This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Feng, Jun; Yang, Laurence T.; Zhu, Qing; Xiang, Yang; Chen, Jinjun; Yan, Zheng
Secure Outsourced Principal Eigentensor Computation for Cyber-Physical-Social Systems

Published in:
IEEE Transactions on Sustainable Computing

DOI:
[10.1109/TSUSC.2018.2881241](https://doi.org/10.1109/TSUSC.2018.2881241)

Published: 01/01/2021

Document Version
Peer reviewed version

Please cite the original version:
Feng, J., Yang, L. T., Zhu, Q., Xiang, Y., Chen, J., & Yan, Z. (2021). Secure Outsourced Principal Eigentensor Computation for Cyber-Physical-Social Systems. *IEEE Transactions on Sustainable Computing*, 6(1), 119 - 130. <https://doi.org/10.1109/TSUSC.2018.2881241>

This is the accepted version of the original article published by IEEE.

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Secure Outsourced Principal Eigentensor Computation for Cyber-Physical-Social Systems

Jun Feng, Laurence T. Yang, *Senior Member, IEEE*, Qing Zhu, Yang Xiang, *Senior Member, IEEE*, Jinjun Chen, *Senior Member, IEEE*, and Zheng Yan, *Senior Member, IEEE*

Abstract—Cyber-physical-social systems (CPSS) are revolutionizing the relationships between humans, computers and things. Outsourcing computation to cloud can offer resources-constrained enterprises and consumers sustainable computing in CPSS. However, ensuring the security of data in such an outsourced environment remains a research challenge. Principal eigentensor computation has emerged as a powerful tool dealing with multidimensional cyber-physical-social systems data. In this paper, we present two novel secure principal eigentensor computation (SPEC) schemes for sustainable CPSS. To the best of our knowledge, this is the first effort to address SPEC over encrypted data in cloud without the interaction need between multiple users and cloud. More specifically, we leverage cloud server and trusted hardware component to design a collaborative cloud model. Using the model, we propose (1) a basic SPEC scheme based on homomorphic computing and (2) an efficient SPEC scheme that combines the advantages of homomorphic computing and garbled circuits, and exploits packing technology to reduce computational cost. Finally, we theoretically and empirically analyze the security and efficiency of our SPEC schemes. Findings demonstrate that the proposed schemes provide a secure and efficient way of outsourcing computation for CPSS. In addition, from the cloud user's perspective, our proposal is lightweight.

Index Terms—Cloud computing, privacy, cyber-physical-social systems, tensor, principal eigentensor computation, sustainable computing.

1 INTRODUCTION

WITH the rapid proliferation of intelligent sensors and devices, cyber-physical systems (CPS), the integrated systems of cyber space and physical space, have attracted tremendous attention [1]. As the extension of the CPS, cyber-physical-social systems (CPSS) are the integrated systems of cyber space, physical space, and social space [2, 3]. CPSS involve symbiotic networks of human (social networks), intelligent sensors, switches, routers, mobile phone terminals, and information networks. CPSS have the potential to provide enormous novel approaches for sustainable computing. Due to the increasing popularity of CPSS, huge amounts of data generated in CPSS form cyber-physical-social big data,

which can be used to enhance the quality of sustainable computing.

Cloud computing is an emerging and promising computing tool in CPSS. Meeting the ever-growing expectations of users requires data processing and analytical capabilities beyond those of traditional computer infrastructures in CPSS. This, perhaps, explains the increasingly popularity of cloud computing due to the capability to support sustainable computing based on a pay-as-you-use business model [4]. Hence, we have witnessed a recent trend in processing cyber-physical-social big data in the cloud, despite the underlying cloud security issues, for example potential leakage of user's network data (particularly in the aftermath of the revelations by Edward Snowden), risks of malicious insider attacks, and inference threats [5, 6]. Thus, addressing such concerns is key to widespread cloud adoption in CPSS [7].

While cloud computing can be seen as an extension of the "traditional" computing, its design is significantly more complicated due to the interconnectivity and other requirements. Unsurprisingly, designing secure cloud computing has been the subject of recent focus [7], such as ensuring security of virtual machines on the cloud server [8], and ensuring data security and enabling secure computation on the cloud server [9, 10]. However, designing efficient secure computation remains one of the most cloud research challenges. This is the gap we seek to address in this paper.

Specifically, we explore the potential of computing principal eigentensor for tensor in cloud-enabled CPSS. Tensor model has been adopted in a good deal of cyber-physical-social big data analytics research [11]. In our previous work [12], for example, we presented a tensor-based model for big data representation and demonstrated the effectiveness

- J. Feng is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China. E-mail: junfeng989@gmail.com.
- L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada, and the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China. E-mail: ltyang@ieee.org.
- Q. Zhu is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: qingzhu2020@gmail.com.
- Y. Xiang is with Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia, E-mail: yxiang@swin.edu.au.
- J. Chen is with Swinburne Data Science Research Institute, Swinburne University of Technology, Australia, E-mail: jinjun.chen@gmail.com.
- Z. Yan is with the State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an 710071, China and with the Department of Communications and Networking, Aalto University, Espoo 02150, Finland, E-mail: zyan@xidian.edu.cn.

Manuscript received April 1, 2018; revised August 10, 2018.

of the approach. The principal eigentensor for tensor is a natural generalization of the principal eigenvector for matrix. The principal eigentensor indicates the ultimate steady state distribution of high-order data and has various novel applications. For example, the principal eigentensor can be used in network traffic prediction based on spatio-temporal context and latent social relationships. This allows us to generate better effectiveness than principal eigenvector. As principal eigentensor computation is time-consuming when meeting big data, it is a preferred option to carry out the computations in the cloud.

To the best of our knowledge, there is no prior work in the design of secure principal eigentensor computation which can achieve protecting users' privacy in cloud. We identify several challenges associated with the naive utilization of principal eigentensor computation. Firstly, to preserve data security, the principal eigentensor should be computed without the cloud having access to or inferring data information. Secondly, the secure principal eigentensor computation should be efficient and scalable to be practical for deployment. Thirdly, the computations (e.g. multilinear mode product, tensor-constant division, comparison of 2-norm of tensor difference and constant, and acquisition of principal eigentensor) are involved in principal eigentensor computation. However, it is not known how to design a secure protocols that will securely perform these complex computations in clouds. Finally, in practice, users generally have limited computation and communication resources (e.g. on their mobile devices), and principal eigentensor computation is time-consuming. Thus, it is not realistic to expect the users to remain online after supplying their data (e.g. loss of network connectivity) or for the data owners to carry out tasks while computing principal eigentensor. Therefore, secure principal eigentensor computation needs to be lightweight for users.

In this paper, we present two novel secure principal eigentensor computation (SPEC) schemes for protecting users' privacies in sustainable cyber-physical-social systems. Both schemes are designed to securely and efficiently compute the principal eigentensor over encrypted data in the cloud. We regard the main contributions of this paper to be as follows:

- We design a basic SPEC scheme which leverages homomorphic computing properties, and uses the cloud and the trusted hardware component to collaboratively and securely compute the principal eigentensor. To implement the scheme, we design a set of secure building blocks including secure multilinear mode product protocol, secure tensor addition protocol, secure tensor-constant division protocol, secure comparison of 2-norm of tensor difference and constant protocol on cloud, and secure acquisition of principal eigentensor for data users.
- We then present an efficient SPEC scheme, which combines the advantages of homomorphic computing and garbled circuits, and exploits packing technology to improve the efficiency of the basic SPEC scheme in a collaborative cloud model. The improved scheme builds on some protocols involving secure packing-based multilinear mode prod-

uct, secure packing-based tensor-constant division, improved secure comparison of 2-norm of tensor difference and constant, and secure packing-based acquisition of principal eigentensor. These protocols are very significant for tensor-based secure computation. To our best knowledge, this is the first work to achieve secure and efficient tensor computations over encrypted data in the cloud.

The SPEC schemes allow us to utilize the computational capabilities of the cloud, without compromising on data security. In our schemes, once the encrypted data have been outsourced to the cloud, data owners do not need to participate in any computations. In addition, the data users only need to undertake few addition computations upon receiving the un-encrypted data from the cloud and the trusted hardware component at the conclusion of the secure computing scheme. The schemes enable lightweight secure computing from the view of the data owners and the data users.

We also demonstrate the security of the SPPEC schemes under the semi-honest model. We also analyze both SPPEC schemes, in terms of accuracy, cost of users, and time in cloud server. We evaluate the utility of the SPPEC schemes using synthetic datasets. The results confirm that the SPEC schemes makes the outsourcing computation secure and efficient.

The remainder of this paper is organized as follows. In the next section, we introduce the preliminaries required in the understanding of this paper. The collaborative cloud model and the framework of secure principal eigentensor computation using the model are presented in Section 3. The basic SPEC scheme and the efficient SPEC scheme built on the collaborative cloud model are presented in Sections 4 and 5, respectively. Section 6 evaluates the security and performance of the SPPEC schemes. We review related works in Section 7, and conclude the paper in Section 8.

2 PRELIMINARIES

2.1 Homomorphic Computing

The Paillier cryptosystem [13] is an additive homomorphic and probabilistic asymmetric encryption scheme. For any given ciphertexts $\llbracket m_1 \rrbracket$, $\llbracket m_2 \rrbracket$ and constant c , the Paillier cryptosystem exhibits the following properties: Homomorphic addition computing $\llbracket m_1 + m_2 \rrbracket = \llbracket m_1 \rrbracket \times \llbracket m_2 \rrbracket \bmod n^2$; Homomorphic multiplication computing $\llbracket cm_1 \rrbracket = \llbracket m_1 \rrbracket^c \bmod n^2$.

2.2 Garbled Circuits

Yao's garbled circuit protocol [14] is one of the most effective protocols for secure cloud-based computation. The protocol is composed of two stages: circuit garbling and circuit evaluation.

In the circuit garbling stage, the circuit constructor CC chooses two random values K_i^0 and K_i^1 , corresponding to the bit values $a_i = 0$ and $a_i = 1$ respectively, for each wire i of the circuit. Next, for each gate g with wires i, j and t , CC calculates 4 values, $E_{(K_i^{a_i}, K_j^{a_j})}(K_t^{g(a_i, a_j)})$ for $a_i, a_j \in \{0, 1\}$, forming garbled table. After this, CC sends the

permuted garbled table and the random values of its input bit values to the circuit evaluator CE .

In the circuit evaluation stage, CE obtains the random values of its input bits making use of the oblivious transfer protocol. Afterwards, CE calculates the entire circuit gate-by-gate to get the random values of the output bits by decrypting the garbled table. Finally, the bits of output is able to be recovered from their corresponding random values.

2.3 Principal Eigentensor

Tensors and principal eigentensor have been applied in quantities of data analyses, involving social networks and big data technologies [11].

Let $\mathcal{A} \in \mathbb{R}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}$ and $\mathcal{X} \in \mathbb{R}^{U \times T \times L}$ be a 6-th order tensor and a 3-th order tensor, respectively. If the equation, $\mathcal{A} \times_{U_1 \times T_1 \times L_1}^{K \times T \times L} \mathcal{X} = \mathcal{X}$, is satisfied, then tensor $\mathcal{X}$ is known as the principal eigentensor.

The method of solving principal eigentensor is multilinear mode product power algorithm. The inputs for the algorithm are tensors $\mathcal{A} \in \mathbb{R}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}$ and $\mathcal{X}_0 \in \mathbb{R}^{U \times T \times L}$. Suppose x_{utl} is an element of the tensor $\mathcal{X}_0$, then x_{utl} should satisfy $x_{utl} \geq 0$ and the tensor $\mathcal{X}_0$ should satisfy $\|\mathcal{X}_0\|_1^F = 1$. The output is tensor $\mathcal{X}$. The basic idea is to make the tensor irreducible using primitivity adjustment method and to adopt the power method to obtain the principal eigentensor. The main steps of the multilinear mode product power algorithm is outlined as follows:

- 1) Let $n = 1$; 2) Compute $\mathcal{X}_n^{U \times T \times L} = \alpha \cdot \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2} \times_{U_1, T_1, L_1}^{U, T, L} \mathcal{X}_{n-1}^{U \times T \times L} + (1 - \alpha) \cdot E$;
- 3) If $\|\mathcal{X}_n - \mathcal{X}_{n-1}\|_2^F < w$, then $\mathcal{X} = \mathcal{X}_n$, break; else $n = n + 1$, goto 2).

3 PROBLEM STATEMENT

This section presents the collaborative cloud model that leverages cloud servers and trusted hardware components, and utilizes homomorphic computing and garbled circuits. The framework for the SPEC is also presented.

3.1 Collaborative Cloud Model

In the collaborative cloud model, the entities are multiple data owners, a cloud server, a trusted hardware component [15] and data users. The data owners encrypt their data and send these ciphertexts to one or more cloud servers. These data users will receive a result generated by computing a function on the data owners' data. The cloud servers and trusted hardware components collaboratively implement the secure computation of the function.

The process of securely computing function F over encrypted data $\llbracket d \rrbracket$ on the cloud server and whose output is the data x_1 , is referred to as secure function computation (SFC). We define the SFC process as follows: $SFC(\llbracket d \rrbracket) \rightarrow x_1$. At the conclusion of the SPEC process, the result x_1 is known only to the data users and nothing is revealed to the cloud servers and trusted hardware components.

Let FC denote the process of calculating the function F in the plain-domain without any privacy protection. Suppose data x is the output of the function F on data d in the plain-domain. The FC process is defined as follows: $FC(d) \rightarrow x$.

Moreover, we need to ensure that the result of the SFC process equals that of the FC process. Formally, the process SFC should satisfy the following equation: $x = x_1$.

In the collaborative cloud model, we are restricted from exposing data d or the intermediate results to any participants, and only the data users have access to the output result x . We also assume that all parties are in a semi-trusted model [16].

3.2 Framework of SPEC

To demonstrate how the collaborative cloud model can be used to implement secure computation, we use the principal eigentensor computation as an example. Fig. 1 outlines a framework for SPEC, and consists of the following three phases.

Fig. 1. Framework of SPEC Using the Collaborative Cloud for CPSS.

- 1) **Initialization:** The trusted hardware component uses the Paillier homomorphic cryptosystem to generate a public key pk and a private key sk for CPSS, and publishes pk to the other users. By using the Paillier encryption with pk , the data x from data owners is encrypted, $\llbracket x \rrbracket = E_{pk}(x)$, prior to outsourcing $\llbracket x \rrbracket$ to a cloud server. Encrypted tensor $\llbracket \mathcal{A} \rrbracket$ is formed by the encrypted data elements on the cloud server.
- 2) **SPEC Using the Collaborative Cloud Model:** The cloud server and the trusted hardware component collectively and securely calculate the principal eigentensor of the tensor $\mathcal{A}$ using multilinear mode product power method, homomorphic computing properties, and garbled circuits. The cloud server obtains the encrypted principal eigentensor $\llbracket \mathcal{X} \rrbracket$.
- 3) **Acquisition of Principal Eigentensor for the Data Users:** Finally, the data users securely obtain the principal eigentensor $\mathcal{X}$ with the assistance of the cloud server holding $\llbracket \mathcal{X} \rrbracket$ and the trusted hardware component holding sk .

4 A BASIC SPEC SCHEME

Using the collaborative cloud model, we now present a basic SPEC scheme based on homomorphic computing. We

first describe a transformational multilinear mode product power method for solving principal eigentensor. Then, for simplicity, we divide the SPEC scheme into five secure building blocks, namely: secure multilinear mode product protocol, secure tensor addition protocol, secure tensor-constant division protocol, secure comparison of 2-norm of tensor difference and constant protocol on cloud, and secure acquisition of principal eigentensor for data users. These building blocks together form the SPEC scheme.

4.1 A Transformational Method for Solving Principal Eigentensor

Since only integers can be encrypted by the Paillier cryptosystem, all variables in the principal eigentensor computation should be quantized to the nearest integer value. This may, however, result in error. Thus, to reduce the possibility of error, the variables are scaled using three positive integers ρ_1 , ρ_2 and ρ_3 (referred to as scaling factors). At the end of our scheme, the corresponding results (used to recover the true results) can be obtained.

Algorithm 1 Transformational Method for Solving Principal Eigentensor

Input: Tensors $\mathcal{A} \in \mathbb{R}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}$ and $\mathcal{X}_0 \in \mathbb{R}^{U \times T \times L}$.

Output: Principal eigentensor $\rho_3 \mathcal{X} \in \mathbb{R}^{U \times T \times L}$.

```

1: Let  $n = 1$ 
2: Compute  $\rho_1 \rho_2 \rho_3 \mathcal{X}_n^{U \times T \times L} = (\rho_1 \cdot \alpha) \cdot ((\rho_2 \cdot \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}) \times_{U_1, T_1, L_1}^{U, T, L} (\rho_3 \cdot \mathcal{X}_{n-1}^{U \times T \times L})) + \rho_2 \cdot ((\rho_1 \cdot (1 - \alpha)) \cdot (\rho_3 \cdot E))$ 
3:  $\rho_3 \mathcal{X}_n^{U \times T \times L} = (\rho_1 \rho_2 \rho_3 \mathcal{X}_n^{U \times T \times L}) / (\rho_1 \rho_2)$ 
4: if  $\|\rho_3 \mathcal{X}_n - \rho_3 \mathcal{X}_{n-1}\|_2^F < \rho_3 w$  then
5:    $\rho_3 \mathcal{X} = \rho_3 \mathcal{X}_n$ 
6:   break
7: else
8:    $n = n + 1$ 
9:   goto 2
10: end if
```

In order to improve accuracy, a transformational multilinear mode product power method for solving principal eigentensor is proposed (see Algorithm 1). Using Algorithm 1, we can obtain the result of the corresponding step in multilinear mode product power algorithm (see Section 2.3) without scaling. For example, let us look at the step $\rho_1 \rho_2 \rho_3 \mathcal{X}_n^{U \times T \times L} = (\rho_1 \cdot \alpha) \cdot ((\rho_2 \cdot \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}) \times_{U_1, T_1, L_1}^{U, T, L} (\rho_3 \cdot \mathcal{X}_{n-1}^{U \times T \times L})) + \rho_2 \cdot ((\rho_1 \cdot (1 - \alpha)) \cdot (\rho_3 \cdot E))$ in Algorithm 1, and the step $\mathcal{X}_n^{U \times T \times L} = \alpha \cdot \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2} \times_{U_1, T_1, L_1}^{U, T, L} \mathcal{X}_{n-1}^{U \times T \times L} + (1 - \alpha) \cdot E$ in multilinear mode product power algorithm. From the result $\rho_1 \rho_2 \rho_3 \mathcal{X}_n^{U \times T \times L}$ of Algorithm 1, we arrive at the result $\mathcal{X}_n^{U \times T \times L}$ for multilinear mode product power algorithm (i.e., $\mathcal{X}_n^{U \times T \times L} = (\rho_1 \rho_2 \rho_3 \mathcal{X}_n^{U \times T \times L}) / (\rho_1 \rho_2 \rho_3)$). To avoid overflows after many iterations, step 3 is added to Algorithm 1.

From Algorithm 1, we need to resolve the following issues, namely: multilinear mode product, tensor addition, tensor-constant division, comparison of 2-norm of tensor difference and constant, and acquisition of principal eigentensor over encrypted data.

4.2 Secure Multilinear Mode Product and Secure Tensor Addition

The cloud server with input two encrypted tensors $[\mathcal{A}] \in \mathbb{R}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}$ and $[\mathcal{B}] \in \mathbb{R}^{U \times T \times L}$ and the trusted hardware component with private key sk will collaboratively and securely compute the Encrypted tensor $[\mathcal{C}]$, where $\mathcal{C}^{U_2 \times T_2 \times L_2} = \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2} \times_{U_1, T_1, L_1}^{U, T, L} \mathcal{B}^{U \times T \times L}$. $\mathcal{A}$ and $\mathcal{B}$ are not known to either the cloud server or the trusted hardware component. The output $[\mathcal{C}]$ is known only to the cloud server. During this protocol, no information about $\mathcal{A}$ and $\mathcal{B}$ will be revealed to the cloud server and the trusted hardware component. The operations in multilinear mode product are multiplication and addition. Therefore, the secure multilinear mode product protocol (SMMP) using the collaborative cloud model can be implemented by exploiting secure multiplication protocol (SM) [17] and the Paillier addition property. Formally, the protocol is defined as follows: $SMMP([\mathcal{A}], [\mathcal{B}]) \rightarrow [[\mathcal{A} \times_{U_1, T_1, L_1}^{U, T, L} \mathcal{B}]]$.

The secure tensor addition protocol using the collaborative cloud model considers the cloud server with two encrypted tensors $[\mathcal{A}], [\mathcal{B}] \in \mathbb{R}^{U \times T \times L}$ securely computes the encrypted tensor $[A + B]$ directly adopting the Paillier addition property. The entry of $[A + B]$ is $[a_{utl}] \cdot [b_{utl}]$. The protocol is defined as follows: $STA([\mathcal{A}], [\mathcal{B}]) \rightarrow [A + B]$.

4.3 Secure Tensor-Constant Division

The secure tensor-constant division protocol using the collaborative cloud model (STCD) considers the cloud server with input encrypted tensor $[A]$ and constant c , and the trusted hardware component with the private key sk and c . Here, the tensor A is neither known to the cloud server nor the trusted hardware component. The goal of the STCD is to collaboratively compute the encrypted tensor $[A/c]$ known only to the cloud server. During STCD, no information about the tensor A is exposed to the cloud server and the trusted hardware component. The basic idea is based on Observation 1. The proposed protocol is described in Algorithm 2. Formally, the protocol is defined as follows: $STCD([\mathcal{A}], c) \rightarrow [A/c]$.

Observation 1: For any given integers a and c , the following property holds: $a/c = (a + rc)/c - r$, where r is a random number.

For every entry a_{utl} ($1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$) in the tensor A , the cloud server randomly picks an integer r , and randomizes a_{utl} by computing $[a'_{utl}] = [a_{utl}] \cdot [rc]$, and then sends the encrypted value $[a'_{utl}]$ to the trusted hardware component. Upon receiving the ciphertext $[a'_{utl}]$ from the cloud server, the trusted hardware component decrypts it and computes $b'_{utl} = a'_{utl}/c$ to obtain $a_{utl}/c + r$. Note that the trusted hardware component does not know the random number r , so it cannot obtain the result a_{utl}/c . Afterwards, the trusted hardware component encrypts b'_{utl} and sends the ciphertext $[b'_{utl}]$ to the cloud server. Finally, applying the Paillier properties, the cloud server removes the random value r from $[b'_{utl}]$ to obtain the encrypted value $[a_{utl}/c]$ (i.e. $[b_{utl}] = [a_{utl}/c + r] [-r]$). Without the private key sk , the cloud server does not see the plaintext result a/c .

Algorithm 2 Secure Tensor-Constant Division (STCD)

Input: Cloud Server: Encrypted tensor $\llbracket A \rrbracket \in \mathbb{R}^{U \times T \times L}$ and constant c . Trusted Hardware Component: Private key sk and constant c .

Output: Encrypted tensor $\llbracket B \rrbracket$ only to cloud server, where $B = A/c$.

```

1: for  $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$  do
2:   Cloud Server:
3:     choose a random integer  $r$ 
4:      $\llbracket rc \rrbracket \leftarrow E_{pk}(r \cdot c)$ 
5:      $\llbracket a'_{utl} \rrbracket \leftarrow \llbracket a_{utl} \rrbracket \llbracket rc \rrbracket$ 
6:     send  $\llbracket a'_{utl} \rrbracket$  to trusted hardware component
7:   Trusted Hardware Component:
8:     receive  $\llbracket a'_{utl} \rrbracket$ 
9:      $a'_{utl} \leftarrow D_{sk}(\llbracket a'_{utl} \rrbracket)$ 
10:     $b'_{utl} \leftarrow a'_{utl}/c$ 
11:     $\llbracket b'_{utl} \rrbracket \leftarrow E_{pk}(b'_{utl})$ 
12:    send  $\llbracket b'_{utl} \rrbracket$  to Cloud Server
13:   Cloud Server:
14:     receive  $\llbracket b'_{utl} \rrbracket$ 
15:      $\llbracket -r \rrbracket \leftarrow E_{pk}(-r)$ 
16:      $\llbracket b_{utl} \rrbracket \leftarrow \llbracket b'_{utl} \rrbracket \llbracket -r \rrbracket$ 
17: end for

```

4.4 Secure Comparison of 2-Norm of Tensor Difference and Constant

The secure comparison of 2-norm of tensor difference and constant protocol using the collaborative cloud model (SCNC) assumes that the cloud server holds two encrypted tensors $\llbracket A \rrbracket, \llbracket B \rrbracket \in \mathbb{R}^{U \times T \times L}$, and constant c , and the trusted hardware component holds private key sk and c . The goal is to collaboratively and securely compare 2-Norm of tensor difference with constant. If $\|A - B\|_2^F < c$, then the protocol outputs 1; otherwise, 0. Algorithm 3 shows the proposed method and the protocol is written as follows: $SCNC(\llbracket A \rrbracket, \llbracket B \rrbracket, c) \rightarrow (\|A - B\|_2^F < c)$.

Algorithm 3 Secure Comparison of 2-Norm of Tensor Difference and Constant (SCNC)

Input: Cloud Server: Two encrypted tensors $\llbracket A \rrbracket, \llbracket B \rrbracket \in \mathbb{R}^{U \times T \times L}$, and constant c . Trusted hardware component: Private key sk and constant c .

Output: Comparison result r , where $r = (\|A - B\|_2^F < c)$.

```

1: for  $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$  do
2:   Cloud Server:  $\llbracket d_{utl} \rrbracket \leftarrow \llbracket a_{utl} \rrbracket \cdot \llbracket b_{utl} \rrbracket^{n-1}$ 
3:   Cloud Server and Trusted Hardware Component:
4:      $\llbracket d_{utl}^2 \rrbracket \leftarrow SM(\llbracket d_{utl} \rrbracket, \llbracket d_{utl} \rrbracket)$ 
5: end for
6: Cloud Server:  $\llbracket (\|D\|_2^F)^2 \rrbracket \leftarrow \prod_{utl} \llbracket d_{utl}^2 \rrbracket$ 
7: Cloud Server and Trusted Hardware Component:
8:    $r \leftarrow SC(\llbracket (\|D\|_2^F)^2 \rrbracket, c)$ 

```

The encrypted difference of the tensors A and B can be calculated using the Paillier properties in the cloud server in step 2. Let the difference tensor be D . Next, the problem becomes how to compare the tensor 2-norm $\|D\|_2^F$ with the constant c . The 2-norm computing $\|D\|_2^F =$

$\sqrt{\sum_{u=1}^U \sum_{t=1}^T \sum_{l=1}^L (D_{utl})^2}$ includes square root operation. However, the square root operation over encrypted data is more complex and time-consuming. Fortunately, the comparison result of $\|D\|_2^F$ and c equals that of $(\|D\|_2^F)^2$ and c^2 . Therefore, we now need to compare $(\|D\|_2^F)^2$ with c^2 over encrypted data, which removes the square root operation. The cloud server and the trusted hardware component collaboratively compute $\llbracket (\|D\|_2^F)^2 \rrbracket$ using the SM protocol and the Paillier addition property in steps 4 and 6. Then, the cloud server and the trusted hardware component jointly use secure comparison protocol (SC) [18] based on homomorphic encryption to compare the squared tensor 2-norm $(\|D\|_2^F)^2$ with c^2 on the ciphertext $\llbracket (\|D\|_2^F)^2 \rrbracket$ and the constant c^2 in step 8.

4.5 Secure Acquisition of Principal Eigentensor for Data Users

After the cloud server obtains the encrypted principal eigentensor $\llbracket X \rrbracket$, we need to develop a secure and efficient protocol which enables data users to obtain the principal eigentensor X . The protocol does not allow the cloud server to learn anything about the tensor X . The secure acquisition of principal eigentensor protocol using the collaborative cloud model (SA) is given in Algorithm 4. Formally, the protocol is defined as follows: $SA(\llbracket X \rrbracket) \rightarrow X$.

Algorithm 4 Secure Acquisition of Principal Eigentensor for Data Users (SA)

Input: Cloud Server: Encrypted tensor $\llbracket X \rrbracket \in \mathbb{R}^{U \times T \times L}$. Trusted Hardware Component: Private key sk .

Output: Tensor X only to data user.

```

1: for  $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$  do
2:   Cloud Server:
3:     choose a random integer  $r_{utl}$ 
4:      $\llbracket r_{utl} \rrbracket \leftarrow E_{pk}(r_{utl})$ 
5:      $\llbracket x'_{utl} \rrbracket \leftarrow \llbracket x_{utl} \rrbracket \llbracket r_{utl} \rrbracket$ 
6:     send  $\llbracket x'_{utl} \rrbracket$  to trusted hardware component
7:     send  $r'_{utl}$  to data user
8:   Trusted Hardware Component:
9:     receive  $\llbracket x'_{utl} \rrbracket$ 
10:     $x'_{utl} \leftarrow D_{sk}(\llbracket x'_{utl} \rrbracket)$ 
11:    send  $x'_{utl}$  to data user
12:   Data User:
13:     receive  $x'_{utl}$ 
14:     receive  $r_{utl}$ 
15:     $x_{utl} \leftarrow x'_{utl} - r_{utl}$ 
16: end for

```

The basic idea of the SA protocol is that for any given integer x , the equation $x = x + r - r$, where r is a random number, holds. For the entry $\llbracket x_{utl} \rrbracket$ ($1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$) in the tensor $\llbracket X \rrbracket$, the protocol proceeds as follows:

The cloud server first picks a random integer r_{utl} and computes $\llbracket x'_{utl} \rrbracket = \llbracket x_{utl} \rrbracket \llbracket r_{utl} \rrbracket$ to distort the value x_{utl} , and then sends $\llbracket x'_{utl} \rrbracket$ and r_{utl} to the trusted hardware component and the data users, respectively in steps 2 to 7.

Upon receiving $\llbracket x'_{utl} \rrbracket$ from the cloud server, the trusted hardware component decrypts the received value to obtain the distorted value $x'_{utl} = x_{utl} + r_{utl}$ prior to sending x'_{utl} to the data users in steps 8 to 11. Since the trusted hardware component does not hold r_{utl} , it cannot see x_{utl} .

Upon receiving r_{utl} from the cloud server and x'_{utl} from the trusted hardware component, the data users compute $x_{utl} = x'_{utl} - r_{utl}$ to obtain the entry x_{utl} in steps 12 to 15.

5 AN EFFICIENT SPEC SCHEME

This section uses the collaborative cloud model to propose an efficient SPEC scheme based on homomorphic computing and garbled circuit. In other words, the scheme takes advantage of homomorphic computing and garbled circuit, as well as employing packing technology to improve its efficiency. This efficient SPEC scheme improves the SMMP protocol, the STCD protocol, the SCNC protocol and the SA protocol of the basic SPEC scheme.

5.1 Secure Packing-Based Multilinear Mode Product

The SMMP protocol requires a number of (time-consuming) operations in the encrypted-domain. To improve the SMMP protocol's efficiency, we propose a packing-based SMMP protocol using the collaborative cloud model described in Algorithm 5.

In order to explain the packing-based SMMP protocol, we will now outline the packing technology. Assume Alice wishes to send K integers $a_1, a_2, \dots, a_K$, whose digits are less than or equal to w , to Bob. Alice first adopts the following formula to pack the integers:

$$a = \text{Pack}(a_K | \dots | a_2 | a_1) = \sum_{k=1}^K \left(10^{w(k-1)} \cdot a_k \right). \quad (1)$$

Then, Alice encrypts the packed value $\llbracket a \rrbracket = E_{pk}(a)$, and sends the encrypted value $\llbracket a \rrbracket$ to Bob. Bob decrypts $\llbracket a \rrbracket$ and obtains the packed value $a = D_{sk}(\llbracket a \rrbracket)$. From the packed value a , Bob can easily obtain the $a_i (1 \leq i \leq K)$. The data unpacking is defined as:

$$a_K | \dots | a_2 | a_1 = \text{Unpack}(a). \quad (2)$$

In the remaining protocols, the number of values packed into one value is K , and for simplicity, we suppose $K = U \cdot T \cdot L$. To securely calculate the encrypted value $\llbracket a_{utlu_2t_2l_2} b_{utl} \rrbracket (1 \leq u_2 \leq U_2, 1 \leq t_2 \leq T_2, 1 \leq l_2 \leq L_2)$, the protocol proceeds as follows:

The cloud server blinds $a_{utlu_2t_2l_2}$ and b_{utl} for $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$. The cloud server picks $2K$ integers $r_1, r'_1, \dots, r_K, r'_K$ randomly, and packs them into two integers r and r' using Equation 1 in steps 3 and 4. The cloud server encrypts the integers a' and b' in step 5. Then, the cloud server obtains the encrypted packed disturbed values $\llbracket a' \rrbracket = \llbracket a_{UTLu_2t_2l_2} + r_K | \dots | a_{112u_2t_2l_2} + r_2 | a_{111u_2t_2l_2} + r_1 \rrbracket$ and $\llbracket b' \rrbracket = \llbracket b_{UTL} + r'_K | \dots | b_{112} + r'_2 | b_{111} + r'_1 \rrbracket$, and sends them to the trusted hardware component in steps 6 to 10.

The trusted hardware component will decrypt the received encrypted packed values $\llbracket a' \rrbracket$ and $\llbracket b' \rrbracket$, prior to unpacking the packed values using Equation 2, and obtain blinded values $a'_{utlu_2t_2l_2} (a'_{utlu_2t_2l_2} = a_{utlu_2t_2l_2} + r_{utlu_2t_2l_2})$

Algorithm 5 Secure Packing Based Multilinear Mode Product

Input: Cloud Server: Two encrypted tensors $\llbracket A \rrbracket \in \mathbb{R}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2}$ and $\llbracket B \rrbracket \in \mathbb{R}^{U \times T \times L}$. Trusted Hardware Component: Private key sk .

Output: Encrypted tensor $\llbracket C \rrbracket$ only to Cloud Server, where $\mathcal{C}^{U_2 \times T_2 \times L_2} = \mathcal{A}^{U_1 \times T_1 \times L_1 \times U_2 \times T_2 \times L_2} \times_{U_1, T_1, L_1}^{U, T, L} \mathcal{B}^{U \times T \times L}$.

- 1: **for** $1 \leq u_2 \leq U_2, 1 \leq t_2 \leq T_2, 1 \leq l_2 \leq L_2$ **do**
- 2: Cloud Server:
- 3: choose $2K$ random integers $r_1, r'_1, \dots, r_K, r'_K$
- 4: $a' \leftarrow \text{Pack}(r_K | \dots | r_2 | r_1), b' \leftarrow \text{Pack}(r'_K | \dots | r'_2 | r'_1)$
- 5: $\llbracket a' \rrbracket \leftarrow E_{pk}(a'), \llbracket b' \rrbracket \leftarrow E_{pk}(b')$
- 6: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 7: $k \leftarrow TL(u-1) + L(t-1) + l$
- 8: $\llbracket a' \rrbracket \leftarrow \llbracket a' \rrbracket \cdot \llbracket a_{utlu_2t_2l_2} \rrbracket^{10^{w(k-1)}}$
- 9: $\llbracket b' \rrbracket \leftarrow \llbracket b' \rrbracket \cdot \llbracket b_{utl} \rrbracket^{10^{w(k-1)}}$
- 10: send $\llbracket a' \rrbracket, \llbracket b' \rrbracket$ to trusted hardware component
- 11: **end for**
- 12: Trusted Hardware Component:
- 13: receive $\llbracket a' \rrbracket, \llbracket b' \rrbracket$
- 14: $a' \leftarrow D_{sk}(\llbracket a' \rrbracket), b' \leftarrow D_{sk}(\llbracket b' \rrbracket)$
- 15: $a'_K | \dots | a'_2 | a'_1 \leftarrow \text{Unpack}(a')$
- 16: $b'_K | \dots | b'_2 | b'_1 \leftarrow \text{Unpack}(b')$
- 17: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 18: $\llbracket a'_{utlu_2t_2l_2} b'_{utl} \rrbracket \leftarrow E_{pk}(a'_{utlu_2t_2l_2} \cdot b'_{utl})$
- 19: send $\llbracket a'_{utlu_2t_2l_2} b'_{utl} \rrbracket$ to Cloud Server
- 20: **end for**
- 21: Cloud Server:
- 22: receive UTL encrypted values $\llbracket a'_{utlu_2t_2l_2} b'_{utl} \rrbracket$
- 23: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 24: $k \leftarrow TL(u-1) + L(t-1) + l$
- 25: $\llbracket a_{utlu_2t_2l_2} b_{utl} \rrbracket \leftarrow \llbracket a'_{utlu_2t_2l_2} b'_{utl} \rrbracket \cdot \llbracket a_{utlu_2t_2l_2} \rrbracket^{N-r'_k} \cdot \llbracket b_{utl} \rrbracket^{N-r_k} \cdot \llbracket r_k \cdot r'_k \rrbracket^{N-1}$
- 26: **end for**
- 27: $\llbracket c_{u_2t_2l_2} \rrbracket \leftarrow \prod_{utl} \llbracket a_{utlu_2t_2l_2} b_{utl} \rrbracket$
- 28: **end for**

and $b'_{utl} (b'_{utl} = b_{utl} + r_{utl}) (1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L)$ in steps 13 to 16. The trusted hardware component computes $\llbracket a'_{utlu_2t_2l_2} b'_{utl} \rrbracket = E_{pk}(a'_{utlu_2t_2l_2} \cdot b'_{utl})$ and sends it to the cloud server for $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ in steps 17 to 20.

The cloud server then applies the property $a_{utlu_2t_2l_2} b_{utl} = (a_{utlu_2t_2l_2} + r_{utlu_2t_2l_2})(b_{utl} + r_{utl}) - a_{utlu_2t_2l_2} r_{utl} - b_{utl} r_{utlu_2t_2l_2} - r_{utlu_2t_2l_2} r_{utl}$ and the Paillier properties to obtain the ciphertext $\llbracket a_{utlu_2t_2l_2} b_{utl} \rrbracket$, for $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ in steps 22 to 26.

5.2 Secure Packing-Based Tensor-Constant Division

In the STCD protocol, UTL encryptions and decryptions are required. To increase efficiency, a packing-based STCD protocol using the collaborative cloud model is described in Algorithm 6. The protocol uses the packing technology to significantly reduce the time due to encryptions and decryptions.

In the packing-based STCD protocol, the cloud server initially chooses K random integers $r_1, r_2, \dots, r_K$, packs their c -multiple values $c \cdot r_1, c \cdot r_2, \dots, c \cdot r_K$ applying

Algorithm 6 Secure Packing Based Tensor-Constant Division

Input: Cloud Server: Encrypted tensor $\llbracket A \rrbracket \in \mathbb{R}^{U \times T \times L}$ and constant c . Trusted Hardware Component: Private key sk and constant c .

Output: Encrypted tensor $\llbracket B \rrbracket$ only to Cloud Server, where $B = A/c$.

- 1: Cloud Server:
- 2: choose K random integers $r_1, r_2, \dots, r_K$
- 3: $a' = \text{Pack}(c \cdot r_K | \dots | c \cdot r_2 | c \cdot r_1), \llbracket a' \rrbracket \leftarrow E_{pk}(a')$
- 4: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 5: $k \leftarrow TL(u-1) + L(t-1) + l$
- 6: $\llbracket a' \rrbracket \leftarrow \llbracket a' \rrbracket \cdot \llbracket a_{utl} \rrbracket^{10^{w(k-1)}}$
- 7: send $\llbracket a' \rrbracket$ to trusted hardware component
- 8: **end for**
- 9: Trusted Hardware Component:
- 10: receive $\llbracket a' \rrbracket$
- 11: $a' \leftarrow D_{sk}(\llbracket a' \rrbracket), a'_K | \dots | a'_2 | a'_1 \leftarrow \text{UnPack}(a')$
- 12: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 13: $b'_{utl} \leftarrow a'_{utl}/c$
- 14: send $\llbracket b'_{utl} \rrbracket$ to Cloud Server
- 15: **end for**
- 16: Cloud Server:
- 17: receive UTL encrypted values $\llbracket b'_{utl} \rrbracket$
- 18: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 19: $\llbracket b_{utl} \rrbracket \leftarrow \llbracket b'_{utl} \rrbracket \llbracket -r_{utl} \rrbracket$
- 20: **end for**

Equation 1, and encrypts the packed value in steps 2 and 3. Then, the packed value is used to perturb the value a_{utl} , ($1 \leq t \leq T, 1 \leq l \leq L$) in the encrypted-domain. The cloud server obtains the encrypted packed disturbed value $\llbracket a' \rrbracket = \llbracket a_{UTL} + cr_K | \dots | a_{112} + cr_2 | a_{111} + cr_1 \rrbracket$ and sends it to the trusted hardware component in steps 4 to 8.

The trusted hardware component then receives and decrypts $\llbracket a' \rrbracket$ using private key sk to obtain the packed disturbed value. This is subsequently unpacked to obtain the distorted values $a_{UTL} + cr_K, \dots, a_{112} + cr_2, a_{111} + cr_1$ using Equation 2 in steps 10 to 11. The remaining steps of Algorithm 6 are similar to steps 10 to 16 of Algorithm 2.

5.3 Improved Secure Comparison of 2-Norm of Tensor Difference and Constant

Similarly, for the improved efficiency of the SCNC protocol, we use the packing technology to reduce both encryption and decryption times. Furthermore, we improve the SCNC protocol by replacing the underlying homomorphic encryption comparison method with the garbled circuit comparison method. The improved SCNC protocol using the collaborative cloud model is described in Algorithm 7.

To leverage the packing technology in the protocol, the cloud server first blinds d_{utl} ($1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$) with K random integers to obtain the encrypted packed distorted value $\llbracket d' \rrbracket = \llbracket d_{UTL} + r_K | \dots | d_{112} + r_2 | d_{111} + r_1 \rrbracket$ in steps 2-8. Upon receiving the encrypted value $\llbracket d' \rrbracket$, the trusted hardware component decrypts the ciphertext and then unpacks the result to obtain the distorted values $d_{111} + r_1, d_{112} + r_2, \dots, d_{UTL} + r_K$ in steps 11 and 12. The trusted hardware component encrypts the value $(d_{utl} + r_{utl})^2$ and

Algorithm 7 Improved Secure Comparison of 2-Norm of Tensor Difference and Constant

Input: Cloud Server: Two encrypted tensors $\llbracket A \rrbracket \in \mathbb{R}^{U \times T \times L}$ and $\llbracket B \rrbracket \in \mathbb{R}^{U \times T \times L}$, and constant c . Trusted Hardware Component: Private key sk and constant c .

Output: Comparison result r , where $r = (\|A - B\|_2^F < c)$.

- 1: Cloud Server:
- 2: choose K random integers $r_1, r_2, \dots, r_K$
- 3: $d' = \text{Pack}(r_K | \dots | r_2 | r_1), \llbracket d' \rrbracket \leftarrow E_{pk}(d')$
- 4: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 5: $k \leftarrow TL(u-1) + L(t-1) + l$
- 6: $\llbracket d_{utl} \rrbracket \leftarrow \llbracket a_{utl} \rrbracket \cdot \llbracket b_{utl} \rrbracket^{n-1}$
- 7: $\llbracket d' \rrbracket \leftarrow \llbracket d' \rrbracket \cdot \llbracket d_{utl} \rrbracket^{10^{w(k-1)}}$
- 8: **end for**
- 9: send $\llbracket d' \rrbracket$ to Trusted Hardware Component
- 10: Trusted Hardware Component:
- 11: receive $\llbracket d' \rrbracket$
- 12: $d' \leftarrow D_{sk}(\llbracket d' \rrbracket), d'_K | \dots | d'_2 | d'_1 \leftarrow \text{Unpack}(x)$
- 13: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 14: $\llbracket (d'_{utl})^2 \rrbracket \leftarrow E_{pk}(d'_{utl} \cdot d'_{utl})$
- 15: send $\llbracket (d'_{utl})^2 \rrbracket$ to Cloud Server
- 16: **end for**
- 17: Cloud Server:
- 18: receive UTL encrypted values $\llbracket (d'_{utl})^2 \rrbracket$
- 19: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 20: $k \leftarrow TL(u-1) + L(t-1) + l$
- 21: $\llbracket d_{utl}^2 \rrbracket \leftarrow \llbracket (d'_{utl})^2 \rrbracket \cdot (\llbracket d_{utl} \rrbracket^{N-r_k})^2 \cdot \llbracket (r_k)^2 \rrbracket^{N-1}$
- 22: **end for**
- 23: $\llbracket \text{Norm}^2 \rrbracket \leftarrow \prod_{utl} \llbracket d_{utl}^2 \rrbracket$
- 24: Cloud Server and Trusted Hardware Component:
- 25: $r \leftarrow \text{SCGC}(\llbracket \text{Norm}^2 \rrbracket, c)$

sends the result $\llbracket (d'_{utl})^2 \rrbracket$ to the cloud server, for $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ in steps 13 to 16. After this, using the property $(d_{utl})^2 = (d_{utl} + r_{utl})^2 - 2d_{utl}r_{utl} - (r_{utl})^2$ and the Paillier properties, the cloud server computes $\llbracket (d_{utl})^2 \rrbracket$ for $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ in steps 19 to 21.

Fig. 2. Garbled circuits based comparison method.

After step 23, the cloud server holding the ciphertext $\llbracket (\|D\|_2^F)^2 \rrbracket$ and the constant c securely compares $\|D\|_2^F$ and c with the assistance of the trusted hardware component

holding the private key sk . To improve efficiency, step 25 employs the secure comparison method based on garbled circuits (SCGC)–see Fig. 2. For simplicity, we assume $\llbracket a \rrbracket = \left[\left(\left(\|D\|_2^F \right)^2 \right) \right]$ and $b = c^2$. The SCGC method needs one addition garbled circuit and one comparison garbled circuit.

In the preparation stage, the trusted hardware component generates one garbled circuit. The circuit consists of one addition garbled circuit and one comparison garbled circuit, where the output of addition circuit serves as the input of the comparison circuit. The trusted hardware component sends the garbled table, and the random keys (k_0 and k_1) of the output wire to the cloud server.

First, the cloud server picks a random integer r and encrypts it. The cloud server calculates $\llbracket a + r \rrbracket = \llbracket a \rrbracket \llbracket r \rrbracket$ to generate the ciphertext $\llbracket a + r \rrbracket$ of the blinded value of a , and sends the ciphertext to the trusted hardware component.

The trusted hardware component receives and decrypts $\llbracket a + r \rrbracket$ to obtain the blinded value $a + r$.

Next, the trusted hardware component sends the random keys corresponding to $a + r$ to the cloud server. The cloud server obviously acquires the random keys related to r . Then, the cloud server decrypts the garbled table of the addition circuit to obtain the random key corresponding to a using these random keys.

After this, the trusted hardware component sends the random keys related to b to the cloud server. The cloud server decrypts the garbled table of the comparison circuit using these random keys corresponding to a and b to obtain the random key k related to the comparison result.

Finally, the cloud server outputs the comparison result depending on whether k equals k_0 or k_1 .

5.4 Secure Packing-Based Acquisition of Principal Eigentensor for Data Users

UTL encryptions and decryptions are required in the process of the SA protocol. In a similar fashion, the packing technology is employed to improve efficiency of the SA protocol. The proposed packing-based SA protocol using the collaborative cloud model is described in Algorithm 8.

K random numbers in $\mathbb{Z}_n$ are selected and packed into a value x' using Equation 1 in the cloud server, and x' is encrypted in steps 2 and 3. Then, the ciphertext $\llbracket x' \rrbracket = \llbracket x_{UTL} + r_K | \cdots | x_{112} + r_2 | x_{111} + r_1 \rrbracket$ is generated and sent to the trusted hardware component in steps 4 to 8.

Upon receiving from the cloud server, the ciphertext $\llbracket x' \rrbracket$ is decrypted. And the trusted hardware component unpacks the result to obtain $x_{111} + r_1, x_{112} + r_2, x_{UTL} + r_K$ in steps 10 and 11. The rest is the same as Algorithm 4.

6 SECURITY AND PERFORMANCE EVALUATIONS

In this section, we evaluate the security and performance of the SPEC schemes. As this is the first effort to develop secure principal eigentensor computing schemes in cloud, we are not able to benchmark our schemes against another similar scheme. Hence, the performance of our SPEC schemes is evaluated under different parameter settings.

Algorithm 8 Secure Packing Based Acquisition of Principal Eigentensor for Data User

Input: Cloud Server: Encrypted tensor $\llbracket \mathcal{X} \rrbracket \in \mathbb{R}^{U \times T \times L}$.
 Trusted Hardware Component: Private key sk .
Output: Tensor $\mathcal{X}$ only to data user.

- 1: Cloud Server:
- 2: choose K random integers $r_1, r_2, \dots, r_K$
- 3: $x' = \text{Pack}(r_K | \cdots | r_2 | r_1), \llbracket x' \rrbracket \leftarrow E_{pk}(x')$
- 4: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 5: $k \leftarrow TL(u-1) + L(t-1) + l$
- 6: $\llbracket x' \rrbracket \leftarrow \llbracket x' \rrbracket \cdot \llbracket x_{utl} \rrbracket^{10^{w(k-1)}}$
- 7: send $\llbracket x' \rrbracket$ to Trusted Hardware Component and $r_1, r_2, \dots, r_K$ to data user
- 8: **end for**
- 9: Trusted Hardware Component:
- 10: receive $\llbracket x' \rrbracket$
- 11: $x' \leftarrow D_{sk}(\llbracket x' \rrbracket), x'_K | \cdots | x'_2 | x'_1 \leftarrow \text{UnPack}(x')$
- 12: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 13: send x'_{utl} to data user
- 14: **end for**
- 15: Data User:
- 16: receive UTL values x'_{utl} and r_{utl}
- 17: **for** $1 \leq u \leq U, 1 \leq t \leq T, 1 \leq l \leq L$ **do**
- 18: $x_{utl} \leftarrow x'_{utl} - r_{utl}$
- 19: **end for**

6.1 Security Analysis

The security analysis of the SPEC schemes is shown under the semi-trusted model. The adversaries are the data owners, the cloud server, the trusted hardware component and the data users. The data owner can gain access to his/her own data, and not any other data owners' data. With the exception of receiving data from the clouds at the conclusion of the (packing-based) SPA protocol, the data users do not participate in any computations of the SPEC schemes. Therefore, our SPEC schemes protect the data owners' privacy from the data owners and the data users. The cloud server only obtains the ciphertexts of the data owners' data, the ciphertexts of the intermediate results, and the ciphertexts of the final principal eigentensor $\llbracket X \rrbracket$. The ciphertexts are generated by the Paillier cryptosystem with semantic security. Hence, our SPEC schemes are semantically secure and preserve the data owners' privacy from the cloud server. Although the trusted hardware component holds the private key sk , it only obtains some blinded values or random values. Thus, our SPEC schemes also preserve the data owners' privacy from the trusted hardware component.

6.2 Theoretical Evaluations

6.2.1 Accuracy

As previously discussed in Section 4.1, the accuracy of the (efficient) SPEC scheme is affected by the scaling factors. When the scaling factors increase, the accuracy is improved. Once the scaling factors are sufficiently large, the accuracy of the (efficient) SPEC scheme using the collaborative cloud model will be the same as that of the method in the plain-domain, and for the decrypted results of the encrypted

values in the (efficient) SPEC scheme will equal the corresponding times of the values in the plain-domain.

6.2.2 Data Owners Cost and Data Users Cost

After outsourcing the encrypted data to the cloud server, the data owners do not take part in the calculations in the (efficient) SPEC scheme. Consequently, our schemes result in significant savings, in terms of computation and communication resources, to the data owners.

From Algorithm 4 or 8, we determine that data users only need UTL additions in the plain-domain using our (efficient) SPEC scheme. In contrast, the computation complexity for the data users is $O(U_2T_2L_2UTL)$ in each iteration if the computation is undertaken using multilinear mode product power algorithm on the data user's end without considering the privacy concerns. Therefore, for data users, significant computation time reduction is achieved using our (efficient) SPEC scheme.

Thus, we have demonstrated that the SPEC scheme using the collaborative cloud model is lightweight scheme for data owners and data users, and is suitable for deployment on resource-constrained devices such as RFID or mobile devices.

6.2.3 Time Complexity

The time cost of the SPEC scheme using the collaborative cloud model is generated by the (packing-based) SMMP protocol, the STA protocol, the (packing-based) STCD protocol, the (improved) SCNC protocol and the (packing-based) SA protocol. The STA protocol only needs UTL multiplications over encrypted data. The time complexity of the other building block protocols is summarized in Table 1.

TABLE 1
Time Complexity of the Proposed Protocols

Protocol	Encryption	Decryption
SMMP	$4U_2T_2L_2UTL$	$2U_2T_2L_2UTL$
Packing SMMP	$2U_2T_2L_2(\lceil UTL/K \rceil + UTL)$	$2U_2T_2L_2 \lceil UTL/K \rceil$
STCD	$3UTL$	UTL
Packing STCD	$\lceil UTL/K \rceil + 2UTL$	$\lceil UTL/K \rceil$
SCNC	$3UTL$	UTL
Improved SCNC	$\lceil UTL/K \rceil + 2UTL$	$\lceil UTL/K \rceil$
SA	UTL	UTL
Packing SA	$\lceil UTL/K \rceil$	$\lceil UTL/K \rceil$

(Packing-Based) SMMP Protocol: The SMMP protocol needs to execute the SM protocol $U_2T_2L_2UTL$ times. The SM protocol contains 4 encryption operations and 2 decryption operations when plaintexts of two multipliers are not equal. Therefore, the SMMP protocol requires $4U_2T_2L_2UTL$ encryption operations and $2U_2T_2L_2UTL$ decryption operations. The packing-based SMMP protocol requires $2\lceil UTL/K \rceil$ times of packing, which generates $2\lceil UTL/K \rceil$ encryption operations and $2\lceil UTL/K \rceil$ decryption operations in steps 5 and 14 of Algorithm 5. In addition, steps 18 and 24 perform $2U_2T_2L_2UTL$ encryption operations. Consequently, the packing-based SMMP protocol executes $2U_2T_2L_2(\lceil UTL/K \rceil + UTL)$ encryption operations and $2U_2T_2L_2 \lceil UTL/K \rceil$ decryption operations in total.

(Packing-Based) STCD Protocol: The STCD protocol needs 3 encryptions in steps 4, 11 and 15, and 1 decryption in step 9 in each iteration of Algorithm 2. So, the STCD protocol executes $3UTL$ encryptions and UTL decryptions overall. In contrast, the packing-based STCD protocol contains $\lceil UTL/K \rceil$ encryptions and decryptions in steps 3 and 11, which are generated by the packing technology, and $2UTL$ encryptions in steps 14 and 19 of Algorithm 6. Hence, the STCD protocol requires $\lceil UTL/K \rceil + 2UTL$ encryptions and $\lceil UTL/K \rceil$ decryptions.

(Improved) SCNC Protocol: Since the SM protocol contains 3 encryptions and 1 decryption when plaintexts of two multipliers are equal. Therefore, $3UTL$ encryptions and UTL decryptions are required in step 4 of Algorithm 3. In Algorithm 7, due to the use of the packing technology, $\lceil UTL/K \rceil$ encryptions and decryptions are required. Besides, $2UTL$ encryptions are required in steps 14 and 21.

(Packing-Based) SA Protocol: Clearly, UTL encryptions and decryptions are required in steps 4 and 10 of Algorithm 4; while $\lceil UTL/K \rceil$ encryptions and decryptions are required in steps 3 and 11 of Algorithm 8.

6.3 Experimental Evaluations

We now present the evaluations based on synthetic datasets. We implemented the basic SPEC scheme, the efficient SPEC scheme, and the conventional principal eigentensor computing without considering privacy in JAVA. The experiments were executed on computers with Intel(R) Core(TM) i5-3470 CPU, and 16GB RAM. In the experiments, the Paillier cryptosystem and garbled circuits based on FasterGC [19] were adopted. The key size of the Paillier cryptosystem s was set to either 512 bits or 1024 bits long.

In order to evaluate the effectiveness of the (efficient) SPEC scheme, we use the accuracy indicator as the metric defined as: $\rho = 1 - \|X - X'\|/\|X'\|$, where X denotes the principal eigentensor computed by the (efficient) SPEC scheme, and X' represents the principal eigentensor directly calculated by multilinear mode product power algorithm without considering privacy.

The accuracy of the (efficient) SPEC scheme is affected by the scaling factors ρ_1 , ρ_2 and ρ_3 while that of the principal eigentensor computing scheme without any privacy consideration is not. α in Algorithm 1 has one or two decimal places, so hereafter we set $\rho_1 = 100$. ρ_2 and ρ_3 depend on the number UTL of elements in tensor $\mathcal{X}$, so we set $\rho_2 = \sigma(UTL)^2$ and $\rho_3 = \sigma UTU$, where the scaling factors σ are set to 10^2 , 10^3 , 10^4 , 10^5 and 10^6 in the experiments.

Fig. 3. Accuracy of proposed (efficient) SPEC scheme for varying additional scheme scaling factors

Fig. 4. Time of data user using traditional scheme for varying tensor sizes

Fig. 3 illustrates the accuracy of the proposed (packing-based) SPEC scheme for the varying scaling factors. From Fig. 3, it can be observed that the scaling factor plays a crucial part in improving accuracy. And accuracy is low if the scaling factor σ is less than 10^3 . However, accuracy is almost 100% if the scaling factor is greater than 10^4 . This illustrates that the proposed (efficient) SPEC scheme enables the principal eigentensor computing to function and demonstrates the effectiveness of our schemes.

To determine the performance gain for the data user using our proposed (efficient) SPEC scheme, we evaluated the time required in computing principal eigentensor using multilinear mode product power algorithm without any privacy consideration only by the data user (known as traditional scheme), and the time of the data user taken by computing principal eigentensor using our schemes with privacy consideration. The time required using the traditional scheme is depicted in Fig. 4 for varying tensor sizes. We find that the time rapidly increases with tensor size. The time is more than 10 minutes if the tensor size is larger than 4000. At the same time, the time of the data user using our schemes is less than 1 millisecond. The findings demonstrate the efficiency of our (efficient) scheme on the data user's side.

Fig. 5. Time costs of the basic protocols and the improved protocols

To evaluate the building block protocols required in our two SPEC schemes, we compared the basic protocols with the improved protocols. For varying tensor mode length product UTL , the time costs of the basic protocols and the improved protocols are shown in Fig. 5. In the experiments, we assumed the number of the integers packed into one pack is 30 and the key size $s = 512$. From Fig. 5(a), we determine that the time costs of both the SMMP protocol and the packing-based SMMP protocol increase with UTL . For instance, the time increases from 5.5 seconds to 130.5 seconds for the SMMP protocol while the time increases from 3.6 seconds to 86.2 seconds for the packing-based SMMP protocol, when UTL changes from 200 to 1000. At the same time, the packing-based SMMP protocol is more

efficient than the SMMP protocol. For example, the SMMP protocol and the packing-based SMMP protocol take 80.9 seconds and 52.9 seconds respectively when $UTL = 800$. Similarly, the time costs of the STCD protocol and the packing-based STCD protocol are shown in Fig. 5(b). The time costs of the SCNC protocol and the improved SCNC protocol are depicted in Fig. 5(c), and the time costs of the SA protocol and the packing-based SA protocol are depicted in Fig. 5(d). We remark that these experiment results echoed the analysis presented in Section 6.2.3.

Fig. 6. Time costs of the basic SPEC scheme and the efficient SPEC scheme on cloud

Finally, we analyzed the computation costs of the proposed basic SPEC scheme and efficient SPEC scheme by varying the mode length product UTL of the tensor. In our experiments, we supposed the scaling factor $\sigma = 10^5$. Based on the key size $s = 512$, we evaluated the computation costs for our basic SPEC scheme and efficient SPEC scheme. From the findings illustrated in Fig. 6, we observe that the time costs of the basic scheme increase from 16 minutes to 371 minutes for UTL changes from 400 to 2000, while those of the efficient scheme increase from 11 minutes to 252 minutes. In other words, the efficient SPEC scheme achieves a saving of 33.72% in comparison to the basic SPEC scheme.

Several computations of our schemes can be easily parallelized and, in practice, more cloud nodes can be employed. Therefore, the time of our schemes on cloud can be significantly reduced.

7 RELATED WORK

This section reviews related work on secure cloud computing.

Some studies have focused on designing secure cloud models to ensure data security during computation. The fully homomorphic cryptosystem [20] based secure cloud model, for example, uses the fully homomorphic cryptosystem to address data security issues in cloud computing. Using the model, arbitrary functions can be securely computed over encrypted data in the cloud. Yi et al. [21] proposed a secure information retrieval approach, and a new secure searching approach on streaming data [22]. Similarly, Dong et al. [23] presented a private information retrieval protocol. However, due to the prohibitive performance overhead in existing fully homomorphic cryptosystems, the model is far from being practical particularly in a real-world deployment.

Another popular line of research uses partially homomorphic cryptosystem based secure cloud model to securely process data. In such an approach, the cloud offloads majority, or all, computations of the data users in applications such as secure facial expression recognition [24], secure back-propagation neural network learning [25] over unencrypted data in the cloud, and secure query processing [26], secure large-scale similarity search [27] and secure support vector machine [17] over encrypted data in the cloud.

However, our work is significantly different from these existing work. Our work combines the advantages of the partially homomorphic cryptosystem and the garbled circuits to perform data processing tasks without compromising on data security. Furthermore, using the model, we implemented the first secure and efficient principal eigentensor computation in the cloud.

8 CONCLUSION

With the increasing popularity of cloud computing, how to design secure computation is critical in sustainable cloud-assisted cyber-physical-social systems (CPSS). In this paper, we presented the first basic SPEC scheme and the first efficient SPEC for sustainable cloud-assisted cyber-physical-social systems, which can leverages cloud server and trusted hardware component to securely execute all jobs in principal eigentensor computation over encrypted data without the data owners' participation while protecting users' privacies. We demonstrated the security and utility of the schemes (i.e. the collaborative cloud model is suitable for cyber-physical-social systems big data processing on the cloud server since most of the tensor based computation can be conducted securely and efficiently). Our schemes substantially relieve the burden of data storage and computation on the users. Due to the lightwightness of our schemes, resource-constrained users can also deploy our schemes with ease.

REFERENCES

- [1] K. Gai, M. Qiu, H. Zhao, and X. Sun, "Resource Management in Sustainable Cyber-Physical Systems Using Heterogeneous Cloud Computing," *IEEE Trans. Sustainable Computing*, 2017, DOI: 10.1109/TSUSC.2017.2723954.
- [2] X. Wang, L. T. Yang, X. Chen, J. Han, and J. Feng, "A Tensor Computation and Optimization Model for Cyber-Physical-Social Big Data," *IEEE Trans. Sustainable Computing*, 2017, DOI: 10.1109/TSUSC.2017.2777503.
- [3] X. Wang, L. T. Yang, H. Liu, and M. J. Deen, "A Big Data-as-a-Service Framework: State-of-the-art and Perspectives," *IEEE Trans. Big Data*, 2017, DOI: 10.1109/TBDATA.2017.2757942.
- [4] L. T. Yang, G. Huang, J. Feng, and L. Xu, "Parallel GNFS Algorithm Integrated with Parallel Block Wiedemann Algorithm for RSA Security in Cloud Computing," *Information Sciences*, vol. 387, pp. 254–265, 2017.
- [5] K.-K. R. Choo, Y. Fei, Y. Xiang, and Y. Yu, "Embedded Device Forensics and Security," *ACM Trans. Embedded Computing Systems*, vol. 16, no. 2, p. 50, 2016.
- [6] Y. Yang, J. K. Liu, K. Liang, K.-K. R. Choo, and J. Zhou, "Extended Proxy-Assisted Approach: Achieving Revocable Fine-Grained Encryption of Cloud Data," in *Proc. 20th European Symp. Research in Computer Security (ESORICS)*, 2015, pp. 146–166.
- [7] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in Cloud Computing: Opportunities and Challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.
- [8] T. Zhang and R. B. Lee, "CloudMonatt: An Architecture for Security Health Monitoring and Attestation of Virtual Machines in Cloud Computing," in *Proc. ACM/IEEE 42nd Int'l Symp. Computer Architecture (ISCA)*, 2015, pp. 362–374.
- [9] R. Zhang, R. Xue, T. Yu, and L. Liu, "Dynamic and Efficient Private Keyword Search over Inverted Index-Based Encrypted Data," *ACM Trans. Internet Technology*, vol. 16, no. 3, p. 21, 2016.
- [10] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: A Panacea for Healthcare Cloud-Based Data Security and Privacy?" *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, 2018.
- [11] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms," *ACM Trans. Intelligent Systems and Technology*, vol. 8, no. 2, p. 16, 2016.
- [12] P. Wang, L. T. Yang, G. Qian, and F. Lu, "The Cyber-Physical-Social Transition Tensor Service Framework," *IEEE Trans. Sustainable Computing*, 2018, DOI: 10.1109/TSUSC.2018.2799940.
- [13] P. Paillier, "Public-key Cryptosystems Based on Composite Degree Residuosity Classes," in *Proc. 17th Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 1999, pp. 223–238.
- [14] Y. Lindell and B. Pinkas, "A Proof of Security of Yao's Protocol for Two-Party Computation," *J. Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.
- [15] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-Preserving Matrix Factorization," in *Proc. ACM 20th Conf. Computer and Communications Security (CCS)*, 2013, pp. 801–812.
- [16] Y. Lindell, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017.
- [17] F.-J. González-Serrano, Á. Navia-Vázquez, and A. Amor-Martín, "Training Support Vector Machines with Privacy-Protected Data," *Pattern Recognition*, vol. 72, pp. 93–107, 2017.
- [18] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine Learning Classification over Encrypted Data," in *Proc. 22nd Ann. Network and Distributed System Security Symp. (NDSS)*, 2015.
- [19] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster Secure Two-Party Computation Using Garbled Circuits," in *Proc. 20th USENIX Security Symp. (USENIX Security)*, 2011.
- [20] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," in *Proc. 29th Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2010, pp. 24–43.
- [21] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-

Database Private Information Retrieval from Fully Homomorphic Encryption," *IEEE Trans. Knowledge and Data Eng.*, vol. 25, no. 5, pp. 1125–1134, 2013.

- [22] X. Yi, E. Bertino, J. Vaidya, and C. Xing, "Private Searching on Streaming Data Based on Keyword Frequency," *IEEE Trans. Dependable and Secure Computing*, vol. 11, no. 2, pp. 155–167, 2014.
- [23] C. Dong and L. Chen, "A Fast Single Server Private Information Retrieval Protocol with Low Communication Cost," in *Proc. 19th European Symp. Research in Computer Security (ESORICS)*, 2014, pp. 380–399.
- [24] Y. Rahulamathavan, R. C.-W. Phan, J. A. Chambers, and D. J. Parish, "Facial Expression Recognition in the Encrypted Domain Based on Local Fisher Discriminant Analysis," *IEEE Trans. Affective Computing*, vol. 4, no. 1, pp. 83–92, 2013.
- [25] J. Yuan and S. Yu, "Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 1, pp. 212–221, 2014.
- [26] H. Shafagh, A. Hithnawi, A. Dröscher, S. Duquenooy, and W. Hu, "Talos: Encrypted Query Processing for the Internet of Things," in *Proc. 13th ACM Conf. Embedded Networked Sensor Systems*. ACM, 2015, pp. 197–210.
- [27] Q. Wang, M. He, M. Du, S. S. Chow, R. W. Lai, and Q. Zou, "Searchable Encryption over Feature-Rich Data," *IEEE Trans. Dependable and Secure Computing*, vol. 125, no. 3, pp. 496–510, 2016.

Jun Feng is currently studying towards the Ph.D degree in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and in the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen, China. He received the master's degree from the School of Computer Science and Technology, Guizhou University, Guiyang, China, in 2013. His research interests include big data security and privacy, and cloud computing security and

privacy.

Laurence T. Yang received the B.E. degree in Computer Science and Technology from Tsinghua University, Beijing, China, and the Ph.D. degree in Computer Science from the University of Victoria, Victoria, BC, Canada. He is a professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and in the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. He is also a researcher in the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen, China. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data. His research has been supported by the National Sciences and Engineering Research Council, Canada, and the Canada Foundation for Innovation.

Qing Zhu received the B.E. degree in College of Information and Management Science from Henan Agricultural University, Zhengzhou, China, in 2013 and the master's degree in School of Computer Science and Technology from Huazhong University of Science and Technology, Wuhan, China, in 2016. Currently, she is a staff at State Grid Corporation of China. Her research interests include big data and sustainable information security.

Yang Xiang received his PhD in Computer Science from Deakin University, Australia. He is the Dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia. His research interests include cyber security, which covers network and system security, data analytics, distributed systems, and networking. He has published more than 200 research papers in many international journals and conferences. He served as the Associate Editor of *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Security and Communication Networks* (Wiley), and the Editor of *Journal of Network and Computer Applications*.

Jinjun Chen received his Ph.D degree from the Swinburne University of Technology. Currently, he is a professor from Faculty of Science, Engineering and Technology, the Deputy Director of Swinburne Data Science Research Institute and the Director of Data Science Platforms and Systems Program in Swinburne University of Technology, Australia. His research interests include big data, data science, scalable software systems, cloud, security and privacy. He has published more than 130 papers in high-quality journals and conferences. He received various awards such as Vice Chancellor Research Award.

Zheng Yan received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively, the second MEng degree in information security from the National University of Singapore, Singapore, in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland, in 2005 and 2007, respectively. She is currently a professor with the Xidian University, Xi'an, China and a visiting professor with the Aalto University, Espoo, Finland. Her research interests include trust, security and privacy, social networking, cloud computing, networking systems, and data mining.