen, S. Savvides, R. Seidel, and P. Eugster, "Practical confidentiality preserving big data analysis," in *Proc. 6th USENIX Workshop Hot Topics Cloud Comput.*, 2014, p. 10.
[41] Y. Tang, T. Wang, L. Liu, X. Hu, and J. Jang, "Lightweight authentication of freshness in outsourced key-value stores," in *Proc. 30th Annu. Comput. Security Appl. Conf.*, Dec. 8–12, 2014, pp. 176–185.
[42] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Sec.*, vol. 4, no. 4, pp. 277–287, 2005.
[43] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. M. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Proc. 17th Eur. Symp. Res. Comput. Security*, 2012, pp. 505–522.
[44] D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced K-means clustering," in *Proc. 9th ACM Symp. Inform., Comput. Commun. Security*, Jun. 03–06, 2014, pp. 123–134.
[45] F. Liu, W. K. Ng, W. Zhang, D. H. Giang, and S. Han, "Encrypted set intersection protocol for outsourced datasets," in *Proc. 2014 IEEE Int. Conf. Cloud Eng.*, 2014, pp. 135–140.

Ximeng Liu (S'13-M'16) received the BSc degree in electronic engineering from the Xidian University, Xi'an, China, in 2010, and the PhD degrees in cryptography from the Xidian University, China, in 2015. He was the research assistant at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore from 2013 to 2014. He is currently a research fellow at the School of Information System, Singapore Management University, Singapore. His research interests include cloud security, applied cryptography, and big data security. He is a member of the IEEE.

Baodong Qin received the doctor's degree in 2015 from the Shanghai Jiao Tong university, Shanghai, China. He is currently a lecturer at the Southwest University of Science and Technology, China. His research interests include applied cryptography and provable security.

Robert H. Deng has been a professor at the School of Information Systems, Singapore Management University since 2004. Prior to this, he was a principal scientist and a manager of Infocomm Security Department, Institute for Infocomm Research, Singapore. His research interests include data security and privacy, multimedia security, network, and system security. He was the associate editor of the *IEEE Transactions on Information Forensics and Security* from 2009 to 2012. He is currently an associate editor of the *IEEE Transactions on Dependable and Secure Computing*, an associate editor of Security and Communication Networks (John Wiley), and a member of Editorial Board of *Journal of Computer Science and Technology* (the Chinese Academy of Sciences). He is the cochair of the Steering Committee of the ACM Symposium on Information, Computer and Communications Security. He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006. He was named Community Service Star and Showcased Senior Information Security Professional by $(ISC)^2$ under its Asia-Pacific Information Security Leadership Achievements program in 2010. He received the Distinguished Paper Award of the 19th Annual Network and Distributed System Security Symposium (NDSS 2012) and the Best Paper Award of the 13th Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security (CMS 2012). He is a fellow of the IEEE.

Yingjiu Li is currently an associate professor in the School of Information Systems at the Singapore Management University (SMU). His research interests include RFID Security and Privacy, Mobile and System Security, Applied Cryptography and Cloud Security, and Data Application Security and Privacy. He has published over 130 technical papers in international conferences and journals, and served in the program committees for over 80 international conferences and workshops. The URL for his web page is http://www.mysmu.edu/faculty/yjli/. He is a senior member of the ACM and a member of the IEEE Computer Society.