Yu, Xixun; Yan, Zheng; Zhang, Rui

Verifiable outsourced computation over encrypted data

Published in:
Information Sciences

DOI:
10.1016/j.ins.2018.12.022

Published: 01/04/2019

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

Verifiable Outsourced Computation over Encrypted Data

Xixun Yu[a], Zheng Yan[b,a,*], Rui Zhang[c]

[a]*State Key Lab on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an, 710071, China*

[b]*Department of Communications and Networking, Aalto University, Espoo, 02150, Finland*

[c]*Department of Computer and Information Sciences, University of Delaware, Newark, 19716, DE, USA*

Abstract

In recent years, cloud computing has become the most popular and promising service platform. A cloud user can outsource its heavy computation overhead to a cloud service provider (CSP) and let the CSP make the computation instead. In order to guarantee the correctness of the outsourced processing (e.g., machine learning and data mining), a proof should be provided by the CSP in order to make sure that the processing is carried out properly. On the other hand, from the security and privacy points of view, users will always encrypt their sensitive data first before they are outsourced to the CSP rather than sending the raw data directly. However, processing and verifying of encrypted data computation has always been a challenging problem. Homomorphic Encryption (HE) has been proposed to tackle this task on computations over encrypted data and ensure the confidentiality of the data. However, original HE cannot provide an efficient approach to verify the correctness of computation over encrypted data that is processed by CSP. In this paper, we propose a verifiable outsourced computation scheme over encrypted data with the help of fully homomorphic encryption and polynomial factorization algorithm. Our scheme protects user data security in outsourced processing and allows public verification on the computation result processed by CSP with zero knowledge. We then prove the security of our scheme and analyze its performance by comparing it with some latest related works. Performances analysis shows that our scheme reduces the overload of both the cloud users and the verifier.

Keywords: Outsourced computation, Verifiability, Privacy preservation, Fully homomorphic encryption, Polynomial factorization algorithm

1. Introduction

Recent years have witnessed the growing adoption of cloud computing, which allows cloud users to remotely store and process their data in the cloud in order to reduce local storage and computational overhead [28, 29]. Meanwhile, it has become increasingly possible

*Corresponding author.

Email addresses: gabrielyu@126.com (Xixun Yu), zyan@xidian.edu.cn; zheng.yan@aalto.fi (Zheng Yan), ruizhang@udel.edu (Rui Zhang)

September 28, 2018

for sensitive data of the cloud users to be exposed to the cloud service provider (CSP) or the attacker. Protecting the security and privacy of the users' data normally requires the data be encrypted before being outsourced to the CSP [43–46, 48]. Nevertheless, encrypted data makes their computation and processing in the cloud extremely difficult. This fact retards the cloud computing to be widely applied into many applications, such as machine learning and data mining that request privacy protection. Homomorphic Encryption (HE) mechanism was proposed to solve this problem [18, 19]. Schemes (such as RSA, ElGamal, Paillier, etc.) support either additive or multiplicative homomorphism, which can be used to realize cloud computing in certain scenarios for some specific mathematical operations. However, in most general scenarios, computations that are outsourced to the cloud consists of multiple mathematical operations instead of specific single one. In order to carry out these computations in an encrypted form, the algorithms in the cloud should be fully homomorphism, while many traditional encryption algorithms cannot support this property. Fully Homomorphic Encryption (FHE) [24] provides a meaningful approach to this problem. It enables to carry out meaningful computations on polynomials in the encrypted form. Although FHE seems theoretically perfect, it requests huge storage consumption and heavy computation overhead and is thus not very efficient to be applied in many power-limited devices.

On the other hand, cloud users also have serious concerns about the correctness of the data computation results [13, 27, 47]. To address such concerns, the users desire the CSP to provide a commitment for its computation result whereby the users can verify the correctness of the result. In the area of verifiable computation, several solutions such as [5, 6, 14, 25, 30, 34, 35, 37, 38, 42, 49] have been proposed to tackle this problem. In these schemes, a client user outsources his computation task to a worker in the cloud, which in turn carries out the computations and generates a commitment to the computation result at the same time. The users can then verify the result using the commitment to verify whether the worker processes the computation properly. Unfortunately, all these schemes require the user's private information (such as secret keys, etc.) to perform the verification, which means that only the user himself can verify the result. In order to improve the efficiency and transparency of verification, it is preferred that the users can also outsource the verification task to a trusted third party or the public. In this situation, considering the security of users' data, it requires that the commitment should not contain any additional information other than what is required to verify the computation result. Along with this direction, Parno et al. [38] propose a scheme to realize public verification by using Quadratic arithmetic program and Elliptic curve encryption. Their scheme has the advantage that the commitment is of constant size no matter how many computations are executed, but unfortunately cannot guarantee the security of user data.

Although various efforts have been made to tackle the above two challenges, they suffer from some inherent limitations. First, most of the existing works are designed for specific frameworks and cannot be used in other scenarios. It means that even a small change in the framework may cause a failure to the schemes owing to their specific designs. Second, most of the works cannot support public verification, as they always require some users' private information to carry out the verification. Revealing such private information makes

it possible for the attackers to launch various attacks on the users to intrude their privacy.

In order to securely execute the computations in the cloud in a general scenario and guarantee the correctness of cloud computing through public verification by preserving cloud user data privacy, we propose a verifiable outsourced computation scheme over encrypted data by utilizing a Trusted Authenticator (TA) and a Public Auditor Proxy (PAP). In our scheme, the outsourced data are first encrypted using a fully homomorphic encryption scheme and then sent to the CSP for further computation. Thanks to the participant of TA, the computation result is re-encrypted before it is sent to the requesting party (RP) to reduce the computation overhead of the RP. During the audit of computation result, the PAP can verify the correctness of the computation and issue the audit result to the RP by cooperating with the TA without disclosing the plaintext of the outsourced data. Specifically, our contributions in this paper can be summarized as follows.

- An end-to-end system for efficiently verifying computations performed by cloud service provider over encrypted data. It includes a computation procedure that performs the computations in ciphertexts and a verification procedure that first transforms the outsourced function into a verification function and then let the public verify the computations by this function. Moreover, The privacy of the cloud users' data is preserved against the public verifiers in our scheme.
- We formally prove the security of our scheme and evaluate the performance of our scheme and show its efficiency and feasibility in cloud computing. The verification overhead of our system mainly depends on the number of variables in the computation function.

The rest of the paper is organized as below. In Section 2, we briefly discuss the related works in the research field of verifiable computation. In Section 3, we introduce the preliminaries of our scheme. In Section 4, we formulate the problem and introduce our design goals, followed by the detailed design of our proposed scheme. We present our proposed scheme in detail in Section 5. In Section 6, we perform a comprehensive analysis and conduct performance evaluation. Finally, we conclude our paper and suggest future work in the last section.

2. Related Works

In this section, we discuss some work most germane to our work in verifiable computation and homomorphic computation.

2.1. Verifiable Computation

2.1.1. Probabilistically Checkable Proofs (PCPs)

PCP-based schemes were first proposed to solve the problem of verifiable computation [37]. PCP-based schemes allow a user to verify the computation result provided by the CSP, but the user has to store a large amount of information locally to allow subsequent verification of the computation results. Although these schemes provide theoretical

security guarantee, they incur significant computational costs at the user side, which limits their applicability in practice. The follow-up work [34, 38] improves the original PCP scheme by reducing the computational cost. Taking advantage of parallel operations, these schemes incur low computational cost for verification. However, the workload at the prover in these schemes is quadratic to the size of the initial computation owing to the usage of Hadamard PCP [34]. Since these schemes require that the computation be outsourced in batch, the verification process has to wait for all the batches to return before it can decide. The verifiers are designated in the system so that one must have a secret key to verify the result. Furthermore, these schemes cannot guarantee the correctness of computations over encrypted data.

2.1.2. Non-interactive Verifiable Computation (GGP)

Non-interactive verifiable computation is another approach for verifiable computation. It was first introduced by Gennaro *et al.* [21] to allow a weak client to outsource its computation to a powerful worker, which computes the result and generates a commitment for its correctness. With the commitment, the clients can verify the correctness of the returned result. Obviously, the schemes work on the condition that the computational cost of the verification and preparation should be lower than carrying out the computation locally. A number of works were then proposed to increase the efficiency of the scheme [6, 7, 16, 21, 26, 31, 41], both in specific scenarios and a generic scenario. However, these schemes are still not practical due to high computation complexity.

Quadratic Arithmetic Programs (QAPs) proposed by Gennaro et al. [33, 40] is an efficient approach to verify the computations processed by a third distrusted party. It can be easily used in a Ginger's cryptographic framework [38] to realize efficient verifiable computation. Taking advantage of both techniques, it is possible to adapt to the real world. However, it still has the security concern that the verifier has to know the secret key to conduct verification, which limits the usage of the scheme due to privacy intrusion.

Table 1: Scheme comparison

Schemes	PCPs[13]	GGP[12]	FHE[2]	Our scheme
computation confidentiality	×	×	√	√
Public verifiability	√	√	×	√

Note: ×- not supported; √ - supported.

2.2. Homomorphic Computation

2.2.1. Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) [24] is a newly proposed cryptography algorithm that can increase the security of outsourced computation. Its concept is to directly perform computations on ciphertext inputs, which can compute out the encrypted version of the result computed on the same plaintext input corresponding to a function F. Obviously, this algorithm can be used to construct privacy-preserving outsourced computation schemes and

verifiable computation schemes. While the computational complexity and storage consumption of these algorithms are heavy for it to be used in the real world at its early research stage. In order to bring this technique into practice, many schemes [6, 16, 20, 32] take place to either increase the efficiency of the original schemes or construct FHE systems based on other simpler theories. Some schemes [20] use parallel processing techniques (such as Map-reduce) to apportion the computational overhead of FHE.

2.2.2. Homomorphic Authenticator

Homomorphic authenticator [12, 16, 22] is a concurrent work in the area of verifiable computation. Instead of making calculations on initial input data. It first generates a tag on input data, and then use these tags as inputs to outsource to the cloud workers, the workers can directly compute the outsourced function using these tags, and the result tag of the computation is the one that authenticates the same computation result of the initial input data. Gennaro and Wichs [11] first introduced the concept of fully homomorphic Message authenticators (MACs) and they proposed a scheme to realize it. However, the scheme is not secure if an adversary asks to query the verification. Catalano and Fiore [10] then proposed another homomorphic MAC scheme to defend this kind of attacks, but their scheme can only be used in some specific classes of computations.

Concepts and schemes such as multi-function verifiable computation [6, 15, 39] and multi-client verifiable computation [23, 25] were proposed as follow-up works in this area. They further increase the utility of verifiable computation in the real world. However, most of these studies still cannot guarantee the security of the system if there happens to be collusion between malicious servers and clients.

Notably, there are no existing schemes that can generically support both computation security and public verification in this research field. As shown in Table 1, we notice that PCPs and GGP only support verification, while FHE only supports outsourced computation security. Although FHE has the potential to realize the expected properties, such as generality and publicity. It seems inflexible and ineffective to reuse the scheme again as a manner of verification.

3. Preliminaries

3.1. Fully Homomorphic Encryption

The notion of fully homomorphic encryption was first introduced by Rivest in 1978 [36]. However, due to the technical limitation of that time, a promising scheme on fully homomorphic encryption had not been proposed until 2009 by Gentry [24]. He first proposed a scheme called somewhat homomorphic encryption (SHE) using ideal lattices to realize homomorphism that is able to compute only a limited depth of circuits. Then he and his team improved this work and successfully constructed a fully homomorphic encryption scheme (FHE) based on ideal lattices [23]. After Gentry first brought this theory in the real world, fully homomorphic encryption became a hot topic in the research area again. Among all of them, there are three main branches: fully homomorphic encryption based

on ideal lattices [23], fully homomorphic encryption based on integers [17], and fully homomorphic encryption based on Learning with Errors (LWE) or Ring Learning with Errors (RLWE) [9]. In this paper, we use BGV fully homomorphic encryption scheme [8], which is based on RLWE to conduct our simulation.

In this section, we briefly introduce the algorithms of fully homomorphic encryption that are used in the system we proposed below. There are four algorithms in fully homomorphic encryption scheme: Key Generate (Keygen), Encrypt (Enc), Decrypt (Dec), and Evaluate (Eval). The details are given below:

- $\mathsf{Keygen}(1^\lambda) \rightarrow (pk, sk)$: Given security parameter λ, the algorithm outputs a fully homomorphic public and private key pair pk and sk.
- $\mathsf{Enc}(pk, m_i) \rightarrow \varphi_i$: Given public key pk and plaintext m_i, the algorithm encrypts m and outputs a ciphertext φ_i, where m_i denotes the ith plaintext provided by the ith data provider (DP) DP_i and φ_i denotes the corresponding ciphertext of m_i.
- $\mathsf{Dec}(pk, \varphi_i) \rightarrow m_i$: Given secret key sk and the ciphertext φ_i, the algorithm decrypts φ_i and outputs a plaintext m_i.
- $\mathsf{Eval}(pk, C, \Phi) \rightarrow \varphi$: Given public key pk, evaluated circuit C, and a tuple of ciphertext $\Phi = \langle \varphi_1, \ldots, \varphi_i \rangle)$, the algorithm outputs result φ which denotes a ciphertext computation result, where $C(m_1, \ldots, m_i) = Dec(sk, \varphi)$.

The key algorithm of fully homomorphic encryption is *Eval* which computes the data in a ciphered form. Obviously, the encryption of data protects them from composed to unexpected users. Only the party who owns the secret key sk can get access to the plaintext result.

3.2. Polynomial Factorization

We now briefly introduce the background of multivariate polynomials factorization which our scheme relies on.

3.2.1. Multivariate Polynomials

We use a multiset where an element can take place more than once to present a multivariate polynomial. For example, $\{1, 1, 2, 2, 3, 3\}$ is a multiset. Formally, a function $S \rightarrow Z^{\geq 0}$ denotes a map between a multiset S and the multiplicity of each element in that multiset, i.e., $S = \{1, 1, 2, 3, 3, 3\}$, we have $S(1) = 2, S(2) = 1, S(3) = 3$. $|S|$ denotes the degree of a multiset, i.e. for a multiset $\{1, 1, 2, 3, 3, 3\}$, we have $|S| = 6$. Finally, $S_{d,n}$ denotes a family of multisets which have the size of at most d and different elements of at most n. Let f be an n-variable polynomial over , then f can be represented as $f(x) = f(x_1, x_2, \ldots, x_n) \in Z_{p[x]} = \sum_{S \in S_{d,n}} c_S \prod_{i \in S} x_i^{S(x_i)}$ where c_S denotes the coefficient of the corresponding monomial multivariate polynomials. For example, the multiset $\{1, 1, 2, 5, 5, 5\}$ is corresponding to the n-variable polynomials $x_1^2 x_2 x_5^3$.

Especially, as to a multivariate polynomial, we use the maximum degree of the monomial contains in the polynomial to denote the degree of that polynomial, e.g., $4x_1x_2 + 3x_1^3x_2^2x_1$ has the degree of 6.

3.2.2. Multivariate Polynomials Factorization

We have the following theorem regarding multivariate polynomials factorization.

Theorem 1. *Let* $f(x) = f(x_1, x_2, \ldots, x_n) \in Z_p^n[x]$ *be a* n*-variable polynomial. For all* $a \in Z_p^n$ *there exists* $q_i(x) \in Z_p^n[x]$ *where* $f(x) - f(a)$ *can be represented as* $f(x) - f(a) = \sum_{i=1}^{n}(x_i - a_i)q_i(x)$*. Furthermore, there exists a polynomial-time algorithm to find these* $q_i(x)$*.*

PROOF. The proof of this factorization algorithm is straightforward. Given a n-variable polynomial $f(x) - f(a)$ over, we use $x_1 - a_1$ to divide this polynomial to get

$$f(x) - f(a) = (x_1 - a_1)q_i(x_1, x_2, \ldots, x_n) + r_1(x_2, x_3, \ldots, x_n), \tag{1}$$

where $r_1(x_2, x_3, \ldots, x_n)$ is the remainder term that does not contain variable x_1. Continuously, divide the remainder with $(x_2 - a_2)$, then with $(x_3 - a_3)$, and so on. Finally, $f(x) - f(a)$ can be represented as

$$f(x) - f(a) = \sum_{i=1}^{n}(x_i - a_i)q_i(x_1, x_2, \ldots, x_n) + r_n \tag{2}$$

where $r_n \in Z_p$. Since $f(x) - f(a) = 0$ when $x = a$, r_n should be 0 as well. So, we have

$$f(x) - f(a) = \sum_{i=1}^{n}(x_i - a_i)q_i(x). \tag{3}$$

4. Problem Statement

In this section, we first introduce the system model and our design goals and then provide some notations used in our scheme.

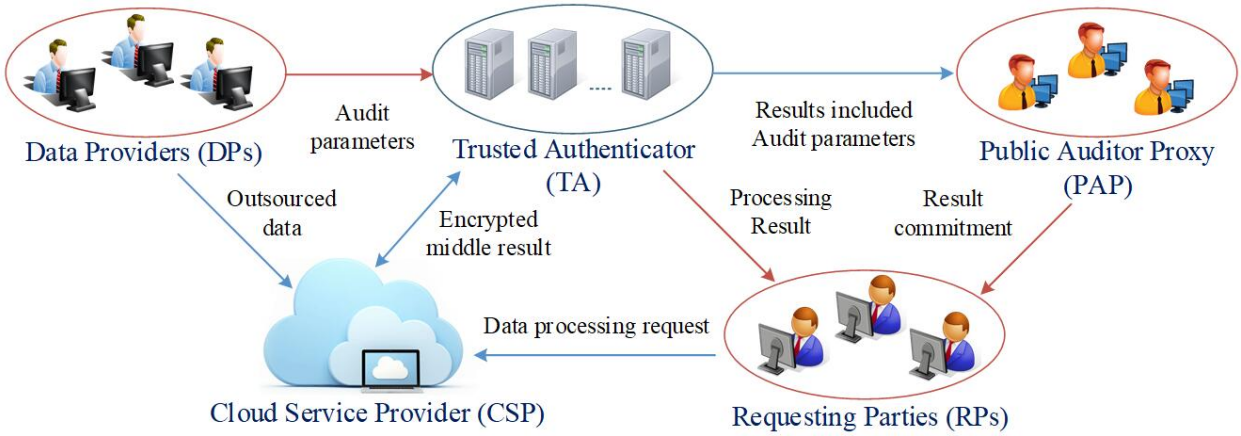

Figure 1: System model

4.1. System Model

We assume a cloud computing system comprising data providers (DPs), a cloud service provider (CSP), requesting parties (RPs), a trust authenticator (TA), and a public auditor proxy (PAP). In the system, the DP is a party who intends to outsource its data computation to the cloud. Since its data may be highly sensitive, the DP wants to keep them from being exposed to other parties. The CSP provides storage and computation services to the DPs. It is responsible for carrying out computations according to the DP's deamnd. Since CSP could be on getting to know the data of DPs and cannot be fully trusted, the DPs prefer uploading encrypted data to the cloud. The TA is introduced as a semi-trusted party which acts like an authentication center for generating system security keys for computation and checking the eligibility of RPs. The RP is a party that requests for the final computation result processed by the CSP. As the cloud may possibly forge or provide wrong computation result to the RP, we introduce the PAP as a party who helps the RP to verify the correctness of the computation result returned by the CSP. PAP is only trusted for verification, which means that no private information, i.e. plaintext input and plaintext output, should be exposed to the PAP. Fig. 1 shows the system model of our scheme.

We assume that the TA and the CSP do not collude with each other. In particular, the CSP cannot achieve the data encryption secret keys of the DPs provided by the TA. It is thus impossible for the CSP to obtain any information of the DPs' original data, except executing computations on the ciphertext. The RP can only get access to the computation result provided by CSP, even if it gets the chance to know the data input from DPs, it cannot decrypt these data to get the initial information of the computation. TA may act as a middle party between CSP and RP in re-encrypting the computation result. Since TA is semi-trusted, in this procedure, it is required that TA cannot get the plaintext computation result. Moreover, TA and PAP do not collude. As the public, PAP cannot gain the data encrypted secret keys, otherwise, it will be aware of all the plaintext input and output.

In our scheme, the data provider outsources all its data set $\{D_1, \ldots, D_i\}$, where $D_i \in Z_p$ to the cloud to calculate a polynomial function $F \in Z_p^n[x]$. With the help of the Trusted Anthenticator (TA), the scheme enables public auditor proxy to verify the computation result $DM \in Z_p^n$ provided by the cloud service provider (CSP) and issue the result to the requesting parties (RPs).

4.2. Design goals

We design our scheme with the following goals in mind.

- **Data confidentiality**: As the semi-trusted-but-curious CSP may snoop the information of the DPs' outsourced data, which break the confidentiality of DPs' data. Our scheme should be equipped with an encryption mechanism to guarantee the computation confidentiality.

- **Public verifiability**: As the CSP may provide fake results due to some financial or other reasons, the scheme should be able to let the result be verified by any PAP to guarantee the correctness of the computation.

- **Efficiency and feasibility**: To support the cloud computing scenario, the power of each entity should be considered. Thus, the computation complexity and storage consumption of different entity should be restricted respectively.

4.3. Notation

Table 2 summarizes the notation used in our scheme.

Table 2: Notations

Symbols	Description	Remark
PK_H	The FHE public key generated by TA	We use this notation to denote various public keys generate by TA
SK_H	The FHE private key generated by TA	We use this notation to denote various secret keys generate by TA
PK_x	The public key of entity x	
SK_x	The secret key of entity x	
$\{D_i\}_{i=1}^n$	The input data to F	
F	The function to evaluate	
$\{ED_i\}_{i=1}^n$	The homomorphically encrypted data of D_i	
DM	The homomorphically encrypted function evaluation result	
$E(PK_x, X)$	The encryption of X with PK_x	
RD_i	The random data chosen by DP in the same form of D_i	RD_i should be refreshed periodically
RDM_i	The homomorphically encrypted function evaluation result of RD_i with the same function F.	

5. The Proposed Scheme

In this section, we first give an overview of our scheme and then describe the concrete construction of our scheme in detail.

5.1. Overview

Our scheme consists of the following six procedures.

1. **System setup**: The TA generate the system's fully homomorphic secret key pairs. Each entity generates its public and private key pair and broadcasts its public key to the whole system.
2. **Data outsourcing**: The DP encrypts the his data, signs them together with the function to be evaluated, and sends them to the CSP for processing.

3. **Function evaluation**: The CSP evaluates the function over received data and then randomly perturbs the computation result.
4. **Data request and response**: The RP requests for the computation result from the TA. The TA verifies the eligibility of the request, re-encrypts the computation result, and sends it to the RP.
5. **Audit preparation**: The DP transforms the outsourced function into an equation via polynomial factorization, generates parameters for subsequent verification.
6. **Result audit and decryption**: The PAP retrives the encrypted data and the verification function from the DP and encrypted computation result from the TA. The RAP then verifies the computation result and sends the result to the RP if deemed correct. The RP decrypts the computation result.

5.2. Detailed Construction

In this section, we detail the construction of our scheme. The following procedures show how the DP outsources its data to the CSP and the RP receives the result along with the proof of correctness.

System setup: Given security parameter λ, the TA executes $\mathsf{Keygen}(1^\lambda)$ to generate a FHE public and private key pair PK_H and SK_H and send them to DPs. Each entity generates its public and private key pairs and broadcast its public key to every other party in the system.

Data outsourcing: DP provides its data set $\{D_i\}_{i=1}^n$ w.r.t. function F. To prevent the CSP from learning original data $\{D_i\}_{i=1}^n$, the DP encrypts each data item D_i with PK_H provided by the TA to generate an encrypted data $ED_i = \mathsf{Enc}(PK_H, D_i)$ and signs $\{ED_i\}_{i=1}^n$ with its private key SK_{DP}. This signature allows other party to verify that ED_i was indeed generated by DP.

Function evaluation: On receiving data $\{ED_i\}_{i=1}^n$ and corresponding signature, the CSP first verifies the signature to check if $\{ED_i\}_{i=1}^n$ was indeed sent by DP. If the verification succeeds, the CSP evaluates function F over $\{ED_i\}_{i=1}^n$ in a fully homomorphic way to obtain $\mathsf{Enc}(PK_H, DM) = Eval(\{ED_1, \ldots, ED_n\}, C)$, where C is the evaluated circuit corresponding to function F. The CSP then perturbs the encrypted result $\mathsf{Enc}(PK_H, DM)$ by multiplying it with a random number r_a to get $\mathsf{Enc}(PK_H, DM * r_a)$.

Data request and response: Suppose that requesting party RP requests for the function evaluation result from the CSP. It first sends a request containing F along with its signature to the CSP. On receiving the request, the CSP first forwards it to the TA to check the eligibility of RP. If RP is deemed eligible, the TA requests the CSP for the perturbed result $\mathsf{Enc}(PK_H, DM * r_a)$. It then decrypts $\mathsf{Enc}(PK_H, DM * r_a)$ using private key SK_H to obtain $DM * r_a$ and re-encrypts $DM * r_a$ with RP's public key to obtain $E(PK_{RP}, DM * r_a)$. The TA then sends $E(PK_{RP}, DM * r_a)$ along with its signature back to the CSP. After verifying the TA's signature, the CSP computes $E(PK_{RP}, r_a)$ and sends $E(PK_{RP}, r_a)$, $E(PK_{RK}, DM * r_a)$ along with its signature to the requesting party RP.

Audit preparation: According to the polynomial factorization formula $f(x) - f(a) = \sum_{i=1}^n (x_i - a_i)q_i(x)$, DP decomposes $F'(x) = F(x) - F(D_1, \ldots, D_n)$ into $\sum_{i=1}^n (x_i - D_i)q_i(x)$, where $x \in Z_p^n$ and $\{q_i(x)\}_{i=1}^n$ are the polynomials generated by multivariate polynomial

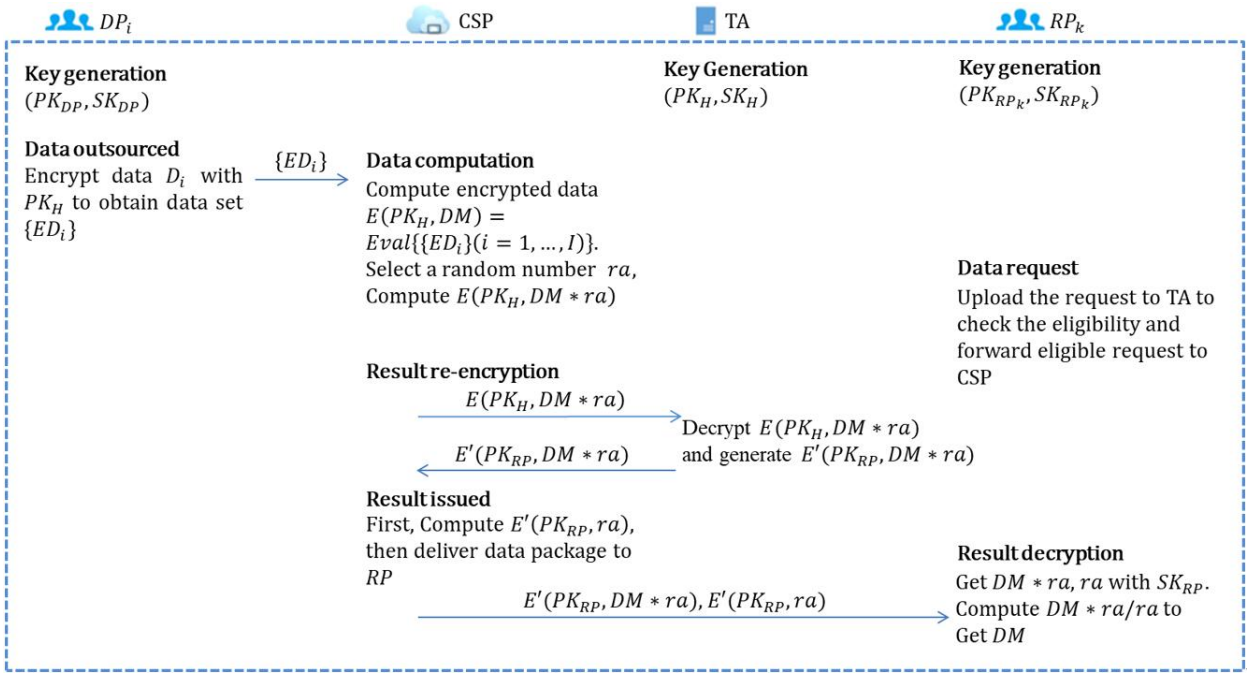

Figure 2: Computation phase

factorization. Meanwhile, DP chooses a set of random data $\{RD_1, \cdots, RD_n\}$ and computes $RDM = F(RD_1, \dots, RD_n)$ and $\{q_i(RD)\}_{i=1}^n$.

For the first time when the PAP is required to audit the data processing result provided by CSP w.r.t. function F from RP, the PAP forwards the request to DP along with its signature. When receiving this request, DP sends $\langle \mathsf{Enc}(PK_H, \{q_i(RD)\}_{i=1}^n), \mathsf{Enc}(PK_H, RDM), F\rangle$ to the PAP. The PAP then stores the information for any subsequent audit w.r.t. function F until DP refreshes $\{RD_i\}_{i=1}^n$.

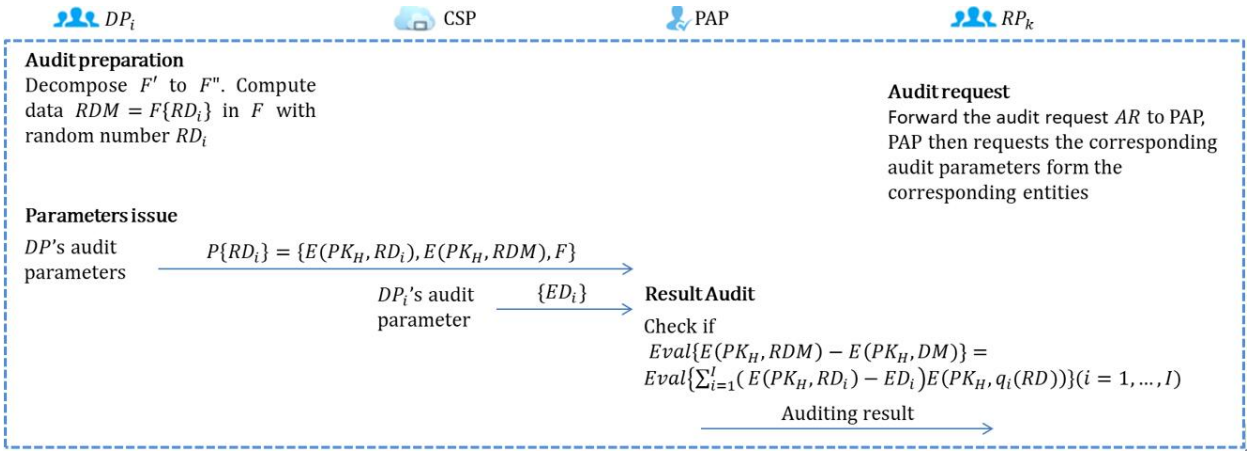

Figure 3: Verification phase

Result audit and decryption: The key operation of the audit process is to evaluate the

polynomial factorization formula in a fully homomorphic manner. Specifically, the PAP first retrieves $\{ED_i\}_{i\in I}$ w.r.t. F from the CSP. The PAP then checks if the following equation holds.

$$Eval\{\mathsf{Enc}(PK_H, RDM) - \mathsf{Enc}(PK_H, DM)\} \stackrel{?}{=} Eval\{\sum_{i=1}^{n}(\mathsf{Enc}(PK_H, RD_i) - ED_i) \mathsf{Enc}(PK_H, q_i(RD))\}. \tag{4}$$

If the verification succeeds, then the computation result is considered correct and incorrect otherwise. Finally, the PAP sends the auditing result to RP. After confirming that the audit result is valid, RP finally gets the result by first decrypting $E(PK_{RP}, DM * r_a)$ and $E(PK_{RP}, r_a)$ with its secret key SK_{RP} to get $DM * r_a$ and r_a whereby to recover the result DM.

Figs. 1 and 2 show the whole procedure of our proposed scheme by computation phase and verification phase, respectively.

6. Security Analysis and Performance Evaluation

In this section, we analyze the security and performance of the proposed scheme. In order to evaluate the performance of our verifiable outsourced computation scheme, we have implemented our system using libraries NTL [4], GMP [2], and FHE [1] to realize the virtual execution of each entity of the system. We run our scheme on a workstation with Intel(R) CoreTM i7 4710HQ CPU and 8-GB RAM running Ubuntu 14.04 to virtually execute the functions of DP, CSP, TA, PAP, and RP. In our implementation, we applied BGV fully homomorphic encryption [36], RSA for Public Key Cryptosystem (PKC), and SHA-1 hash function.

We first prove the security of our scheme. Then we analyze the performance of our protocol in the aspects of computation complexity and communication cost. Next, we test the operation time of the scheme and analyze the scalability. Finally, comparing with our previous work, we further show the improvement and the advantages of our new scheme.

6.1. Security Analysis

We first prove the security guarantee of the proposed system.

Theorem 2. *The computation confidentiality is guaranteed against the CSP.*

Proof. We prove this theorem through the following game. Suppose that an adversary $\mathcal{A}$ can attack our system with non-negligible advantage. It conducts the attack by the following steps.

First, the challenger $\mathcal{C}$ runs the algorithm and generate the public key PK_H to the adversary and keeps the private key SK_H to itself. Then the adversary $\mathcal{A}$ randomly generates two equal-length messages m_0 and m_1 and send them to $\mathcal{C}$. Randomly flipping a coin b, $\mathcal{C}$ encrypts the message m_b using public key PK_H to generate a ciphertext C and sends the ciphertext to $\mathcal{A}$. Finally, $\mathcal{A}$ makes a guess on $b' \in (0, 1)$ for b. $\mathcal{A}$ wins the game if $b = b'$.

The security of our scheme relies on Ring-Learning With Error (RLWE). Assuming RLWE, if m_0 and m_1 are randomly chosen by the challenger $\mathcal{C}$, the ciphertexts C_0 and C_1 are pseudorandom. Thus, when $\mathcal{C}$ flipping a coin b and encrypt the message m_b using public key PK_H to generate a ciphertext C, it is impossible for the adversary $\mathcal{A}$ to guess a valid b' such that $b = b'$ with a probability of higher than $1/2$.

Theorem 3. *The verification information is confidential against detection of PAP.*

PROOF. We prove this theorem by the following game. Suppose that an adversary can attack our scheme by generating a cheating result that can pass the verification with non-negligible advantage. It should conduct the attack by following steps.

First, the challenger $\mathcal{C}$ randomly selects a plaintext set $M = (m_0, \ldots, m_n)$. He then runs the algorithm to encrypt this data set to get the ciphertext set $C = (c_1, \ldots, c_n)$ and send it to the adversary $\mathcal{A}$ to compute the outsourced function F. To cheat the challenger $\mathcal{C}$, $\mathcal{A}$ uses a subset $C' \subset C$ for the function F and computes $CT' = F(C')$, where $CT' \neq CT$. Assume that $\mathcal{A}$ sends CT' to the challenger $\mathcal{C}$. To win the game, challenger $\mathcal{C}$ need find $F'(CT') = \sum_{i=1}^{n}(x_i - D_i)q_i(x)$, which means that $F(x) - F(DM') = \sum_{i=1}^{n}(x_i - D_i)q_i(x)$, where $DM' = D(SK_H, CT')$. Since the function holds the property of homomorphism, it equals to the function that $DM' = f(M) = DM$, which contradicts with $CT' \neq CT$. Thus, $\mathcal{A}$ cannot win the game with non-negligible advantage.

6.2. Performance Evaluation

6.2.1. Computation Complexity Analysis

We now analyze the computation complexity of the proposed scheme. The DP encrypts the outsourced data itself and generates the audit preparation data. Since we only need to generate the preparation data set for each function once during the audit setup procedure and it is computed over plaintext, we ignore the computation of audit preparation. Considering encryption, the computation complexity is proportional to the security parameter λ^2, thus the computational complexity at the DP is $\widetilde{O}(\lambda^2)$. As to the CSP that is responsible for ciphertext computation, since we use fully homomorphic encryption as our tools to compute the ciphertext, the computational complexity for each input data item is $\widetilde{O}(\lambda \times L^3)$, where λ is the security parameter and L is the depth of the circuit, which is proportional to the degree of function F. Since our scheme allows the CSP to deal with n input at one time, the total computational complexity at the CSP is also proportional to n. The computational complexity at the CSP is thus $\widetilde{O}(n \times \lambda \times L^3)$. The TA needs to re-encrypt the computation result, which needs to first decrypts the ciphertext result using the fully homomorphic secret key and then encrypts that message using the RP's public key. The complexity of both operations are both proportional to the security parameter λ and the length $poly(L)$ of the ciphertext. The computational complexity at the TA is thus $O(\lambda \times poly(L))$. Moreover, the RP is responsible for decrypting the final ciphertext result, whose complexity is also proportional to the security parameter λ of the system and the length of the ciphertext. The computational complexity at the RP is thus $O(\lambda \times poly(L))$. Finally, the PAP is in charge of verifying the correctness of the computation result, which needs to use fully

homomorphic encryption algorithms to execute computation. Therefore, its computational complexity is $\widetilde{O}(n' \times \lambda \times L'^3)$, where L' is the depth of the circuit proportional to the function $\sum_{i=1}^{n}(x_i - D_i)q_i(x)$ and n' is the number of variables of the function. Table 2 shows the computational complexity of different procedures at each entity in our system.

Table 3: Computational complexity at different Entities

Entity	Procedure	Computation complexity
DP	Data outsourced	$\widetilde{O}(\lambda^2)$
CSP	Outsourced computation	$\widetilde{O}(n \times \lambda \times L^3)$
TA	Result re-encryption	$O(\lambda \times poly(L))$
PAP	Result verification	$\widetilde{O}(n' \times \lambda \times L'^3)$
RP	Result decryption	$O(\lambda \times poly(L))$

Table 4 compares the computation complexity of the proposed scheme with some latest related works. We can see that by adding a light result re-encryption algorithm to the scheme, our scheme significantly reduces the computation overhead on the RP's side. Meanwhile, under the new verification phase, the computation complexity at the verifier is also reduced to only depending on the number of variables.

Table 4: Computational complexity comparison

Scheme procedure	FHE	GGP	Our scheme
Data outsourced	$\widetilde{O}(\lambda^2)$	$\widetilde{O}(\lambda^2)$	$\widetilde{O}(\lambda^2)$
Outsourced computation	$\widetilde{O}(n \times \lambda \times L^3)$	$\widetilde{O}(n \times \lambda \times L^3)$	$\widetilde{O}(n \times \lambda \times L^3)$
Result re-encryption			$O(\lambda \times poly(L))$
Result verification	$\widetilde{O}(n \times \lambda \times L^3)$	$\widetilde{O}(n' \times \lambda \times L^3)$	$\widetilde{O}(n' \times \lambda \times L'^3)$
Result decryption	$\widetilde{O}(\lambda^2)$	$\widetilde{O}(\lambda^2)$	$O(\lambda \times poly(L))$

Note: $O(n) \ll \widetilde{O}(n)$, $n' < n$, $L' < L$.

6.2.2. Communication Cost

We now analyze the communication cost of the proposed scheme. Since the system setup and the audit preparation procedure only take place in the scheme once, we ignore the communication costs of these two phases and analyze the communication cost of the proposed scheme with regard to the following communication channel:

DP to CSP: Encrypted data are signed and sent from the DP to the CSP. Our implementation shows that the length of the ciphertext is 68 bytes while that of plaintext is 8 bytes. The length of an RSA signature is 256 bytes. In other words, for every data item sent from the DP to the CSP, the communication cost is 68+256=324 bytes. The communication cost is proportional to the number of data items n.

CSP to TA: The CSP needs to request the TA to execute the re-encryption when the RP requests for the final result, it needs to send 68 bytes of encrypted data alongside with

its 256 bytes signature. The total communication cost of this procedure is thus 256+68=324 bytes.

CSP to RP: In our scheme, we use RSA algorithm to re-encrypt the final processing result. To recover the final plaintext result, another auxiliary encrypted version of ra is also sent to the RP with the signature. Since all the three pieces of data are encrypted by RSA, the total communication cost incurred is 256*3=768 bytes.

TA to CSP: After re-encrypting the ciphertext result by RSA, the TA issues the 256 bytes final result along with its signature to the CSP. Since they are encrypted using the RSA algorithm, the total communication cost of this procedure is 256*2=512 bytes.

RP to PAP: The RP sends 3 pieces of data to the PAP to request of the final processing result, which is 68 bytes $\mathsf{Enc}(PK_H, DM)$, and two 256 bytes signatures are generated via RSA. Thus, the total communication cost of this procedure is 68+256*2=532 bytes.

Other communication costs of the scheme are introduced by queries or requests, they are all consists of commands, keys, and signatures which have fixed size. In practice, they do not incur much communication cost into the system. Table 4 shows the main communication costs of these channels in our scheme.

Table 5: Main communication costs between each entity in our system

Communication Channel	Communication package length (byte)	Communication cost
DP to CSP	68+256=324	$O(n)$
RP to CSP	256*2=512	$O(1)$
CSP to TA	26+68=324	$O(1)$
CSP to RP	256*3=768	$O(1)$
TA to CSP	256*2=512	$O(1)$
RP to PAP	68+256*2=532	$O(1)$

Remark 1. First, our scheme allows the DP to outsource its computation to the CSP. Although the computational complexity and the communication cost of computing ciphertext using fully homomorphic encryption are non-trivial, the CSP with adequate computation and storage resources can handle this kind of computations. Second, when there exists many RPs to request for the computation result provided by the CSP, each RP can issue its public key and its corresponding required function F to CSP. Then the CSP checks each RP's eligibility and respond to RP's request using each RP's public key individually and returns the computation result to the corresponding RP. The propose scheme thus supports multiple outputs. Finally, when each PAP asks to verify the computation result, according to function F requested by the PAP, the CSP can provide the necessary information to allow individual verification. Knowing that each PAP only responds to its part of the computation, we reduce the depth of the computation circuit, this computation consumption is reduced a lot. As a result, this scheme also supports big data verification.

6.2.3. Experiment Results

We simulated our system in a workstation running Ubuntu by applying HElib library [3] and tested the operation time of our scheme. In the simulation, we randomly selected N

numbers as values of the input variables of a polynomial function F and ran the function F homomophically to test the computation phase of our system. In order to test the verification phase of our system, we transform function F based on our designed verification algorithm and ran the transformed function homomorphically according to HElib. With the input length of 8 bytes, each homomorphic encryption, homomorphic decryption, addition, and multiplication take 1200 ms, 600 ms, 4 ms, and 10^4 ms, respectively. Figs. 4 and 5 show the simulation results on different numbers of data inputs.

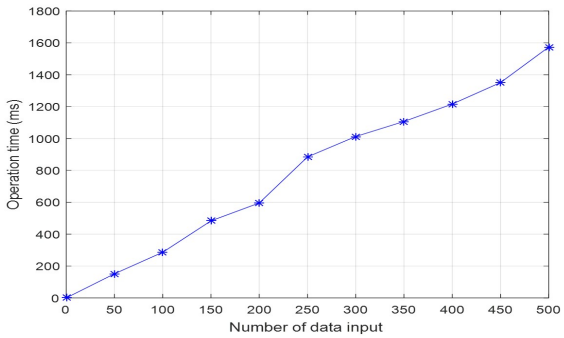

Figure 4: Time consumption of addition

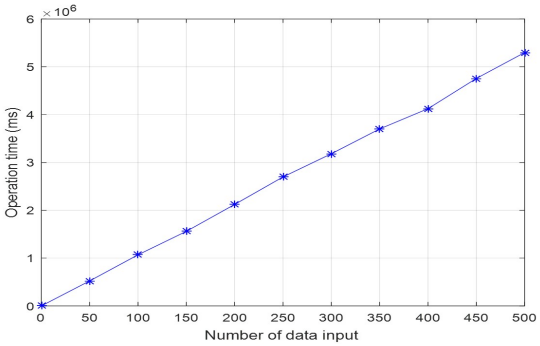

Figure 5: Time consumption of multiplication

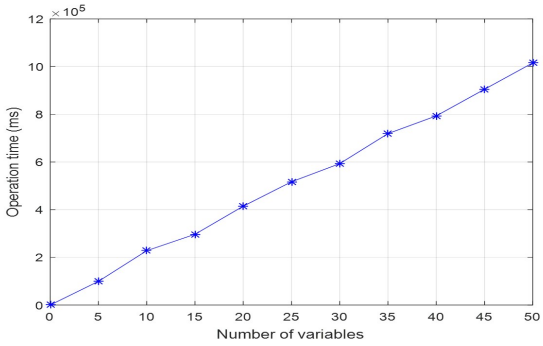

Figure 6: Time consumption of the verification

By analyzing the audit algorithm of our scheme, it shows that the consumption of the auditing procedure is not affected by the degree of the outsourced function F, so our scheme can support functions with high degree. It also shows that the computation time of the auditing procedure is proportional to the number of variables in the function, which has a fixed time of 2×10^4 ms per variable. Fig. 6 shows the computation time of the verification of functions with a different number of variables. Since no matter how complex the outsourced function is, the time consumption of the verification function only depends on the number of variables, our scheme is very efficient in dealing with multi-user computation.

6.3. Comparison with Existing Work

We further compare our proposed scheme with some latest work based on different techniques: FHE [24] and GGP [21] with respect to verification for a polynomial function with

degree L and N variables. Especially, these schemes also support encrypted computation with guarantee of confidentiality of users' data. Fig. 7 shows the performance of our scheme in comparison with the previous systems. We use a polynomial function with L degree and N variables as our test application.

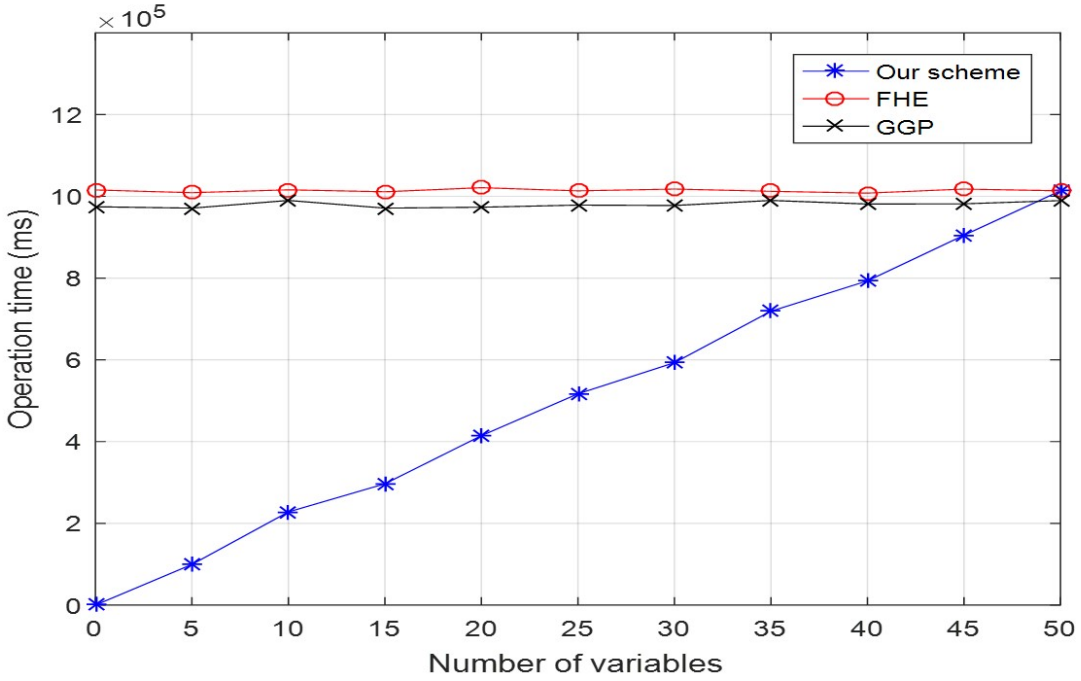

Figure 7: Performance relative to prior schemes. Note: Since multiplication algorithm is always the largest consumption in verification both in our scheme and the prior work, we focus on the multiplication of different variables, so that $L > N$ at all time. Here $L = 50$, the result is similar chosen different L.

Comparing with FHE and GGP, Fig. 7 shows that the verification consumption of our scheme only depends on the number of variables implied into the computation procedure. However, FHE and GGP depend on the degree of the computation function, which is always larger than the number of variables. Thus, at most time, our scheme has a better performance on the verification than that of these schemes and reduce the workload of the verifier. Comparing with some other latest work (such as PCPs, Pinocchio), Table 5 shows that although these schemes have a great performance at verifying the computation result of the CSP and may support public verification, they did not consider the problem of how to guarantee the security of the outsourced data.

In comparison with above prior work, our scheme supports computation confidentiality, public verification, and fast verification. In order to securely computation and verify the outsourced function, our scheme introduces a method to compute the outsourced function and verify the computation result of the CSP both over ciphertexts. Moreover, our proposed audit procedure enables the public to verify the computation result. Finally, the verification consumption of our scheme only depends on the number of variables used in the computation function. That increases the efficiency of the verification.

Table 6: Scheme comparison

Schemes	PCPs[13]	GGP[12]	FHE[2]	Pinocchio[22]	Our scheme
Confidentiality	×	√	√	×	√
Public verification	×	×	×	√	√
Efficient verification	√	×	×	√	√

Note: ×- not supported; √ - supported.

7. Conclusion

In this paper, we have proposed a novel scheme for verifiable computation over encrypted data to ensure trusted cloud computing. By introducing a semi-trusted entity TA and a limited trusted entity PAP, our scheme enables outsourced function evaluation over ciphertexts and supports the public to verify the computation result. It also ensures outsourced data confidentiality during computation verification. We thoroughly evaluate the performance of the proposed scheme with regard to security, computation cost, and communication cost. The evaluation results confirm the efficacy and efficiency of our scheme.

Regarding future work, we will further try other light encryption algorithms instead of fully homomorphic encryption to increase the efficiency of the encrypted data computation part of the scheme. How to perform computing audit in an encrypted form is an interesting and challenging research topic worth our further investigation.

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), Academy of Finland (grant No. 308087), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), and the 111 project (grants B08038 and B16037), as well as the China Scholarship Council for two years study at University of Delaware.

References

[1] Fhe library, 2017.
[2] Gmp library, 2017.
[3] Helib, 2017.
[4] Ntl library, 2017.
[5] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Trans Inf Theor*, 29:208–210, 2006.
[6] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. pages 863–874, 2013.
[7] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *Cryptology–EUROCRYPT*, pages 149–168. Springer, 2011.
[8] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homo-morphic encryption without bootstrapping. In *ACM 3rd Innov. Theor. Comput. Sci. Conf.*, pages 1–25, 2012.

[9] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput*, 43(2):831–871, 2014.

[10] R. Canetti, B. Riva, and G. N. Rothblum. Two protocols for delegation of computation. In *the 6th international conference on information theoretic security, ICITS'12*, pages 37–61, ser, 2012. ICITS'12. Springer-Verlag, Berlin.

[11] D. Catalano and D. Fiore. Practical homomorphic macs for arithmetic circuits. In *EUROCRYPT 2013*, volume 7881, pages 336–352, 2013.

[12] D. Catalano, A. Marcedone, and O. Puglisi. Linearly homomorphic structure preserving signatures: new methodologies and applications. *IACR Cryptology ePrint Archive*, 2013.

[13] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou. New publicly verifiable databases with efficient updates. *IEEE Transactions on Dependable and Secure Computing*, 12(5):546–556, 2015.

[14] X. F. Chen, J. Li, J. F. Ma, J. Weng, and W. Lou. Verifiable computation over large database with incremental updates. *IEEE Transactions on Computers*, 65(10):3184–3195, 2016.

[15] S. G. Choi, J. Katz, R. Kumaresan, and C. Cid. Multi-client non-interactive verifiable computation. In *the 10th theory of cryptography conference on Theory of Cryptography (TCC'13)*, pages 499–518, 2013.

[16] K. M. Chung, Y. Kalai, and S. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Cryptology–CRYPTO*, pages 483–501, 2010.

[17] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology– EUROCRYPT*, pages 24–43. Springer, 2010.

[18] W.X. Ding, Z. Yan, and R. H. Deng. Encrypted data processing with homomorphic re-encryption. *Information Sciences*, 409-410:35–55, 2017.

[19] W.X. Ding, Z. Yan, and R. H. Deng. Privacy-preserving data processing with flexible access control. *IEEE Trans on Dependable and Secure Computing*, 2017.

[20] D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM conference on Computer and communications security*, pages 501–512, 2012.

[21] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computation: outsourcing computation to untrusted workers. pages 465–482, 2010.

[22] R. Gennaro and D. Wichs. Fully homomorphic message authenticators. In *ASIACRYPT 2013, Part II. LNCS*, volume 8270, pages 301–320, 2013.

[23] C. Genrty. Fully homomorphic encryption using ideal lattice. In *the forty-first annual AMC symp. on Theory of computing*, pages 169–178. C. Genrty, 2009.

[24] C. Gentry. A fully homomorphic encryption scheme, 2009.

[25] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *ACM Symp Theory Comput*, pages 113–122, 2008.

[26] S. Hohenberger and Lysyanskaya A. How to securely outsource cryptographic computations. In *Second international conference on Theory of Cryptography (TCC'05)*, pages 264–282.

[27] J. Li, X.Y. Huang, J. W. Li, X. F. Chen, and Y. Xiang. Securely outsourcing attribute-based encryption with checkability. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2201–2210, 2014.

[28] J. Li, J. W. Li, X. F. Chen, C. F. Jia, and W. j. Lou. Identity-based encryption with outsourced revocation in cloud computing. *IEEE Transactions on Computers*, 64(2):425–437, 2015.

[29] J. Li, Y. Li, X. F. Chen, P. Lee, and W. J. Lou. A hybrid cloud approach for secure authorized deduplication. *IEEE Transactions on Parallel and Distributed Systemss*, 26(25):1206–1216, 2015.

[30] J. Li, Y. Li, X. F. Chen, P. Lee, and W. J. Lou. A hybrid cloud approach for secure authorized deduplication. *IEEE Transactions on Parallel and Distributed Systems*, 26(5):1206–1216, 2015.

[31] W. Li, K. Xue, Y. Xue, and J. Hong. Tmacs: a robust and verifiable threshold multi-authority access control system in public cloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1484–1496, 2016.

[32] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *the 10th theory of cryptography conference on Theory of Cryptography (TCC'13)*, pages 222–242, 2013.

[33] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

[34] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: verifiable computation from attribute-based encryption. In *9th international conference on Theory of Cryptography (TCC'12)*, pages 422–439, Heidelberg, 2012. Springer Berlin.

[35] P. Renjith and S. Sabitha. Verifiable el-gamal re-encryption with authenticity in cloud. In *Computing, communications and networking technologies (ICCCNT), 4-6*, pages 1–5, July 2013.

[36] R. L. Rivest, L. Adleman, and N. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–180, 1978.

[37] S. Setty, R. McPherson, A. J. Blumberg, and M. Walfish. Making argument systems for outsourced computation practical (sometimes). 2012.

[38] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish. Taking proof-based verified computation a few steps closer to practicality. 2012.

[39] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS 2011, the internet society*, 2 2011.

[40] J. R. Thaler. Practical verified computation with streaming interactive proofs, 2013.

[41] V. Vu, S. Setty, A. J. Blumbery, and M. Walfish. A hybrid architecture for interactive verifiable computation. In *Security and privacy (SP), 19-22*, pages 223–237, May 2013.

[42] Z. Wen, J. Luo, H. Chen, J. Meng, X. Li, and J. Li. A verifiable data deduplication scheme in cloud computing. pages 85–90, 9 2014.

[43] Z. Yan, W.X. Ding, X.X. Yu, H.Q. Zhu, and R. H. Deng. Deduplication on encrypted big data in cloud. *IEEE Transactions on Big Data*, 2(2):138–150, 2016.

[44] Z. Yan, X.Y. Li, and R. Kantola. Controlling cloud data access based on reputation. *Mobile Networks and Applications*, 20(6):4828–839, 2015.

[45] Z. Yan, X.Y. Li, M.J. Wang, and A.V. Vasilakos. Flexible data access control based on trust and reputation in cloud computing. *IEEE Transactions on Cloud Computing*, pages 485–498, 2017.

[46] Z. Yan and W.Y. Shi. Cloudfile: A cloud data access control system based on mobile social trust. *Journal of Network and Computer Applications*, 86:46–58, 2016.

[47] Z. Yan, X.X. Yu, and W.X. Ding. Context-aware verifiable cloud computing. *IEEE Access*, 5:2211–2227, 2017.

[48] Z. Yan, L.F. Zhang, W.X. Ding, and Q.H. Zheng. Heterogeneous data storage management with deduplication in cloud computing. *IEEE Transactions on Big Data*, 2017.

[49] X. X. Yu, Z. Yan, and V. V. Athanasios. A survey of verifiable computation. *Mobile Network and Applications*, 22(3):438–453, 2017.