IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 13, 2017, accepted January 31, 2017, date of publication February 9, 2017, date of current version March 13, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2666839

Context-Aware Verifiable Cloud Computing

ZHENG YAN[1,2], (Senior Member, IEEE), XIXUN YU[1], AND WENXIU DING[1]
[1]State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an 710071, China
[2]Department of Communications and Networking, Aalto University, 02150 Espoo, Finland

Corresponding author: Z. Yan (zyan@xidian.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800704, in part by the NSFC under Grant 61672410 and Grant U1536202, in part by the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China under Program 2016ZDJC-06, in part by the Ph.D. Grant of the Ministry of Education, China, under Grant JY0300130104, in part by the 111 Project Grant B08038 and Grant B16037, and in part by Aalto University.

ABSTRACT Internet of Things (IoTs) has emerged to motivate various intelligent applications based on the data collected by various ''things.'' Cloud computing plays an important role for big data processing by providing data computing and processing services. However, cloud service providers may invade data privacy and provide inaccurate data processing results to users, and thus cannot be fully trusted. On the other hand, limited by computation resources and capabilities, cloud users mostly cannot independently process big data and perform verification on the correctness of data processing. This raises a special challenge on cloud computing verification, especially when user data are stored at the cloud in an encrypted form and processed for satisfying the requests raised in different contexts. But the current literature still lacks serious studies on this research issue. In this paper, we propose a context-aware verifiable computing scheme based on full homomorphic encryption by deploying an auditing protocol to verify the correctness of the encrypted data processing result. We design four optional auditing protocols to satisfy different security requirements. Their performance is evaluated and compared through performance analysis, algorithm implementation, and system simulation. The results show the effectiveness and efficiency of our designs. The pros and cons of all protocols are also analyzed and discussed based on rigorous comparison.

INDEX TERMS Cloud computing, full homomorphic encryption, privacy protection, verifiable cloud computing.

I. INTRODUCTION

Internet of Things (IoT) is going to create a world where physical objects are seamlessly integrated into information networks in order to provide advanced and intelligent services for human-beings. The interconnected ''things'' such as sensors and mobile devices sense, monitor and collect all kinds of data about human social life. These data can be further aggregated, fused, processed, analyzed and mined in order to extract useful information to enable intelligent and ubiquitous services. The IoT is evolving as an important part of next generation networking paradigm and service infrastructure. Various applications and services of IoT have been emerging into markets in broad areas, e.g., surveillance, health care, security, transport, food safety, and distant object monitor and control.

On the other hand, cloud computing offers a new way of service provision by re-arranging various resources (e.g., storage, computing and applications) and providing them to users based on their demands. The cloud computing provides a large resource pool by linking network resources together. It has desirable properties, such as scalability, elasticity, fault-tolerance, and pay-per-use. Thus, it has become a promising service platform by rearranging the way that information technology services are provided and consumed.

A. MOTIVATIONS

In practice, cloud computing can cooperate with IoT by providing computing services in order to release the load of big data processing at user devices and some service providers. One practical scenario is that the data monitored or sensed from the 'things' can be aggregated and sent to the cloud to process in order to provide data computation and processing results to requesting parties (e.g., IoT service providers). However, Cloud Service Provider (CSP) is a party that cannot be fully trusted by IoT data providers and data requesting parties. The CSP could disclose the privacy of data providers or owners by maliciously miming data. It may provide wrong data processing results to the requesting parties to

intentionally destroy IoT service quality. In this case, how to ensure the facticity and genuine of data sources and the correctness of IoT data processing, computation, and mining becomes a practically crucial issue that greatly impacts the continuous success of IoT and cloud computing, as well as the future Internet.

Since the CSP cannot be fully trusted and the privacy of monitored objects should be preserved, data collected by 'things' are generally provided to the CSP in an encrypted form for further processing. In practice, different types of data could be collected and processed at the cloud. For example, a personal mobile phone can collect its user information about location, calling, radio connec-tivity quality, inbound/outbound data traffic, personal heart beat rate, blood pressure, breathing vol-ume/frequency and so on. The collected data can be fur-ther processed and used by different IoT services to offer a diversity of smart services. The algorithms used for computing or processing different types of data could be different at the cloud. For the purpose of preserving data privacy, the cloud processes the encrypted data and pro-vides processing results to requesting parties mostly in an encrypted form. How to ensure, audit and verify the facticity and correctness of data provision and processing in different contexts is a practically significant issue. In the literature, verifiable computing with context awareness is still an open research topic that has not been well investigated [1], [31], [37], [40].

B. MAIN CONTRIBUTIONS

In this paper, we propose a scheme of verifiable compu-ting with context awareness and privacy preservation in IoT cloud computing. We first apply full homomorphic encryption (FHE) technologies to process data in an encrypted form at CSP in order to protect the privacy of data providers and data owners. We further deploy an auditing protocol to verify the correctness of encrypted data processing by applying a Trusted Auditing Proxy (TAP). Concretely, a Data Provider (DP) encrypts its col-lected data with the homomorphic key offered by the TAP and signs it with a data context identifier (ID). Then it transmits the encrypted data, context ID and the signa-ture to the CSP as an input of multi-party computation. The CSP computes the encrypted data from all DPs based on the context IDs by selecting a corresponding algorithm and signs the computation result (which is in an encrypted form). For accessing the computation result, a requesting party (RP) requests the result from the CSP with regard to a context, the CSP passes the request to the TAP to check its eligibility in order to allow the RP to access the data processing result.

When the RP wants to verify the correctness of data processing and computation of CSP, it reports the pro-cessing result signed by the CSP and its hash code to the TAP. The TAP queries the CSP to get the encrypted data with regard to the RP request in order to audit the data processing at the CSP. For supporting the CSP to check the facticity and genuine of data sources, the scheme re-quests that DP signs its provided data regarding a context in order to allow the TAP later on to figure out malicious DPs during auditing by finding malicious data input through analysis and mining. We design four auditing protocols to satisfy different security requirements. Their performance is evaluated and compared in order to show the pros and cons of each protocol and its feasibility in different scenarios.

Specifically, the contribution of this paper can be summarized as below:

1) We motivate context-aware verifiable computing for cloud and propose an effective scheme to achieve both cloud data privacy and verifiability of cloud data processing.

2) To the best of our knowledge, our scheme is one of the first to realize verifiable cloud computing with con-text-awareness. It supports various data processes by applying full homomorphic technologies and deploying an auditing protocol to verify the correctness of encrypted data processing. The proposed scheme can play as a generic framework for verifiable computing in cloud.

3) We propose four optional auditing protocols in or-der to satisfy different security and performance re-quirements. Three of them can ensure system security in case that RPs could collude with CSP.

4) We analyze scheme security and evaluate the performance of the proposed protocols through rigorous analysis and comparison in order to show their pros and cons, as well as applicability.

II. RELATED WORK

A. PRIVACY-PRESERVING DATA MINING (PPDM)

Privacy-Preserving Data Mining (PPDM) aims to support data mining related computations, processes or operations with privacy preservation [31]. PPDM is a 'must-solve' problem in IoT for securely and intelligently supporting various IoT services in a pervasive and personalized way. From the practical point of view, PPDM is still a challenge, considering trustworthiness, computation complexity and communication cost [31].

Mishra and Chandwani proposed an architecture to enable Secure Multi-party Computation (SMC) by hiding the identities of the parties that take part in the process of Business Process Outsourcing [3]. A class of functions was proposed to provide additional abilities to a party to split its huge data before providing it for computation, making it almost intractable for other parties to know the actual source of the data in order to achieve secure and privacy-preserving data mining. Liu et al. proposed a secure multi-party multi-data ranking protocol and a secure multi-party addition protocol to complete private-preserving sequential pattern mining [4].

A number of operations on securely input data are supported in PPDM. Zhu et al. proposed schemes for securely extracting knowledge from two or more parties' private data [5]. They presented three different approaches to privacy-preserving Add to Multiply Protocol, as well as further extended it to privacy-preserving Adding to Scalar Product Protocol. Wang and Luo studied a private preserving

shared dot product protocol that is a main building block of various data mining algorithms with privacy concerns, and provides fundamental security guarantee for many PPDM algorithms [6]. They constructed a privacy-preserving two-party shared dot product protocol based on some basic cryptographic techniques, which is provably secure in a semi-honest model. Shen, Han and Shan proposed a Horizontal Distribution of the Privacy Protection DK-Means (HDPPDK-Means) algorithm based on Horizontal partitioned database and DK-means idea to realize distributed clustering and applied a secure multi-party computation protocol to achieve privacy preservation [7].

Many studies focused on supporting specific data mining techniques with privacy preservation. Statistical hypothesis test is an important data analysis technique. Liu and Zhang investigated nonparametric Sign Test (NST) theory in such a context that two parties, each with a private dataset, would like to conduct a sign test on their joint dataset, but neither of them is willing to disclose its private dataset to any other third parties [8]. Their proposed protocol does not make use of any third party nor cryptographic primitives.

Association rule mining is one of the hottest research areas, which investigates the automatic extraction of previously unknown patterns or rules from large amounts of data. Zhan et al. developed a secure protocol for multiple parties to conduct this desired computation in a distributed way and to exchange the data while keeping it private by using homomorphic encryption techniques [9]. Kantarcioglu and Clifton [10] proposed two protocols to implement privacy-preserving mining of association rules over horizontally partitioned data. Zhang and Zhao [11] further revised its security proof. Privacy-preserving association rule mining was surveyed by Wang [12] with regard to basic concepts, general principles and methods.

In the case that agencies want to conduct a linear regression analysis with complete records without disclosing values of their own attributes, Ashish et al. described an algorithm that enables agencies to compute the exact regression coefficients of the global regression equation and also perform some basic goodness-of-fit diagnostics while protecting the confidentiality of their data [13]. This work can be applied for distributed computation for regression analyses in data mining.

Finding the nearest k objects to a query object (k-NN queries) is a fundamental operation for many data mining algorithms to enable clustering, classification and outlier detection tasks. Efficient solutions for k-NN queries for vertically partitioned data were proposed by Amirbekyan and Estivill-Castro [17]. These solutions include L ∞ (or Chessboard) metric as well as detailed privacy-preserving computation of all other Minkowski metrics, privacy preserving algorithms for combinations of local metrics into a global metric that handles large dimensionality and diversity of attributes common in vertically partitioned data, a privacy-preserving SASH (a very successful data structure for associative queries in high dimensions) for managing very large data sets. Liu *et al.* [14] presented privacy preserving algorithms for DBSCAN clustering for the horizontally, vertically and arbitrarily partitioned data distributed between two parties. DBSCAN [15] is also a popular density-based clustering algorithm for discovering clusters in large spatial databases with noise.

Gradient descent is a widely used method for solving optimization and learning problems. Wan *et al.* [16] presented a generic formulation of secure gradient descent methods with privacy preservation. It underlies many commonly used techniques in data mining and machine learning, such as neural networks, Bayesian networks, genetic algorithms, and simulated annealing.

Current solutions of PPDM are still not practical. The existing methods are impractical, ineffective, inefficient or inflexible with regard to generality, trustworthiness, computation complexity and communication cost. Seldom, a PPDM protocol can satisfy all essential requirements for practical usage. Particularly, existing studies have not investigated all data mining related operations with privacy preservation. In addition, few studies touched the issue of verifiable computing if the input data are encrypted and the data mining process is conducted in an encrypted form by CSP that cannot be fully trusted. Existing work has not studied how to verify whether the encrypted data processing result is correct when the processed data is in an encrypted form. A generic and feasible framework for verifiable computing is still missed in the literature.

B. SMC APPLICATIONS

Secure Multi-party Computation (SMC) deals with the problem of secure computation among participants who are not trusted with each other, particularly with the preference of privacy preserving computational geometry. SMC refers to the parties, who participate in the computation with their own secret inputs, wish to cooperatively compute a function. When the computation is over, each party can receive its own correct output with correctness insurance, and know its own output only with privacy preservation. It is an important research topic in IoT. The problems of SMC are specifically different in different scenarios. Based on the problems solved, SMC mechanisms can be classified into four categories: privacy-preserving database query, privacy-preserving scientific computation, privacy-preserving intrusion detection and privacy-preserving data mining. A detailed survey on SMC technologies is provided in [31].

SMC together with homomorphic encryption is widely applied into many areas, such as distributed electronic contract management [18], smart meter based load management [19], healthcare frauds and abuses [20], policy-agile encrypted networking for defense, law enforcement, intelligence community, and commercial networks [21], privacy preserving path inclusion [28], privacy preserving string matching [22], privacy-enhanced recommender system in a social trust network [23], user profile matching in social networking [24], credit check applications [25], private collaborative forecasting and benchmarking [26], privacy-preserving

genomic computations [27], protection against insider threats (e.g., business partners) [29], privacy preserving electronic voting [30], and so on. But one important issue missed in the above studies is verifiable computing that can audit the correctness of encrypted data processing.

C. CLOUD COMPUTING AUDITING

Current researches about verifiable cloud computing focus on auditing cloud data storage and data integrity with regard to data management, such as insertion, deletion, and addition [1], [37], [40]. It has not yet investigated data calculation and computation seriously. Yang et al. proposed an approach to fast detect data errors in big sensor data sets based on a scale-free network topology and most of detection operations can be conducted in limited temporal or spatial data blocks instead of a whole big data set [38]. Some researchers applied a MapReduce framework to anonymize large-scale data sets in cloud [39].

We notify that all above presented work did not consider how to solve the problems of verifiable computing with context awareness as described in Section 1. The literature still lacks a generic and feasible framework to provide verifiable computing on encrypted data processing and computation in cloud.

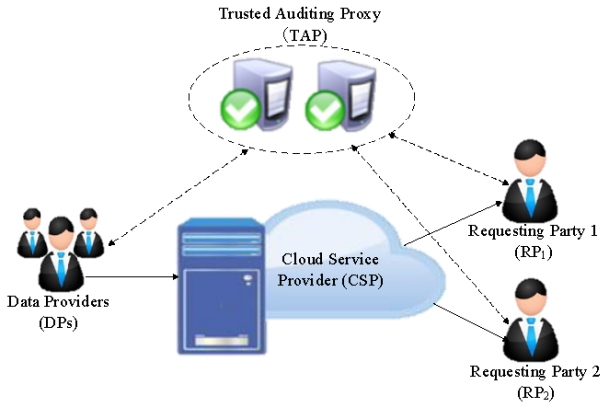

FIGURE 1. System model.

III. PROBLEM STATEMENT

A. SYSTEM AND THREAT MODELS

We consider an IoT cloud computing system that involves four different kinds of entities, as illustrated in Fig.1: the Data Providers (DPs) that interact with the physical world, detect/monitor/sense information of objects in different contexts and provide collected data to a CSP for processing; the CSP that has functions and capabilities that the DPs do not have and can process the data provided by the DPs. It could be curious to explore the privacy of physical objects based on the data provided by the DPs although CSP follows the design of system protocols. Thus, the DPs encrypt the data provided to the CSP; the TAP is responsible for issuing essential keys to DPs for homomorphic encryption, checking the eligibility of data access at the CSP, issuing access keys to eligible Requesting Parties (RPs), and auditing the facticity and genuine of data sources and the correctness of CSP data processing and computation; the RP that requests CSP's data processing results in different contexts for offering intelligent and ubiquitous services to IoT end users. We assume that all system entities communicate with each other via a secure channel (e.g., OpenSSL).

In the system, TAP and CSP do not collude. RP can only access the final data processing results from CSP. CSP has no rights to offer the data collected from DPs to RPs. RP can request TAP and delegate TAP to audit the facticity and genuine of data collection and processing. We assume that the algorithms used at CSP can be supported by full homomorphic encryption. That is the output of the multiple encrypted data that input into a computing algorithm is equal to the encrypted result of corresponding multiple plain data input into the same computing algorithm. And the encrypted result can be decrypted with a corresponding decryption key. CSP cannot be fully trusted. It could be curious on raw data provided by DPs and may provide wrong data processing results to RPs.

Collusion between CSP and its users (e.g., RPs) is quite possible. In practice, such collusion could make CSP lose reputation due to information leakage. A negative impact of bad reputation is CSP will lose its users. The deduction of the number of CSP users will finally make it lose profits. On the other hand, CSP users could lose their convenience and benefits of outsourcing data and data processing at CSP due to its bad reputation. Thus, the collusion between CSP and its users is not profitable for both of them. Concrete analysis based on Game Theory is provided in [36]. Therefore, we have a good reason to hold such an assumption: CSP does not collude with its users, e.g., colluding with some RPs to gain raw data and allow other RPs to access data stored at CSP. But some incidental collusion between CSP and RPs could happen, which requests higher level security assurance in the system design.

B. DESIGN GOALS

To achieve trustworthy data processing and avoid potential risks in cloud services, our design should achieve the following security and performance goals: 1) Security and safety: it is hard or impossible for CSP to get the raw data during data processing; the data processed at the cloud can only be accessed by eligible system entities; 2) Generality: the proposed scheme supports various data processes in different contexts in the cloud. 3) Reasonable overhead: the scheme fulfills data processing and auditing with reasonable computation and communication costs.

IV. TAP 4 PROPOSED SCHEME

A. PRELIMINARIES, NOTATIONS AND DEFINITIONS

1) FULL HOMOMORPHIC ENCRYPTION

Full Homomorphic Encryption (FHE) mainly consists of four algorithms: Key-Generate (KG), Encrypt (E), Decrypt (D)

TABLE 1. Notations.

key	Description	Remark
PK_H	The public key for homomorphic encryption;	This key can be dynamically changed. Multiple such keys exist in the system.
SK_H	The private key for homomorphic decryption;	This key is used for homomorphic decryption.
PK_x	The public key of entity x;	The entity can be DP, RP, CSP or TAP.
SK_x	The secret key of entity x;	
DP_i	The ith data provider;	
RP_k	The kth requesting party;	
C_j	The jth context;	Parameter j is the data context identifier (ID).
$D_{i,j}$	The data provided by DP_i in context C_j;	
$E(PK_H, D_{i,j})$	The homomorphic encryption of $D_{i,j}$ with PK_H;	
F_j	The data processing function or algorithm w.r.t. C_j at CSP;	
DM_j	The plain data processing result in context C_j;	
$E(PK_H, DM_j)$	The encrypted data processing result in context C_j at CSP;	
$E'(PK_x, m)$	The public key encryption function using encryption key PK_x to encrypt m;	
$E(PK_y, m)$	The homomorphic encryption function using encryption key PK_y to encrypt m;	
$Sign(SK_x, m)$	The signature of data m signed by SK_x.	

and Evaluate ($EV(pk, Cir, c)$, where pk is a public key used to calculate ciphertexts. In these four algorithms, the Evaluate algorithm does computation on a set of ciphertext $c = \langle c_1, \ldots, c_t \rangle (c_i = E(pk, m_i))$, which is the input of circuit Cir. Circuit Cir represents a function or an algorithm. Based on these four algorithms, we define FHE as below.

For any key pair (pk, sk) generated by KG algorithm, any circuit Cir, any plaintext $m = \langle m_1, \ldots, m_t \rangle$, and any ciphertext $c = \langle c_1, \ldots, c_t \rangle$ $(c_i = E\ (pk, m_i))$:

$$\text{if } c^{'} = EV(pk, Cir, c) \quad (1)$$

$$\text{then } D\left(sk, c^{'}\right) = Cir(m_1, \ldots, m_t). \quad (2)$$

This means that we can do operations on the ciphertext in order to get the encrypted version of the result of plaintext operations. In scheme implementation and evaluation, we use Brakerski Gentry Vaikuntanathan (BGV) FHE [32].

2) LIMITATIONS OF PARTIAL HOMOMORPHIC ENCRYPTION

Partial Homomorphic Encryption (PHE) is a cryptographic algorithm that can achieve the same goal of processing ciphertexts as that of the FHE. It preforms much better than FHE in terms of computation and storage overhead. However, it can only support some specific operations, e.g., addition and subtraction. This means it is only applicable in some specific scenarios. The goal of our work is to design a generic scheme of verifiable cloud computing that can adapt to various operations, thus can be applied into different applications. Due to this reason, we choose the FHE to demonstrate the design of our scheme, even though it has high computational complexity and high storage consumption.

3) NOTATIONS

For easy presentation, the notations used in the proposed scheme are summarized in Table 1.

B. SYSTEM SETUP

During system setup, each system entity x generates its own public and private key pairs: PK_x and SK_x. TAP generates PK_H and SK_H, and issues PK_H to DP_i(i $= 1, \ldots,$ I). Note that this key can be dynamically changed or different with regard to different contexts. Multiple PK_H keys could exist in the system. Herein, for simplification, we only use PK_H to denote the homomorphic encryption key and SK_H the homomorphic decryption key. Meanwhile, each system entity announces its public key PK_X to other system entities.

C. SYSTEM PROCEDURE

System procedure is briefly described in Fig.2. DPs provide sensed data in different contexts to CSP. CSP processes and computes those data based on the context ID by applying different algorithms. If RP requests the result of data processing from CSP, CSP passes the request to TAP for eligibility check. If the check is positive, TAP issues the decryption key to RP that can be gained with the RP's private key. Thus, RP can access the requested data. In case that RP wants to check the correctness of data processing and computation at CSP, it approaches TAP and provides the evidence that allows TAP to conduct auditing.

Protocol 1:

1) DATA PROVISION

For DP_i to provide data $D_{i,j}$ collected in context C_j to CSP and in order to preserve the privacy of the monitored objects and the data itself, DP_i encrypts $D_{i,j}$ using PK_Hissued by TAP as $E(PK_H, D_{i,j})$. Meanwhile, it signs the hash code of data package $P(D_{i,j}) = \{E(PK_H, D_{i,j}), C_j\}$ for non-repudiation verification on data provision. DP_i then sends $P(D_{i,j})$, $Sign(SK_{DP_i}, P(D_{i,j}))$ to CSP.

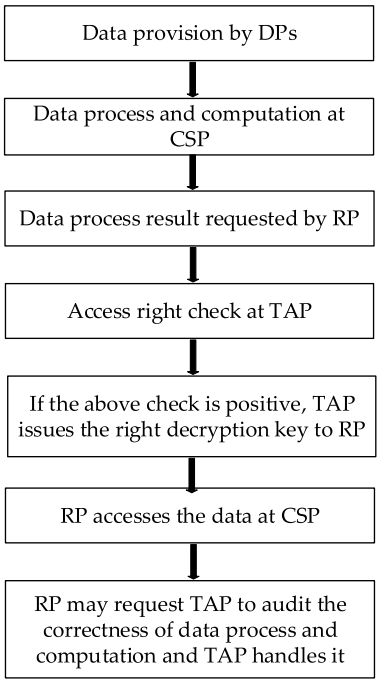

FIGURE 2. System procedure.

2) PRIVACY PRESERVING DATA COMPUTING

CSP processes data, it selects algorithm F_j based on C_j to process the collected encrypted data $E(PK_H, D_{i,j})$ in context C_j and gains the encrypted form of data processing result $E(PK_H, DM_j)$, that is: $E(PK_H, DM_j) = F_j E(PK_H, D_{i,j})\}$, $(i = 1, \ldots, I)$.

3) RP DATA REQUEST AND AUTHORIZATION

RP_k requests CSP for the result of data processing and computation in C_j by sending a requesting message that contains $R_k = \{PK_{RP_k}, C_j\}$ and $Sign(SK_{RP_k}, R_k)$. Once receiving the request, the CSP passes the request to TAP for checking its access eligibility. If the check based on the current access policy is positive, the TAP issues encrypted SK_H, i.e., $E'(PK_{RP_k}, SK_H)$ with RP_k's public key based on a public key encryption scheme (e.g., RSA). TAP issues $E'(PK_{RP_k}, SK_H)$ to RP directly or through CSP.

4) DATA ACCESS

CSP receives $E'\left(PK_{RP_k}, SK_H\right)$, which means TAP issues RP_k the right to access DM_j. It then delivers the data package$E\left(PK_H, DM_j\right)$, $E'\left(PK_{RP_k}, SK_H\right)$, $Sign(SK_{CSP}, E(PK_H, DM_j), E'(PK_{RP_k}, SK_H), C_j)$, and C_j to RP_k. After receiving the package, RP_k can decrypt $E'(PK_{RP_k}, SK_H)$ with its SK_{RP_k} to get SK_H, which is further used to get the plaintext of DM_j.

5) DATA AUDITING

RP may not trust the processing result of CSP. In this case, it requests TAP to audit the correctness of data processing and computation by providing C_j, the hash code of DM_j, $h\left(DM_j\right)$, the signature of CSP data provision, i.e., $Sign_{CSP} = Sign(SK_{CSP}, E(PK_H, DM_j), E'(PK_{RP_k}, SK_H), C_j)$. Notably, the auditing request should be signed by RP to ensure non-repudiation. Thereby, we get the package of an auditing request $AR_k = \{C_j, h\left(DM_j\right), Sign_{CSP}, Sign(SK_{RP_k}, \{C_j, h\left(DM_j\right), Sign_{CSP})\}$. After receiving AR_k, TAP handles it by querying CSP to get F_j and all $E(PK_H, D_{i,j})$ used for generating $E(PK_H, DM_j)$. TAP decrypts $E(PK_H, D_{i,j})$ to get all $D_{i,j}$ and input them into F_j to get plain DM_j, that is $DM_j = F_j(\{D_{i,j}\})$ $(i = 1, \ldots, I)$. TAP further compares the hash code of DM_j output from F_j and the one provided by RP in order to judge if the data computation and processing at CSP is correct.

TAP can also further investigate the facticity and genuine of data sources (DPs). It studies the abnormality of data collection based on historical data mining and pattern learning. Fig.3 shows the detailed protocol as described above.

D. CSP-RP COLLUSION ATTACK AND SCHEME IMPROVEMENT

In this sub-section, we improve the above protocol in order to resist CSP-RP collusion attack although the incentive of this collusion is not high based on our previous analysis [36]. In the CSP-RP attack, CSP may collude with RP_k to gain the data processing result DM_j. Based on the above protocol, eligible RP_k can finally get $E'(PK_{RP_k}, SK_H)$ encrypted with its own public key, and further get the homomorphic secret key SK_H. If RP_k colludes with CSP, it discloses SK_H to CSP. Thus, CSP is able to decrypt data package $P(D_{i,j})$, which means the raw data provided by DP is exposed to CSP and the security of the system is broken. For enhancing system security, we propose Protocol 2 that improves Protocol 1 and resists the CSP-RP collusion attack. In this protocol, it is impossible for RP to know SK_H, thus it can overcome the risk raised by RP to disclose SK_H to CSP. The details of this protocol are illustrated in Fig. 4 and described below.

Protocol 2:

1) DATA PROVISION

This step is the same as the data provision described in Fig. 3.

2) PRIVACY PRESERVING DATA COMPUTING

This step is the same as the privacy preserving data computation as described above in Fig 3.

3) RP DATA REQUEST AND AUTHORIZATION

RP_k requests CSP for the result of data processing and computation in C_j by sending a requesting message that contains $R_k = \{PK_{RP_k}, C_j\}$ and $Sign(SK_{RP_k}, R_k)$. Once receiving the request, the CSP passes the request to TAP for checking its access right. If the check based on the current access policy is positive, the TAP requests the data processing result $E\left(PK_H, DM_j\right)$ from CSP. After receiving $E\left(PK_H, DM_j\right)$ with $Sign_{CSP} = Sign(SK_{CSP}, \{E(PK_H, DM_j)\})$, the TAP first decrypts $E\left(PK_H, DM_j\right)$ to get DM_j, and then re-encrypts

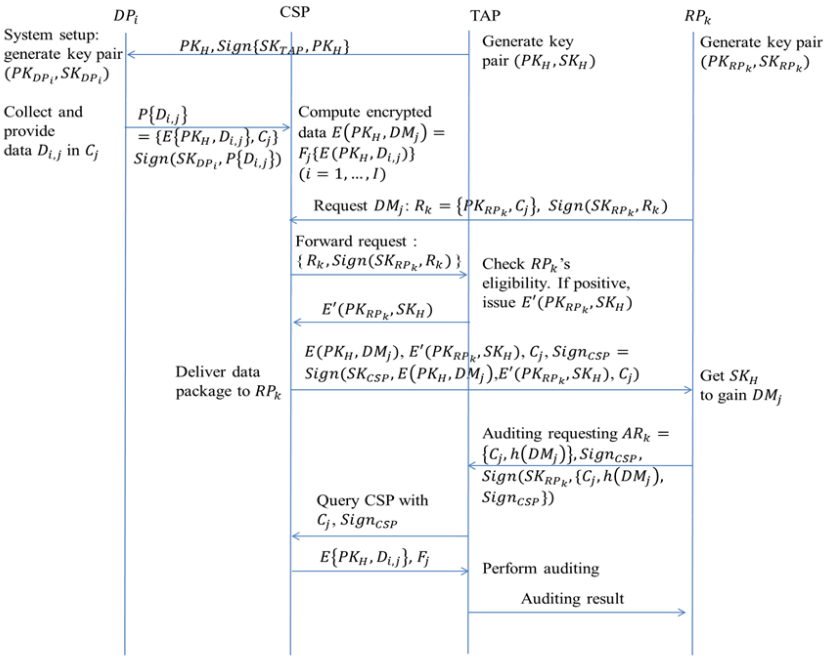

FIGURE 3. Protocol 1 - context-aware verifiable computing protocol with auditing.

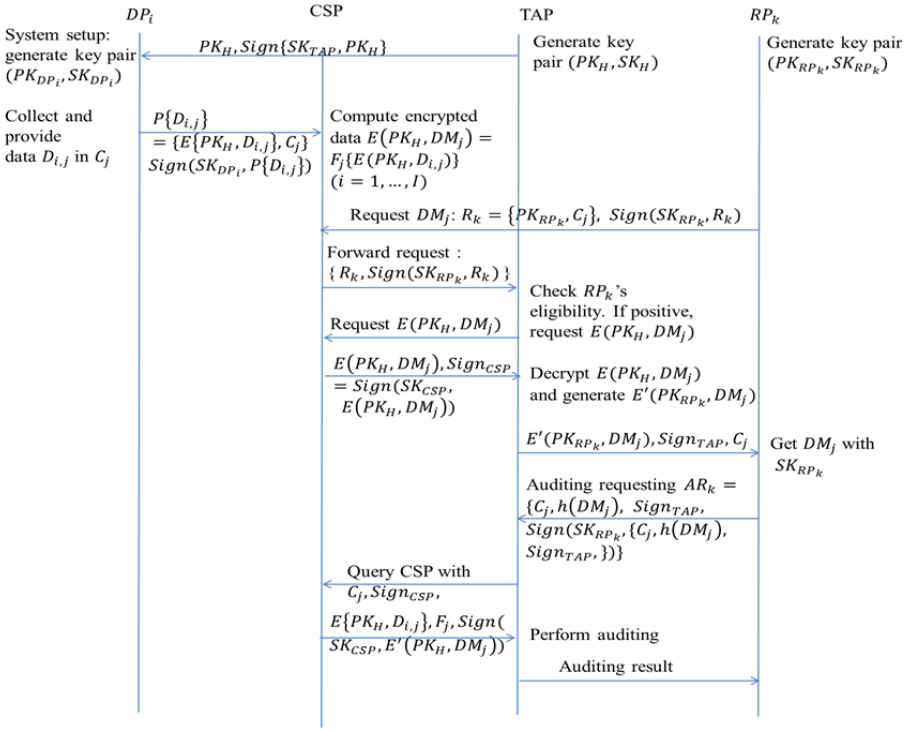

FIGURE 4. Protocol 2 - improved verifiable computing protocol for resisting CSP-RP collusion attack.

DM_j with the RP_k's public key PK_{RP_k} to get $E'(PK_{RP_k}, DM_j)$ based on a public key encryption scheme (e.g., RSA) and issues $E'(PK_{RP_k}, DM_j)$ to RP with its signature $Sign_{TAP} = Sign(SK_{TAP}, E'(PK_{RP_k}, DM_j))$.

4) DATA ACCESS

After receiving $E'(PK_{RP_k}, DM_j)$, RP_k can decrypt it with its own secret key SK_{RP_k} to get of DM_j.

5) DATA AUDITING

RP can request TAP to audit the correctness of data processing and computation by providing C_j, the hash code of DM_j, $h\left(DM_j\right)$, and $Sign_{TAP}$. Note that the auditing request should be signed by RP to ensure non-repudiation. Thereby, the auditing request AR_k contains $AR_k = \{C_j, h\left(DM_j\right), Sign_{TAP}, Sign(SK_{RP_k}, \{C_j, h\left(DM_j\right), Sign_{TAP}\})\}$. In this case, TAP handles it by querying CSP to get F_j and all $E(PK_H, D_{i,j})$ used for generating $E(PK_H, DM_j)$. TAP decrypts $E(PK_H, D_{i,j})$ to get all $D_{i,j}$ and input them into F_j to get plain DM_j, that is $DM_j = F_j(\{D_{i,j}\})(i = 1, \ldots, I)$. TAP further compares the hash code of DM_j output from F_j and the one provided by RP in order to judge if the data computation and processing at CSP is correct.

Comparing with the original Protocol 1, Protocol 2 performs re-encryption at TAP to convert the encrypted version of DM_j from $E\left(PK_H, DM_j\right)$ to $E'\left(PK_{RP_k}, DM_j\right)$, which can be decrypted by corresponding RP_k directly. RP_k will never have the chance to get access to the homomorphic secret key SK_H. Therefore, this protocol can reduce the risk caused by the CSP-RP collusion attack.

E. FURTHER OPTIMIZATION

In this section, we show a potential problem of Protocol 2 and then propose two optimized protocols to overcome it. As specified in our design goals, data should be processed and computed in a confidential measure at CSP and only eligible parties can access the processing results. TAP is responsible for issuing access rights and auditing. Normally, we do not expect TAP to know DM_j during data request and access. In Protocol 2, TAP gets to know the data processing result DM_j before auditing during the procedure of re-encryption. This could cause some problem since TAP may not be an eligible party to access DM_j even though it is fully trusted for auditing when RP delegate it for this purpose.

In order to avoid this problem, we further proposed two optimized protocols. In these two protocols, we let CSP fuzzifies DM_j by selecting a random denoted ra as a mask, computing $E(PK_H, DM_j^* ra)$, and then using different methods to remove ra and finally get DM_j. The detailed protocols are described below.

Protocol 3:

1) DATA PROVISION

This step is the same as the data provision described in Fig. 3.

2) PRIVACY PRESERVING DATA COMPUTING

CSP processes data, it selects algorithm F_j based onC_j to process the collected encrypted data $E(PK_H, D_{i,j})$ in context C_j and gains the encrypted form of data processing result $E(PK_H, DM_j)$, that is: $E(PK_H, DM_j) = F_j E(PK_H, D_{i,j})\}$, $(i = 1, \ldots, I)$. The CSP then select a random ra and compute $E(PK_H, DM_j^* ra)$.

3) RP DATA REQUEST AND AUTHORIZATION

RP_k requests CSP for the result of data processing and computation in C_j by sending a requesting message that contains $R_k = \{PK_{RP_k}, C_j\}$ and $Sign(SK_{RP_k}, R_k)$. Once receiving the request, the CSP passes the request to TAP for checking its access right. If the check based on the current access policy is positive, the TAP then requests the data processing result from CSP. After receiving the package $E\left(PK_H, DM_j^* ra\right), sign_{CSP}$, the TAP first decrypt $E\left(PK_H, DM_j^* ra\right)$ to get $DM_j^* ra$, and then re-encrypt $DM_j^* ra$ with the RP_k's public key PK_{RP_k} to get $E(PK_{RP_k}, DM_j^* ra)$ based on a homomorphic encryption scheme and issues $E(PK_{RP_k}, DM_j^* ra)$ to RP_k through CSP. Note that PK_{RP_k} used herein is a homomorphic public key of RP_k and SK_{RP_k} is a corresponding homomorphic secret key of RP_k.

4) DATA ACCESS

CSP receives $E\left(PK_{RP_k}, DM_j^* ra\right)$, which means TAP issues RP_k the right to access DM_j. CSP uses a homomorphic multiplication algorithm to erase ra from $E\left(PK_{RP_k}, DM_j^* ra\right)$:

$$E(PK_{RP_k}, DM_j^* ra)^* E(PK_{RP_k}, 1/ra) = E(PK_{RP_k}, DM_j). \quad (3)$$

And then it issues $E(PK_{RP_k}, DM_j)$ and $Sign_{CSP} = Sign(SK_{CSP}, E(PK_{RP_k}, DM_j), C_j)$ to RP_k. The RP_k can decrypt it with its own secret key SK_{RP_k} to get the plaintext of DM_j.

5) DATA AUDITING

RP may not trust the processing result of CSP. In this case, it requests TAP to audit the correctness of data processing and computation by providing C_j, the hash code of DM_j, $h\left(DM_j\right)$, the signature of CSP data provision, i.e., $Sign_{CSP} = Sign(SK_{CSP}, E(PK_{RP_k}, DM_j), C_j)$. Note that the auditing request should be signed by RP to ensure non-repudiation. Thereby, the auditing request AR_k contains $AR_k = \{C_j, h\left(DM_j\right), Sign_{CSP}, Sign(SK_{RP_k}, \{C_j, h\left(DM_j\right), Sign_{CSP}\})\}$. In this case, TAP handles it by querying CSP to get F_j and all $E(PK_H, D_{i,j})$ used for generating $E(PK_H, DM_j)$. TAP decrypts $E(PK_H, D_{i,j})$ to get all $D_{i,j}$ and input them into F_j to get plain DM_j, that is $DM_j = F_j(\{D_{i,j}\})$ $(i = 1, \ldots, I)$. TAP further compares the hash code of DM_j output from $\mathrm{F_j}$ and the one provided by RP in order to judge if the data computation and processing at CSP is correct.

Fig.5 shows the detailed procedures of Protocol 3. In this protocol, we let CSP fuzzify DM_j by selecting a random ra and computing $E(PK_H, DM_j^* ra)$. For defuzzifying $E(PK_{RP_k}, DM_j^* ra)$ to get $E(PK_{RP_k}, DM_j)$, we use a homomorphic multiplication algorithm to remove ra from $E\left(PK_{RP_k}, DM_j^* ra\right)$. In Protocol 3, TAP is unable to get DM_j during re-encryption, which solves the problem as mentioned above.

Protocol 4:

Homomorphic operation normally has a high computational cost. In order to achieve sound performance, we try to

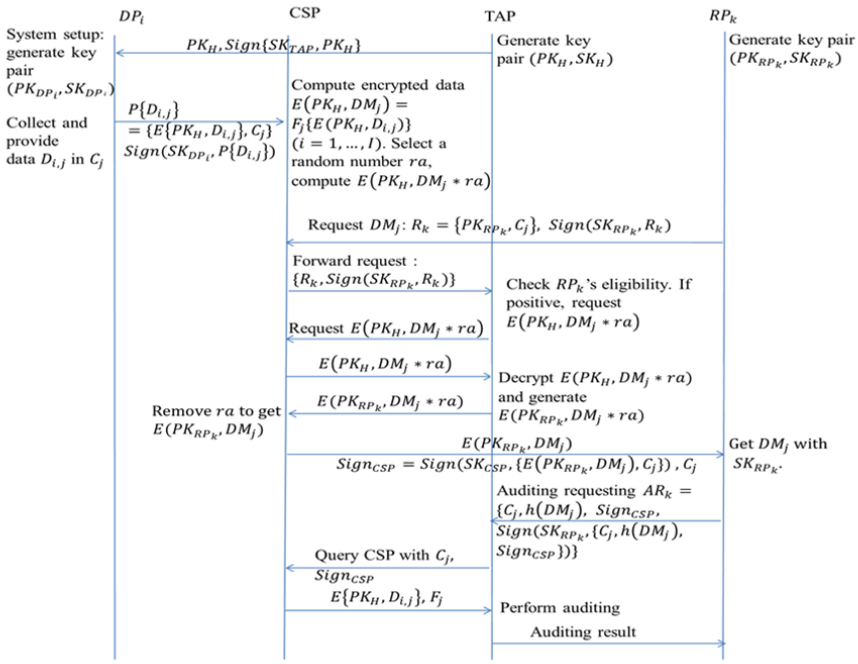

FIGURE 5. Protocol 3 - optimized verifiable computing protocol for avoiding TAP to know DM_j before auditing.

avoid it. We further propose Protocol 4 that removes the mask ra by avoiding homomorphic operation in order to improve Protocol 3.

6) DATA PROVISION

This step is the same as the design in other protocols.

7) PRIVACY PRESERVING DATA COMPUTING

This step is designed the same as that in Protocol 3.

8) RP DATA REQUEST AND AUTHORIZATION

RP_k requests CSP for the result of data processing and computation in C_j by sending a requesting message that contains $R_k = \{PK_{RP_k}, C_j\}$ and $Sign(SK_{RP_k}, R_k)$. Once receiving the request, the CSP passes the request to TAP for checking its access right. If the check based on the current access policy is positive, the TAP requests the data processing result from CSP. After receiving the package $E\left(PK_H, DM_j^* ra\right)$, $sign_{CSP}$, the TAP first decrypt $E\left(PK_H, DM_j^* ra\right)$ to get $DM_j^* ra$, and then re-encrypt $DM_j^* ra$ with the RP_k's public key PK_{RP_k} to get $E'(PK_{RP_k}, DM_j^* ra)$ based on a public key encryption scheme (e.g., RSA) and issues$E'(PK_{RP_k}, DM_j^* ra)$ to $\mathrm{RP_k}$ through CSP. Note that PK_{RP_k} used herein is a public key of RP_k and SK_{RP_k} is a corresponding secret key of RP_k.

9) DATA ACCESS

CSP receives $E'\left(PK_{RP_k}, DM_j^* ra\right)$, which means TAP issues RP_k the right to access DM_j. It then computes $E'\left(PK_{RP_k}, ra\right)$, and delivers the data package $E'\left(PK_{RP_k}, DM_j^* ra\right)$, $E'\left(PK_{RP_k}, ra\right)$, $Sign_{CSP} = Sign(SK_{CSP}, \{E'\left(PK_{RP_k}, DM_j^* ra\right), E'\left(PK_{RP_k}, ra\right), C_j\})$, and C_j to $\mathrm{RP_k}$. After receiving $E'(PK_{RP_k}, DM_j^* ra)$, RP_k can decrypt it with its own secret key SK_{RP_k} to get $DM_j^* ra$. While receiving $E'(PK_{RP_k}, ra)$, RP_k can decrypt it to get ra as well. Finally, RP_k compute $DM_j^* ra/ra$ to get the plaintext of DM_j.

10) DATA AUDITING

RP may not trust the processing result of CSP. In this case, it requests TAP to audit the correctness of data processing and computation by providing C_j, the hash code of DM_j, $h\left(DM_j\right)$, the signature of CSP data provision, i.e., $Sign_{CSP} = Sign(SK_{CSP}, \{E'\left(PK_{RP_k}, DM_j^* ra\right), E'\left(PK_{RP_k}, ra\right), C_j\})$. Note that the auditing request should be signed by RP to ensure non-repudiation. Thereby, the auditing request AR_k contains $AR_k = \{C_j, h\left(DM_j\right), Sign_{CSP}, Sign(SK_{RP_k}, \{C_j, h\left(DM_j\right), Sign_{CSP}\})\}$. In this case, TAP handles it by querying CSP to get F_j and all $E(PK_H, D_{i,j})$ used for generating $E(PK_H, DM_j)$. TAP decrypts $E(PK_H, D_{i,j})$ to get all $D_{i,j}$ and input them into F_j to get plain DM_j, that is $DM_j = F_j(\{D_{i,j}\})$ $(i = 1, \ldots, I)$. TAP further compares the hash code of DM_j output fromF_j and the one provided by RP in order to judge if the data computation and processing at CSP is correct.

For defuzzifying $E'(PK_{RP_k}, DM_j^* ra)$ to get $E'\left(PK_{RP_k}, DM_j\right)$, CSP shares $E'(PK_{RP_k}, ra)$ with RP_k together with $E'(PK_{RP_k}, DM_j^* ra)$. This allows RP_k to get both $DM_j^* ra$ and ra. Thus, it can get the plaintext of DM_j

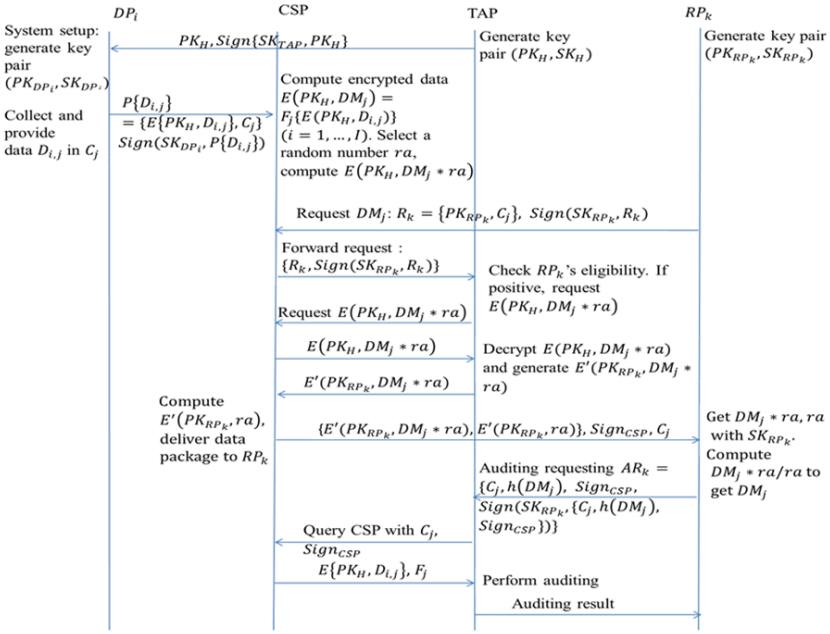

FIGURE 6. Protocol 4 - optimized verifiable computing protocol for avoiding TAP to know DM_j before auditing.

through a simple division. Fig.6 shows the detailed protocol as described above.

Although Protocols 2, 3, and 4 improve system security, they all result in extra costs on computation and/or communication. In the next section, we will analyze all the four protocols proposed above and compare them with each other to show their pros and cons in details with regard to different application scenarios and security requirements. Under the same system model and platform, we should flexibly deploy a suitable protocol in practice by balancing the requirements between security and performance.

F. JUSTIFICATION OF DESIGN

The proposed schemes were designed due to the following advantages.

Privacy preservation: the scheme ensures data mining/processing/computing privacy at CSP. CSP has no way to learn the plain data of DPs' input and their processing output. Thus, it is impossible to intrude the privacy of data and related objects. DP_i cannot know DM_j, thus one DP_i is impossible to obtain the data of other DPs. Meanwhile, only eligible RPs can access the result of data processing and computation and check its correctness if wanted. In particular, we designed Protocol 3 and Protocol 4 to conceal the data processing result from TAP in order to gain advanced security. Notably, Protocol 4 was further proposed in order to improve the efficiency of Protocol 3 in order to avoid homomorphic operations when removing the mask *ra*.

Verifiable computing: the correctness of the data processing result of CSP can be verified by TAP triggered by RP. Thus it is impossible for CSP to behave dishonestly or maliciously during data processing and computation.

Facticity enhancement: introducing TAP enhances the facticity of DP. TAP can mine the collected data of CSP and analyze if the data source has some abnormal behaviors by comparing newly generated data patterns with historical ones and analyzing collected data from different DPs.

Context awareness: the scheme supports data processing under different contexts by applying different algorithms, requesting data processing results based on context IDs, and auditing the correctness of data processing in different contexts.

Generality: the scheme supports various data processing/computing/mining cases served at the cloud. Meanwhile, it supports auditing data processing/computing/mining at a distrusted or semi-trusted CSP under any situations and contexts. Limitation could be caused by the shortcomings of FHE operations. But our scheme provides a generic framework to perform verifiable cloud computing. It is flexible to adopt the proposed four protocols to satisfy different security demands, as analyzed in Section 5.6.

V. PERFORMANCE ANALYSIS AND EVALUATION

We evaluated the performance of four protocols by analyzing and comparing their security, computational complexity, and communication cost. We then report the results of experimental performance tests on operation time and further analyze their scalability. The pros and cons of all protocols are analyzed and discussed based on rigorous comparison.

We implemented the proposed protocols in a workstation with Intel(R) CoreTM i7 4710HQ CPU and 8-GB RAM, running Ubuntu 14.04 that virtually executes the functions of DP, CSP, TAP and RP based on libraries NTL [33],

GMP [34], and FHE [35]. In our implementation, we applied BGV full homomorphic encryption [32], RSA for Public Key Cryptosystem (PKC) and SHA-1 hash function. We used a function provided by the FHE library to randomly generate plaintext with a length of 8 bytes to simulate the raw data provided by DP. Then we encrypted plaintexts into ciphertexts and conducted a number of multiplications and additions to simulate data processes at CSP. For simulating the audit process, we decrypted the ciphertext of data at TAP, calculated the result of data processing based on plaintexts, then compared its hash code with the one provided by an auditing requestor. In our implementation, RSA key size is 256 bytes. SHA-1 hash code has 20 bytes. FHE public key (PK_H) size is 180 bytes and secret key (SK_H) size is 192 bytes.

A. SECURITY ANALYSIS

The security of this scheme is ensured by FHE theory and Public Key Infrastructure (PKI) theory. In addition, we apply an auditing protocol based on TAP to ensure the correctness of data processing result. Although CSP cannot be fully trusted, we achieve security as analyzed below.

Proposition 1: It is impossible for CSP to get the raw data during data processing.

Proof: In the designed scheme, only $E(PK_H, D_{i,j})$ is provided by DPs to CSP. The FHE theory allows CSP to process the data in ciphertexts, thus no plaintext shows up at CSP during data processing. At the same time, it is hard for CSP to achieve SK_H to decrypt the ciphertext and further get $D_{i,j}$ and DM_j since TAP is a trusted party. It does not disclose SK_H and DM_j to CSP. In our scheme, data confidentiality is achieved by FHE and PKI. We apply digital signature using PKI Digital Signature algorithm (e.g., RSA) to ensure the integrity and correctness of the data processing results, achieve non-repudiation in each step of the four protocols and realize DM_j re-encryption for eligible access in Protocol 2 and Protocol 4.

Proposition 2: DM_j can only be accessed by eligible system entities.

Proof: TAP is responsible for issuing the right to access DM_j. It controls SK_H disclosure in Protocol 1 and handles DM_j re-encryption in Protocols 2-4. All above are based on the eligibility check on RP performed at TAP. Only the RP that passes this check can be issued SK_H or get re-encrypted DM_j from TAP to finally gain DM_j.

B. COMPUTATIONAL COMPLEXITY

We analyze the computational complexity of the scheme at different system entities: DP, CSP, RP, and TAP. Since generating and verifying digital signature do not cost much computation, we ignored them in the following analysis.

DP: DP is responsible for data provision by encrypting the data and signing the encrypted data package in all of the four protocols. The schemes uses full homomorphic encryption algorithm to encrypt raw data, the computational cost of data encryption depends on the size of the underlying data and it is inevitable in any cryptographic methods. According to the BGV algorithm, it requires one vector multiplication and one vector addition to complete each encryption. Thus the computation complexity of DP is O (1) in all four protocols.

CSP: In our scheme, CSP has no knowledge of stored data. Due to the variety of algorithms that could be applied in data processing, it is hard to judge the computational complexity of CSP. Herein, we only discuss the most cost-consuming operation: Multiplication. The reason is two-ciphertext multiplication takes about 10000 milliseconds, which consumes much longer time than two-ciphertext addition (4 milliseconds). Multiplication between two ciphertexts requires one vector addition and three vector multiplications. And the computation complexity of CSP relies on the number of data (N) input for processing, thus it is $O(N)$ in Protocol 1 and Protocol 2. In Protocol 3, CSP needs to add mask ra and erase it from $E\left(PK_{RP_k}, DM_j^* ra\right)$ by computing $E\left(PK_{RP_k}, DM_j^* ra\right)^* E(PK_{RP_k}, 1/ra) = E(PK_{RP_k}, DM_j)$ later on. In Protocol 4, CSP needs to add mask ra and encrypt ra with PK_{RP_k} as $E'(PK_{RP_k}, ra)$. But the above computation consumptions only perform once. Thus, the computation complexity at CSP in Protocol 3 and Protocol 4 is also O (N).

TAP: TAP is a trusted third-party responsible for checking the integrity and the correctness of data processing results in all of the four protocols. For auditing data processing in Protocol 1, it requests all the ciphertexts from CSP and decrypts them. Each decryption requires one vector multiplication and two vector modules. Then it deals with the plaintexts and does the similar computations as CSP does, which requires only one vector multiplication between two plaintexts. The computation complexity of TAP relates to the number of data, thus it is $O(N)$. In Protocol 2, one more homomorphic decryption is needed at TAP during DM_j re-encryption to get $E'(PK_{RP_k}, DM_j)$. In Protocol 3, one more homomorphic decryption and one homomorphic encryption are needed at TAP when re-encrypting $E(PK_H, DM_j^* ra)$ to get $E(PK_{RP_k}, DM_j^* ra)$. In Protocol 4, one more homomorphic decryption and one public key encryption are needed at TAP when re-encrypting $E(PK_H, DM_j^* ra)$ to get $E'(PK_{RP_k}, DM_j^* ra)$. As can be seen from the above, the computation complexity of TAP in all four protocols is $O(N)$. The expense spent in auditing is the same in the four protocols

RP: RP only needs to decrypt the result of data processing, which requires one vector multiplication and two vector modules in Protocol 1 and Protocol 3 caused by homomorphic decryption. In protocol 2 and Protocol 4, RSA decryption is performed at RP, which requires one exponential operation in Protocol 2 and two RSA decryptions in Protocol 4. Thus the computation complexity of RP is O (1) in all four protocols.

Table 2 lists the computational operations carried out by different entities and their computation complexity of the four protocols. Although the computation complexity of the four protocols is the same in each system entity, Protocol 1 outperforms at CSP and TAP. Considering the powerful computation and processing capability of CSP and TAP, the extra

TABLE 2. Computational operations and computation complexity.

Roles	Algorithms	Computational operations	Computation Complexity
DP	Encryption	Protocols 1-4: 1 vector multiplication and 1 vector addition	$\mathcal{O}(1)$
CSP	Multiplication Computation	Protocol 1 & Protocol 2: (1 vector addition and 3 vector multiplications)*N Protocol 3: (1 vector addition and 3 vector multiplications)*(N+2) Protocol 4: (1 vector addition and 3 vector multiplications)*(N+1)	$\mathcal{O}(N)$
TAP	Decryption	Protocol 1: (1 vector multiplication and 2 vector modules)*N Protocol 2: (1 vector multiplication and 2 vector modules)*(N+1) Protocol 3: (1 vector multiplication and 2 vector modules)*(N+1) + 1 vector multiplication and 1 vector addition Protocol 4: (1 vector multiplication and 2 vector modules)*(N+1)	$\mathcal{O}(N)$
	Auditing	1 vector multiplication	$\mathcal{O}(1)$
RP	Decryption	Protocol 1: 1 vector multiplication and 2 vector modules Protocol 2: 1 exponential operation Protocol 3: 1 vector multiplication and 2 vector modules Protocol 4: 2 exponential operations	$\mathcal{O}(1)$

N: the number of data input for processing.

costs introduced in Protocol 3 and Protocol 4 at CSP and TAP are not heavy. At RP, Protocol 2 and Protocol 4 outperform Protocol 1 and Protocol 3.

C. COMMUNICATION COST

The communication cost of the scheme is analyzed below. Since each DP only has to share its public key once and TAP only has to send PK_H (180 bytes) to each DP once at system initialization, these costs can be ignored.

DP to CSP: When delivered from DP to CSP, data are encrypted and signed in all four protocols. Our implementation shows that each piece of ciphertext has the length of 68 bytes if the plaintext data is 8 bytes. The size of a RSA signature is 256 bytes. Thus, the total size of communication package from DP to CSP is 68+256=324 bytes. The communication cost from DP to CSP is proportional to the number of DPs in the system.

CSP to RP: Each RP receives four pieces of data from CSP – the homomorphic ciphertext with the length 68 bytes if the plaintext data is 8 bytes, a signature with the length of 256 bytes, context identifier that has 8 bytes and $E(PK_{RP_k}, SK_H)$ that has 256 bytes (since we use RSA algorithm to encrypt SK_H with the length of 192 bytes, and the RSA encryption key we chose has the length of 256 bytes) in Protocol 1. Thus, the total communication package length from CSP to RP in Protocol 1 is 68+256+8+256=588 bytes. In Protocol 2, no message is directly sent from CSP to RP. While in Protocol 3, after the re-encryption operation, RP receives three pieces of data from CSP – the homomorphic ciphertext $E\left(PK_{RP_k}, DM_j\right)$ with the length of 68 bytes, a signature with the length of 256 bytes and context identifier that has 8 bytes. Thus, the total communication package length from CSP to RP in Protocol 3 is 68+256+8=332 bytes. In Protocol 4, CSP needs to send RP two RSA ciphertexts with the length of 512 bytes, context identifier and a signature. Thus, the total communication package length from CSP to RP in Protocol 4 is 256*2+8+256=776 bytes.

RP to TAP: TAP receives 4 pieces of data from RP for auditing in all the four protocols: context identifier that has 8 bytes, $h(DM_j)$ that has 20 bytes (SHA-1 hash function was applied), and 2 signatures each with the length of 256 bytes. Thus, the total communication cost from RP to TAP is 8+20+256*2=540 bytes.

CSP to TAP: During auditing of all four protocols, CSP needs to deliver all the ciphertexts collected from DPs to TAP, each of which has the length of 68 bytes. The communication cost is proportional to the number of ciphertexts. If there are N pieces of ciphertexts, the communication cost is 68*N bytes. For supporting RP to request a data processing result, the length of the communication package from CSP to TAP is fixed as 520 bytes in Protocol 1, 520 + the size of $E(PK_H, DM_j)$ and a signature that is 844 bytes in Protocol 2, and 520 + the size of $E(PK_H, DM_j^* ra) = 588$ bytes in Protocol 3 and Protocol 4 based on our implementation.

TAP to CSP: TAP also needs to communicate with CSP to issue $E'(PK_{RP_k}, SK_H)$ (256 bytes) in Protocol 1, provide re-encryption result $E(PK_{RP_k}, DM_j^* ra)$ (68 bytes) in Protocol 3 and $E'(PK_{RP_k}, DM_j^* ra)$ (256 bytes) in Protocol 4, respectively. Note that the sizes of encryption results could be very different due to the difference of input data sizes. The above data sizes are reported based on the settings of our implementation.

Other communication costs are introduced by queries or requests, which mainly contain some commands, keys and signatures that have fixed sizes. In total, they do not cause much communication expense. Table 3 lists the main communication costs of the proposed protocols based on our implementation. We can see that they are not high in general.

TABLE 3. Main communication costs.

Communication Channel	Communication Package Length (byte)	Communication Complexity
DP to CSP	Protocols 1-4: 68+256=324	$\mathcal{O}(1)$
RP to CSP	Protocols 1-4: 256+8+256=520	$\mathcal{O}(1)$
CSP to RP	Protocol 1: 68+256+8+256=588 Protocol 2: 0 Protocol 3: 68+256+8=332 Protocol 4: 256*2+8+256=776	$\mathcal{O}(1)$
RP to TAP	Protocol 1-4: 8+20+256∗2=540	$\mathcal{O}(1)$
CSP to TAP	Protocols 1-4: 68 ∗ N+Size(F_j) (for auditing) Protocol 1: 256+8+256=520 Protocol 2: 844 Protocol 3 & Protocol 4: 588 (for requesting one data processing result)	$\mathcal{O}$(N) $\mathcal{O}(1)$
TAP to CSP	Protocol 1: 256 (for requesting one data processing result) and 256+8=264 (for auditing) Protocol 2: 264 (for auditing) Protocol 3: 68 (for re-encryption) and 264 (for auditing) Protocol 4: 256 (for re-encryption) and 264 (for auditing)	$\mathcal{O}(1)$

N: the number of data input for processing

TABLE 4. Total communication costs of four protocols (unit: byte).

Protocol 1	Protocol 2	Protocol 3	Protocol 4
2472+68 ∗ N+Size(F_j)	2492+68 ∗ *N*+Size(F_j)	2626+68 ∗ *N*+Size(F_j)	3268+68 ∗ *N*+Size(F_j)

Table 4 lists the total communication costs of the four protocols. We observe that the communication cost of Protocol 1 is the lowest, while Protocol 4 has the highest communication cost. But the communication cost difference of the four protocols is trivial. Their costs are similar.

D. OPERATION EFFICIENCY

We mainly tested the operation time of the scheme in terms of different operations: encryption, data processing and decryption, as shown in Fig.7-9. The "data number" in Fig.7-9 refers to the number of data input by DPs. InTable 5, we report the operation time carried out by some basic operations. Compared with homomorphic operations, RSA operation time is very trivial. This result supports our analysis on computation complexity.

Encryption: Homomorphic encryption mainly takes place at DPs. Each encryption costs 1200 milliseconds. The encryption time is lineally increased with the number of data collected by DP, as shown in Fig.6. Each DP encrypts its data separately.

Data Processing: Data processing takes place at CSP and TAP. CSP deals with data in encrypted forms, while TAP in plaintext forms during auditing. Since most algorithms can be divided into a number of additions and multiplications, we only tested these two operations. As shown in Fig.8.a, the operation time of ciphertext addition almost

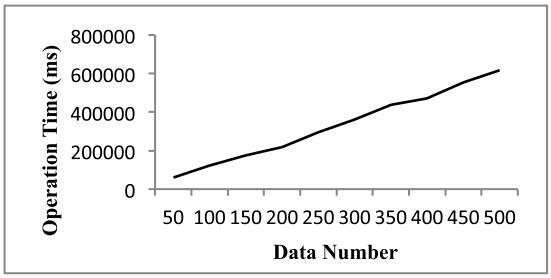

FIGURE 7. Operation time of homomorphic encryption.

proportionally increases with the number of data, which costs about 4 milliseconds (ms) to add two ciphertexts. The operation time of ciphertext multiplication proportionally increased with the number of data too, which costs about 10000 milliseconds to multiply two ciphertexts, shown in Fig.8.b. We found that main computation cost is caused by multiplication at CSP. For TAP, data in plaintexts are processed in a similar way to the computations at CSP based on the same data processing algorithms. Fig.8.c shows that the operation time of plaintext addition at TAP proportionally increases with the number of data, which costs about 8 microseconds (*us*) to add two plaintexts. The operation time of plaintext multiplications at TAP also proportionally increases with the number of data, which costs about

TABLE 5. Operation time of basic operations (unit: millisecond).

Operation Name	Homomorphic Encryption	Homomorphic Addition	Homomorphic Multiplication	Homomorphic Decryption	RSA Signature, decryption and encryption
Operation Time	1200	4	10000	600	0.3

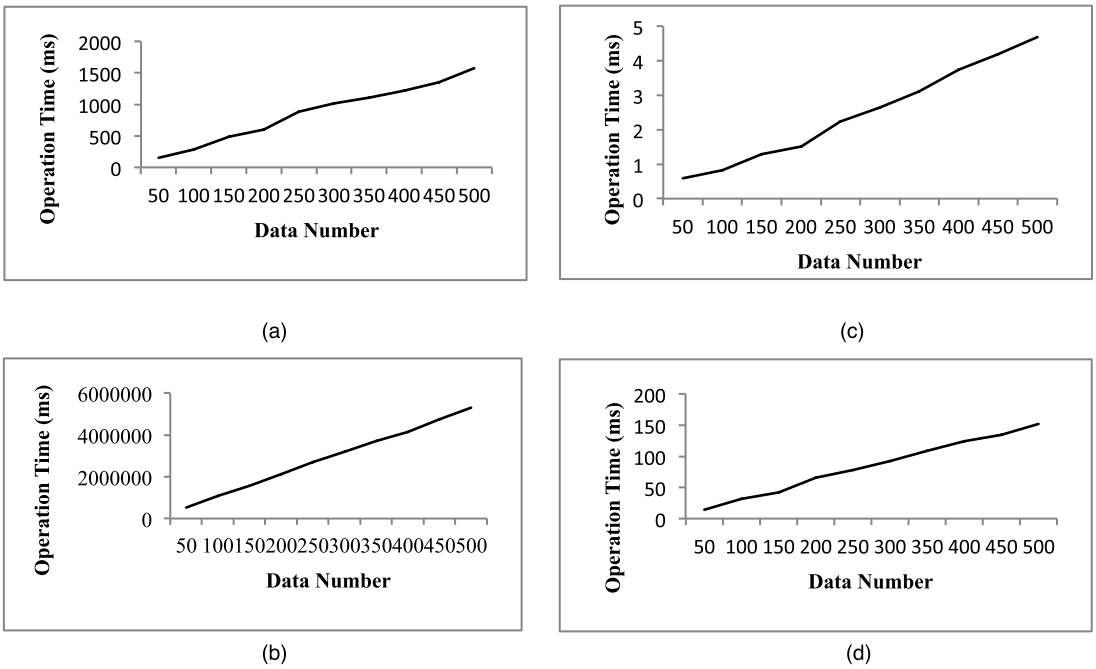

FIGURE 8. (a) Operation time of ciphertext addition; (b) Operation time of ciphertext multiplication; (c) Operation time of plaintext addition; (d) Operation time of plaintext multiplication.

300 microseconds to multiply two plaintexts, shown in Fig.8.d. We can see that multiplication operations cause the main computational cost at TAP. Based on the above tests, we found that multipli cation operation consumes much more time than addition operation at CSP and TAP.

Decryption: The homomorphic decryption takes place at RPs and TAP. The result shown in Fig.9 indicates that the operation time of decryption proportionally increases with the number of ciphertexts. RP needs to decrypt the ciphertext of data processing result to get the final plaintext result. During auditing, TAP needs to decrypt the ciphertexts of all related data. It takes about 600 milliseconds to decrypt one ciphertext to get the corresponding plaintext.

Table 6 summarizes the operation time of each entity in Protocol 1.

In protocols 2, 3, and 4, we introduce re-encryption operation to the protocols, which increase the operation time of TAP and CSP in the access of data processing result. Although TAP have to compute homomorphic decryption, another homomorphic encryption or RSA encryption, RP only has to conduct a RSA decryption in Protocol 2, a homomorphic decryption in Protocol 3 and two RSA decryptions and a simple division in Protocol 4 instead of a fully homomorphic decryption. We can see that Protocol 2 and

TABLE 6. Operation time at each system entity in protocol 1(unit: millisecond).

Operation			Time
DP	Data encryption		1200
	Signature		0.3
CSP	Data processing (ciphertext)	Single addition	4
		Single multiplication	10000
TAP	Data decryption (per data)		600
	Data processing (plaintext)	Single addition	0.008
		Single multiplication	0.3
RP	Data decryption		600

Protocol 4 improve the performance of data processing result access at RP. In Protocol 3 and Protocol 4, we further hide plain data processing result from TAP, which increases a bit computation overhead at CSP in the access of data processing result.

Table 7, Table 8 and Table 9 show the differences of operation time of the four protocols at TAP, RP and CSP in terms of data processing result access. We can see from Table 7 that, Protocol 1 is the most efficient design with regard to the computation cost at TAP, while Protocol 3 is the least efficient

TABLE 7. Operation time at TAP in the access of data processing result (unit: millisecond).

Protocol	Operation	Time
Protocol 1	1 RSA encryption	0.3
Protocol 2	1 homomorphic decryption + 1 RSA encryption	600.3
Protocol 3	1 homomorphic decryption + 1 homomorphic encryption	1800
Protocol 4	1 homomorphic decryption + 1 RSA encryption	600.3

TABLE 8. Operation time at RP in the access of data processing result (unit: millisecond).

Protocol	Operation	Time
Protocol 1	1 homomorphic decryption	600
Protocol 2	1 RSA decryption	0.3
Protocol 3	1 homomorphic decryption	600
Protocol 4	2 RSA encryption	0.6

TABLE 9. Operation time at CSP in the access of data processing result (unit: millisecond).

Protocol	Operation	Time
Protocol 1	None	None
Protocol 2	None	None
Protocol 3	1 homomorphic encryption and 1 homomorphic multiplication	11200
Protocol 4	1 RSA encryption	0.3

and Protocol 2 and Protocol 4 sit in the middle. From Table 8, we can conclude that the computation workload is not heavy in each protocol. But Protocol 2 and Protocol 4 work very efficiently at RP. With regard to the computation cost at CSP, Protocol 3 is the least efficient, while Protocols 1, 2 and 4 have zero or a bit computation overhead.

E. SCALABILITY

In our scheme, most of the computations are carried out by CSP and TAP, which are assumed to have sufficient capacity and capability. DPs and RPs only perform data encryption and decryption. They do not deal with complicated algorithms for data processing. We analyze system scalability from the view of different system entities as below.

DPs: In our scheme, DPs only need to conduct homomorphic encryptions that take about 1200ms on each data in our test. Each signature generation only takes 0.3ms, which can be ignored. Notably, the number of DPs does not affect the above operation time. This means that no matter how many DPs send their data to CSP, the time spent at each DP is fixed.

CSP: CSP takes responsibility to process ciphertexts. Since the ciphertext is the data encrypted by a full homomorphic encryption algorithm, the computation of the ciphertext could be very complicated. It is indicated that this cost is proportional to the number of data uploaded to CSP regarding a specific context that costs about 4ms for one addition and 10000ms for one multiplication. Assumed that the processing capability at CSP is powerful, the scheme design shows potential for big data processing, especially for the data processing that only contains a huge number of addition operations and has limited number of multiplication operations. Considering the cost introduced by data access control on processing result, we can see from Table 9 that, Protocol 1 and Protocol 2 can perfectly support the data access requests from a huge number of RPs (i.e., scalability). Protocol 4 can also support scalability very well. But Protocol 3 performs the worst with regard to scalability.

TAP: TAP is responsible for auditing that consists of ciphertext decryption and plaintext processing. Our testing result shows that it takes 600ms to perform one decryption, much longer than the processing time of plaintexts. As TAP has to decrypt each piece of ciphertext sent by CSP, this time consumption is proportional to the number of ciphertexts. Considering that the processing capability of TAP is also powerful, the scheme design is suitable for such data processing. Considering the cost introduced by data access control on processing result, we can see from Table 7 that Protocol 1 performs the best, Protocol 3 performs the worst, while the performance of Protocol 2 and Protocol 4 is in the middle.

RPs: RP takes about 600ms for each homomorphic decryption and 0.3ms for each RSA decryption. They are not affected by the number of DPs. All of the four protocols support scalability, in which Protocol 2 and Protocol 4 perform better than Protocol 1 and Protocol 3.

F. PROTOCOL COMPARISON

In this section, we summarize the above performance analysis and comprehensively compare the four protocols in term of security, computational overhead, communication cost and scalability. Based on the comparison, we further discuss the pros and cons of each protocol, and comment their applicability. The comparison result is shown in Table 10.

Regarding security, Protocol 1 suffers from the CSP-RP collusion attack, while other protocols can resist it. But Protocol 2 discloses DM_j to TAP before auditing, which is not expected in many situations. Protocol 3 and Protocol 4 overcome this problem, thus they can achieve a high level of security. In terms of computational overhead, the four protocols hold the same computational complexity. But considering the concrete computation operations, the computational overhead of Protocol 1 is the lowest, Protocol 2 and Protocol 4 rank in the middle, while Protocol 3 performs the highest. The communication costs of the four protocols are quite similar, in which Protocol 1 is the lowest, Protocol 4 is the highest, and other two protocols are in middle levels. Talking about scalability, Protocol 1 is the best and Protocol 3 is the worst, while Protocol 2 and Protocol 4 perform in the middle. In Table 10, we assign the points to each protocol based on their performance (the best: *** points; the

TABLE 10. Protocol Comparison (unit: millisecond).

Protocol	Security	Computational Overhead	Communication Cost	Scalability	Pros and Cons, Applicability
Protocol 1	Low (*)	Low (***)	Low (***)	Best (***)	Security level is low, but perform efficiently with low computation and communication costs, and good scalability (10 points); it can be applied in the case that security requirement is not high
Protocol 2	Middle (**)	Middle (**)	Middle- (**)	Middle (**)	Security level is not high, but perform well in a middle level regarding computation and communication costs, and scalability (8 points); it can be applied in the case that the requirements on security, efficiency and scalability are not high
Protocol 3	High (***)	High (*)	Middle+ (**)	Worst (*)	Security level is high, but perform the worst with high computational overhead and low scalability (7 points); it can be applied in the case that the requirements on security and communication cost are high
Protocol 4	High (***)	Middle (**)	High (but also very good) (**)	Middle (**)	Security level is high, but perform well in the middle level with reasonable computational overhead and communication cost, and sound scalability (9 points); it can be applied in the case that the requirements on security, efficiency and scalability are high

Notes: *** - the best; * - the worst; ** - the middle performance.

worst: * points; and the middle: ** points). Through comparison, we can see that Protocol 1 can be applied into the situation when security requirement is not high since it can achieve good performance with regard to computation, communication and scalability. In the case that the security requirement is high, we recommend to adopt Protocol 4 since it ensures high security and at the same time offers sound performance on efficiency and scalability.

VI. CONCLUSION

In this paper, we proposed an effective context-aware verifiable computing scheme with four auditing protocols for enhancing the trust of cloud computing. By introducing the TAP and applying full homomorphic encryption, we audit the correctness of data processing at cloud even though data computation is conducted in an encrypted form. The proposed auditing protocols can help a requesting party to check the integrity and correctness of the data processing conducted at the CSP. Based on full homomorphic encryption, it is hard for the cloud to get the raw data that DPs do not want to expose. In addition, based on digital signature, other security properties such as integrity and non-repudiation can be ensured. The proposed scheme can serve as a generic framework to support verifiable computing based on different data processing algorithms for the cloud in various contexts. Four optional auditing protocols were designed to satisfy different security requirements. Their performance was evaluated and compared through serious analysis with regard to security, computational overhead, communication cost and scalability. The evaluation results show the effectiveness and efficiency of our designs. Through rigorous comparison, we further summarize the pros and cons of each protocol and comment their applicability.

Regarding future work, we will further improve the scheme in case that TAP cannot be fully trusted to obtain raw data from DPs. How to perform cloud computing auditing in an encrypted form is an interesting and challenging research topic worth our further investigation.

REFERENCES

[1] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[2] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[3] D. K. Mishra and M. Chandwani, "Anonymity enabled secure multi-party computation for Indian BPO," in *Proc. IEEE Region 10 Conf. (TENCON)*, Oct./Nov. 2007, pp. 1–4.

[4] W. Liu, S.-S. Luo, Y.-B. Wang, and Z.-T. Jiang, "A protocol of secure multi-party multi-data ranking and its application in privacy preserving sequential pattern mining," in *Proc. 4th Int. Joint Conf. Comput. Sci. Optim. (CSO)*, Apr. 2011, pp. 272–275.

[5] Y. Zhu, L. Huang, W. Yang, D. Li, Y. Luo, and F. Dong, "Three new approaches to privacy-preserving add to multiply protocol and its application," in *Proc. 2nd Int. Workshop Knowl. Discovery Data Mining (WKDD)*, 2009, pp. 554–558.

[6] T. Wang and W. Luo, "Design and analysis of private-preserving dot product protocol," in *Proc. Int. Conf. Electron. Comput. Technol.*, Feb. 2009, pp. 531–535.

[7] Y. Shen, J. Han, and H. Shan, "The research of privacy-preserving clustering algorithm," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Inform. (IITSI)*, 2010, pp. 324–327.

[8] M.-C. Liu and N. Zhang, "A solution to privacy-preserving two-party sign test on vertically partitioned data ($P^2$2NSTv) using data disguising techniques," in *Proc. Int. Conf. Netw. Inf. Technol. (ICNIT)*, 2010, pp. 526–534.

[9] J. Zhan, S. Matwin, and L. Chang, "Privacy-preserving collaborative association rule mining," *J. Netw. Comput. Appl.*, vol. 30, no. 3, pp. 1216–1227, 2007.

[10] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1026–1037, Sep. 2004.

[11] F. Zhang and G. Zhao, "A more well-founded security proof of the privacy-preserving distributed mining of association rules protocols," in *Proc. 1st Int. Workshop Model Driven Service Eng. Data Quality Secur.*, 2009, pp. 25–28.
[12] P. Wang, "Research on privacy preserving association rule mining a survey," in *Proc. 2nd IEEE Int. Conf. Inf. Manage. Eng. (ICIME)*, Apr. 2010, pp. 194–198.
[13] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter, "Privacy preserving regression modelling via distributed computation," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 677–682.
[14] J. Liu, J. Z. Huang, J. Luo, and L. Xiong, "Privacy preserving distributed DBSCAN clustering," in *Proc. Joint EDBT/ICDT Workshops*, 2012, pp. 177–185.
[15] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.
[16] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee, "Privacy-preservation for gradient descent methods," in *Proc. 13th ACM Int. Conf. Knowl. Discovery Data Dining SIGKDD*, 2007, pp. 775–783.
[17] A. Amirbekyan and V. Estivill-Castro, "Practical protocol for Yao's millionaires problem enables secure multi-party computation of metrics and efficient privacy-preserving k-NN for large data sets," *Knowl. Inf. Syst.*, vol. 21, no. 3, pp. 327–363, 2009.
[18] F. Herrmann, D. Khadraoui, and Y. Lanuel, "Secure multi-party computation problem for distributed electronic contract management," in *Proc. Inf. Commun. Technol.*, 2006, pp. 274–279.
[19] C. Thoma, T. Cui, and F. Franchetti, "Secure multiparty computation based privacy preserving smart metering system," in *Proc. North Amer. Power Symp. (NAPS)*, 2012, pp. 1–6.
[20] P. Jangde and D. K. Mishra, "A secure multiparty computation solution to healthcare frauds and abuses," in *Proc. 2nd Int. Conf. Intell. Syst., Modelling Simulation (ISMS)*, 2011, pp. 139–142.
[21] R. Krishnan and R. Sundaram, "Policy-agile encrypted networks via secure function computation," in *Proc. Military Commun. Conf. (MILCOM)*, 2010, pp. 954–959.
[22] Y. Luo, L. Shi, C. Zhang, and J. Zhang, "Privacy-preserving protocols for string matching," in *Proc. 4th Int. Conf. Netw. Syst. Secur. (NSS)*, 2010, pp. 481–485.
[23] Z. Erkin, T. Veugen, and R. L. Lagendijk, "Generating private recommendations in a social trust network," in *Proc. Int. Conf. Comput. Aspects Social Netw. (CASoN)*, 2011, pp. 82–87.
[24] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 2024–2033, May 2013.
[25] K. Frikken, M. Atallah, and C. Zhang, "Privacy-preserving credit checking," in *Proc. 6th ACM Conf. Electron. Commerce*, 2005, pp. 147–154.
[26] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2004, pp. 103–114.
[27] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 338–347.
[28] M. Huang, B. Lin, and Y. Yang, "Privacy-preserving path-inclusion protocol through oblivious automata," in *Proc. IEEE Int. Conf. Intell. Control, Autom. Detection High-End Equip. (ICADE)*, Jul. 2012, pp. 128–132.
[29] F. Kerschbaum and R. J. Deitos, "Security against the business partner," in *Proc. ACM Workshop Secure Web Services*, 2008, pp. 1–10.
[30] L. Pang, M.-H. Sun, S.-S. Luo, B. Wang, and Y. Xin, "Full privacy preserving electronic voting scheme," *J. China Univ. Posts Telecommun.*, vol. 19, no. 4, pp. 86–93, Aug. 2012.
[31] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.
[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. ACM 3rd Innov. Theor. Comput. Sci. Conf.*, 2012, pp. 1–25.
[33] *NTL*, accessed on Feb. 1, 2017. [Online]. Available: https://github.com/PlanetaryResources/NTL-Asteroid-Data-Hunter
[34] *GMP*, acceesed on Feb. 1, 2017. [Online]. Available: https://github.com/Rupan/gmp
[35] *FHE*, acceesed on Feb. 1, 2017. [Online]. Available: https://github.com/rdancer/fhe
[36] L. Gao, Z. Yan, and L. T. Yang, "Game theoretical analysis on acceptance of a cloud data access control system based on reputation," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2632110.
[37] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and IoT: A big picture," *Future Generat. Comput. Syst.*, vol. 49, pp. 58–67, Aug. 2015.
[38] C. Yang, C. Liu, X. Zhang, S. Nepal, and J. Chen, "A time efficient approach for detecting errors in big sensor data on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 329–339, Feb. 2015.
[39] X. Zhang, L. T. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using MapReduce on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 363–373, Feb. 2014.
[40] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: Top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2609–2622, Sep. 2015.
[41] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generat. Comput. Syst.*, vol. 62, pp. 175–189, Sep. 2016.

ZHENG YAN (M'06–SM'14) is currently a Professor with Xidian University, Xi'an, China, and a Visiting Professor with Aalto University, Espoo, Finland. She has authored over 150 peer-reviewed publications and two books. She is the Inventor and Co-inventor of 50 patents and patent applications. Her research interests are in trust, security and privacy. She serves as an Associate Editor of the *Information Sciences*, the IEEE INTERNET OF THINGS JOURNAL, the *Information Fusion*, the JNCA, the IEEE ACCESS, and *Security and Communication Networks* and nine journals. She also serves as an organization and program committee member for numerous international conferences and workshops.

XIXUN YU received the B.Eng. degree in telecommunications engineering from Xidian University, Xi'an, China, in 2015. He is currently pursuing the Ph.D. degree in information security with the School of Cyber Engineering, Xidian University. His research interests are in cloud security and verifiable computing.

WENXIU DING received the B.Eng. degree in information security from Xidian University, Xi'an, China, in 2012. She is currently pursuing the Ph.D. degree in information security with the School of Cyber Engineering, Xidian University. She was a Research Assistant with the School of Information Systems, Singapore Management University, from 2015 to 2016. Her research interests include RFID authentication, privacy preservation, data mining, and trust management.