IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INTERNET-OF-THINGS ATTACKS AND DEFENSES: RECENT ADVANCES AND CHALLENGES

Received October 7, 2020, accepted October 18, 2020, date of publication October 23, 2020, date of current version November 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033306

Securely Outsource Modular Exponentiations With Single Untrusted Cloud Server

DEZHI AN, YAN LI, SHENGCAI ZHANG, AND JUN LU
School of Cyber Security, Gansu University of Political Science and Law, Lanzhou 730070, China

Corresponding author: Dezhi An (adz6199@gsli.edu.cn)

This work was supported in part by the Team Project of Collaborative Innovation in Universities of Gansu Province under Grant 2017C-16, and in part by the Major Project of Gansu University of Political Science and Law under Grant 2016XZD12.

ABSTRACT Modular exponentiation is one of the most fundamental operations in lots of encryption and signature systems. Due to the heavy computation cost of modular exponentiation, many schemes have been put forward to securely outsource modular exponentiation to cloud. However, most of the existing approaches need two non-colluded cloud servers to securely complete the modular exponentiation, which will result in private data leakage upon the cloud servers collude. Besides, most existing schemes assume both base and exponent in modular exponentiation are private, which does not conform to many real-world applications. For example, in public key encryption system that uses modular exponentiation, one of base and exponent is the public key, and the other is a message. Usually, only the message should be privately protected. In this paper, we propose two secure outsourcing schemes for fixed-base (public base and private exponent) and fixed-exponent (private base and public exponent), respectively. In the proposed schemes, we employ only one cloud server and can thus avoid collusion attack. Further, we achieve an efficient and secure Paillier encryption outsourcing scheme based on our secure modular exponentiation outsourcing methods. Additionally, we theoretically analyze our overheads and leverage simulation experiments to evaluate our proposed solutions, which show our schemes can achieve high efficiency.

INDEX TERMS Cloud computing, outsource-secure algorithm, modular exponentiations, single server.

I. INTRODUCTION

Cloud computing, as a new network service model, has become a hot topic, since it can economically and efficiently provide users with dynamic storage and computation services in a pay-as-you-go manner. The users can thus avoid large capital expenditures in hardware, software deployment and maintenance. Besides, by leveraging the computing resource of cloud servers, a resource-constrained device in the Internet of Things (IoT) can finish their computation tasks that go beyond its computation capability.

Nevertheless, it will inevitably bring some new security challenges when users outsource the computation tasks to the cloud [1], [20], [27], as the direct physical control will be transferred to the cloud server. Firstly, the cloud servers cannot be fully trusted and the outsourced computation tasks usually contain some private information. Therefore, the most basic security requirement is confidentiality of the computation tasks. Although fully homomorphic encryption provides a theoretical way for the problem, there is no practical fully homomorphic encryption scheme now. Secondly, the untrusted servers may not return the right results for some reasons, such as loaf on the job, software bugs, hardware error. Accordingly, the users should be able to verify the validity of the results with little cost which are returned by the cloud servers (the cost must be far less than the outsourcing task).

As well known, modular exponentiation is one of the most fundamental and expensive operations in most encryption and signature systems, such as RSA, Paillier encryption system [31]. For resource-limited devices in IoT, both the running time and cost of the computations are very huge. Thus, it is fatal for some schemes which need to complete some modular exponentiations in a short time. Most of the secure outsourcing of modular exponentiation is implemented by employing two cloud servers [2], [3], [6], [19]. However, all of these schemes cannot resist the collusion attack of the cloud servers. Ding *et al.* [26] recently proposes

The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

a secure modular exponentiation outsourcing scheme. Nevertheless, they assume both base and exponent in modular exponentiation are private, which does not conform to many real-world applications. For example, in public key encryption system that uses modular exponentiation, one of the base and exponent is public key, and the other is message. In fact, only the message should be privately protected. In this paper, we propose two new secure outsourcing algorithms for fixed-base (public base and private exponent) and fixed-exponent (private base and public exponent) in the model of a single malicious cloud server. Further, we achieve an efficient and secure Paillier encryption outsourcing scheme based on our secure modular exponentiation outsourcing methods. Additionally, we theoretically analyze our overheads and leverage simulation experiments to evaluate our proposed solution, which shows our solutions can achieve high efficiency. In general, our main contributions in this paper are as follows.

- We propose two secure outsourcing schemes for fixed-base (public base and private exponent) and fixed-exponent (private base and public exponent) modular exponentiation, respectively. Our proposed schemes require only one untrusted server.
- Based on our modular exponentiation outsourcing schemes, we propose an efficient and secure Paillier encryption outsourcing scheme.
- We theoretically analyze our schemes in security, efficiency and checkability, and conduct simulation experiments to evaluate our schemes. The results show that our schemes can securely complete the modular exponentiation and encryption tasks outsourcing in an efficient manner.

The rest of this paper is organized as follows. In Section II, we simply review the related work. In Section III, we present the system model and security definitions of our schemes. In Section IV, we describe our two secure modular exponentiations outsourcing schemes, and present the security and complexity analysis. Then, we propose secure Paillier encryption outsourcing scheme in Section V. Finally, we conclude this paper in Section VI.

II. RELATED WORK

How to securely outsource expensive computations has been attracting the attention from computer science community. Hohenberger and Lysyanskaya [3] presented the formal security definition of secure outsourcing model and proposed a CCA2 secure outsourcing encryption scheme. Barbosa and Farshim [5] presented a modular construction of delegatable homomorphic encryption from fully homomorphic encryption, functional encryption and MAC, and then described how to build a secure outsourcing computation scheme from delegatable homomorphic encryption. However, fully homomorphic encryption is limited to immaturity and it cannot be put into practice.

There are also a lot of research works on the securely outsourcing computation. Chen *et al.* [6] proposed a new secure outsourcing algorithm EXP which improved in both efficiency and checkability. Atallah and Frikken [7], Benjamin and Atallah [8] proposed protocols to solve the problem of secure outsourcing for widely applicable linear algebra computations with homomorphic encryptions. Applebaum *et al.* [9] in 2010 presented an improved protocol based on the weak secret hiding assumption. Chevallier-Mames *et al.* [10] proposed the first algorithm for a secure delegation of elliptic-curve pairings in the model of one untrusted server. Then Parno *et al.* [11] described a construction of a multi-function computation delegation scheme.

However, outsourcers can't fully trust the cloud server. Plenty of works have been done to achieve secure and efficient outsourcing computations by the untrusted server. Mironov and Golle [22] first presented the notion of ringers to verify the truth of results from untrusted servers. Gennaro *et al.* [12] first integrated the notion of checkability to solve the issue of trust. In 2005, Hohenberger and Lysyanskaya [3] proposed a checkable algorithm that verify the results by two untrusted servers. Based on this work, they further achieved secure outsourcing encryption and signature, but the checkability of the algorithm is only $\frac{1}{2}$. Chen *et al.* [6] further studied and proposed an improved version, they rise the checkability to $\frac{2}{3}$. Ye *et al.* [21] made further improvements in checkability and it comes to $\frac{19}{20}$. Focusing on fixed-base and fixed-exponent, Ma *et al.* [2] proposed three secure outsourcing schemes in 2013 and the checkability reach $\frac{3}{4}$. One of their schemes is in the model of one untrust server and the checkability almost reaches 1. Wang *et al.* [13] proposed a new secure outsourcing scheme of modular exponentiation in one untrusted server, while their checkability is just $\frac{1}{2}$. Xue *et al.* [29] proposed a new joint distribution estimation under Local Differential Privacy which can be used to Naive Bayes Classification.

Recently, Zhao *et al.* [23] improved the checkability to nearly 1 with the method of blinding matrix, while the server needs to undertake too much computation. In addition, Cai *et al.* [24] proposed a new algorithm SmExp whose checkability can almost reach 1 with a single sever. Ding *et al.* [26] improved the checkability of algorithm EXP and SEXP to $\frac{119}{120}$ and $\frac{74}{75}$, while the client needs to do part of modular exponentiations.

III. SYSTEM MODEL AND SECURITY DEFINITIONS

A. SYSTEM MODEL

In this paper, our system model is composed of a client and a server. The client has input data x, and wants to gain the result $f(x)$ where the function f is appointed by the client. However, because of lacking enough computing resource, the client can not finish it by itself. Therefore, the client needs to complete it with the help of cloud servers. Considering that the input data x may be private, cloud servers should know

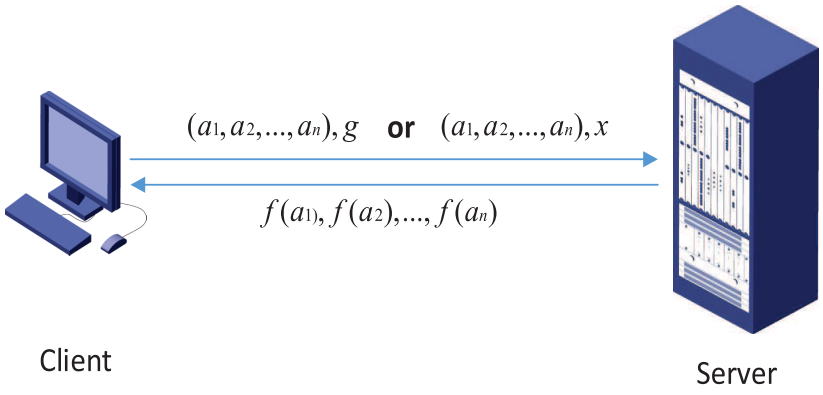

FIGURE 1. System model.

nothing about x. Thus the client will transform the private data such that the original input data can be well protected. With function f and the transformed input data, the cloud servers can complete calculation tasks. After receiving the computing results from the cloud server, the client can get the final result $f(x)$ with little cost. Meanwhile, the client also can verify the correctness of cloud server's returned data. In this paper, the function f represents modular exponentiation. Generally, this kind of modular exponentiation scheme can be described as follows:

1) Given modular exponentiations which will be computed, client invokes *Rand*.
2) The client blinds the inputs with the *rand* pairs and sends them to cloud server.
3) After receiving these transformed values, the cloud server compute and return the results to the client.
4) After receiving the results from the cloud server, the client verifies the correctness of the results. If the results are not right, the cloud server outputs "error". Otherwise, the client computes the real results of the modular exponentiations.

In our schemes, we propose two system models for two different situations, which is shown in Fig.1.

- For the fixed-base modular exponentiation (public base and private exponent) scheme, the client wants to compute $(g^{x_1}, g^{x_2}, \ldots, g^{x_t}) \bmod p$ where base g is public and exponent x_i $(1 \leq i \leq t)$ is private. With our method, the client divides the private exponent set into a public set A. The cloud server returns $g^{a_i} \bmod p$ $(a_i \in A)$ to the client. Finally, with the returned data from cloud server, the client can get g^{x_i} $(1 \leq i \leq t)$.
- For the fixed-exponent modular exponentiation (private base and public exponent) scheme, what the client needs is the result of $(g_1^x, g_2^x, \ldots, g_t^x) \bmod p$ where exponent x is public and base g_i $(1 \leq i \leq t)$ is private. With our method, the client divides the private base set into a public set A. The cloud server returns the result $a_i^x \bmod p$ $(a_i \in A)$ to the client. In the end, the client can get g_i^x $(1 \leq i \leq t)$ by some transformations of the results from cloud server.

B. SECURITY DEFINITIONS

In this paper, we follow the security definitions of outsourcing cryptographic computations proposed by Hohenberger and Lysyanskaya in [3]. Generally, it is said that the computation-constrained client T securely outsources some computation tasks to cloud server U, and (T, U) is an outsource-secure implementation of a cryptographic algorithm Alg, if (1) T and U correctly achieve Alg, i.e., Alg $= T^U$, and (2) assume that, instead of cloud server U, T is given oracle access to an adversary U' who records all its computation during every implementation and tries to act maliciously, but the adversary U' can learn nothing useful about the client's input and output. We will introduce Hohenberger and Lysyanskaya's formal definitions as follows.

Definition 1 (Algorithm With Outsource-I/O [3]): An algorithm Alg obeys the outsource input/output specification if it takes in five inputs, and produces three outputs: Alg $(x_{hs}, x_{hp}, x_{hu}, x_{ap}, x_{au}) \rightarrow (y_s, y_p, y_u)$.

Inputs: The first three inputs are generated by an honest party, and are classified by in what degree the adversary $A = (E, U')$ *has knowledge about them, where* E *is the adversarial environment that submits adversarially chosen inputs to Alg, and* U' *is the adversarial software operating in place of oracle* U.

- *The honest, secret input* x_{hs}*, which is unknown to both* E *and* U'.
- *The honest, protected input* x_{hp}*, which may be known by* E*, but is protected from* U'.
- *The honest, unprotected input* x_{hu}*, which may be known by both* E *and* U.

In addition, there are two adversarially-chosen inputs generated by the environment E*:*

- *The adversarial, protected input* x_{ap}*, which is known to* E*, but protected from* U'.
- *The adversarial, unprotected input* x_{au}*, which may be known to* E *and* U.

Outputs: Similarly, the outputs are defined as follows.

- *The secret output* y_s*, which is unknown to both* E *and* U'.

- *The protected output y_p, which may be known to E, but is unknown to U'.*
- *The unprotected y_u, which may be known by both parts of A.*

Definition 2 (Outsource-Security [3]): Let Alg be an algorithm with outsource I/O. A pair of algorithm (T, U) is said to be an outsource-secure implementation of Alg if:

1) *Correctness: T^U is a correct implementation of Alg.*
2) *Security: For all probabilistic polynomial-time adversaries $A = (E, U')$, there exist probabilistic expected polynomial-time simulators (S_1, S_2) such that the following pairs of random variables are computationally indistinguishable.*

- *Pair one: $EVIEW_{real} \sim EVIEW_{ideal}$*
  The real process:

$$EVIEW^i_{real} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, EVEIVEW^{i-1}_{real}, x^i_{hp}, x^i_{hu});$$
$$(tstate^i, ustate^i, y^i_s, y^i_p, y^i_u) \leftarrow T^{u'(ustate^{i-1})}(tstate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}).\}$$

  Thus, the view of E in the i-th round of the real process is $EVIEW^i_{real} = (estate^i, y^i_p, y^i_u)$. The overall view is just its view in the last round, i.e., $EVIEW_{real} = EVIEW^i_{real}$ if $stop^i = TRUE$. The ideal process also proceeds in rounds. In the ideal process, we have a stateful simulator S_1 who, shielded from the secret input x^i_{hs}, but given the non-secret outputs that Alg produces when run all the inputs for round i, decides to either output the values (y^i_p, y^i_u) generated by Alg, or replace them with some other values (Y^i_p, Y^i_u). Note that this is captured by having the indicator variable rep^i be a bit that determines whether y^i_p will be replaced with Y^i_p. In doing so, it is allowed to query oracle U'; moreover, U' saves its state as in the real experiment.
  The ideal process:

$$EVIEW^i_{ideal} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, EVEIVEW^{i-1}_{real}, x^i_{hp}, x^i_{hu});$$
$$(astate^i, y^i_s, y^i_p, y^i_u) \leftarrow Alg(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});$$
$$(sstate^i, ustate^i, Y^i_p, Y^i_u, ind^i) \leftarrow S_1^{U'(ustate^{i-1})}(sstate^{i-1}, \ldots, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}, y^i_p, y^i_u);$$
$$(z^i_p, z^i_u) = ind^i(Y^i_p, Y^I_u) + (1 - ind^i)(y^i_p, y^i_u).\}$$

  Thus, the view of E in the i-th round of the ideal process is $EVIEW^i_{ideal} = (estate^i, z^i_p, z^i_u)$. The overall view is just its view in the last round, i.e.,
  $EVIEW_{ideal} = EVIEW^i_{ideal}$ if $stop^i = TRUE$.
- *Pair two: $UVIEW_{real} \sim UVIEW_{ideal}$*
  The view that the untrusted software U' obtains by participating in the real process described in Pair One. $UVIEW_{real} = ustate^i$ if $stop^i = TRUE$.

  In the ideal process, we have a stateful simulator S_2 who, equipped with only the unprotected inputs (x^i_{hu}, x^i_{au}), queries U'. As before, U' may maintain state. The ideal process:

$$EVIEW^i_{ideal} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, EVEIVEW^{i-1}_{real}, x^i_{hp}, x^i_{hu}, y^{i-1}, y^{i-1});$$
$$(astate^i, y^i_s, y^i_p, y^i_u) \leftarrow Alg(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});$$
$$(sstate^i, ustate^i, Y^i_p, Y^i_u, ind^i) \leftarrow S_1^{U'(ustate^{i-1})}(sstate^{i-1}, x^{j^i}_{hu}, x^i_{au});$$

  Thus, the view of U' in the i-th round of the ideal process is $UVIEW^i_{ideal} = ustate^i$. The overall view is just its view in the last round, i.e.,
  $UVIEW_{ideal} = UVIEW^i_{ideal}$ if $stop^i = TRUE$.

Definition 3 (β-Checkable [3]): A pair of algorithms (T, U) is said to be a β-checkable implementation of Alg if (1) T^U is a correct implementation of Alg and (2) for any input x, if U' deviates from its advertised functionality during the execution of $T^{U'}(x)$, T will detect the error with probability no less than β.

Definition 4 (α-Efficient [3]): A pair of algorithms (T, U) is said to be an α-efficient implementation of Alg if (1) T^U is a correct implementation of Alg and (2) for any input x, the running time of T is no more than an α-multiplicative factor of the running time of Alg.

C. RAND ALGORITHM

The subroutine *Rand* is used to expedite calculation the speed of scheme [3]. The main inputs for *Rand* are a prime p and a base $g \in Z^*_p$ while the output is a random independent pair $(a, g^a \bmod p)$ for some $a \in Z^*_p$. We use $Rand_1$ and $Rand_2$ for our schemes respectively. $Rand_1$ is used to generate random pair $(a, g^a \bmod p)$ like *Rand* described above for the scheme of fixed-base exponent exponentiation. In addition, $Rand_2$ is used to generate random pair $(g, g^x \bmod p)$ for the scheme of fixed-exponent base exponentiation where $x \in Z_p, g \in G_p$. Until now, EBPV generator, taking into account the efficiency and security, is the most excellent algorithm, which runs in time $O(log^2 n)$ for an n-bit exponent.

IV. SECURE OUTSOURCING PROTOCOLS OF MODULAR EXPONENTIATIONS

In this paper, we propose two algorithms for secure outsourcing modular exponentiations. Both schemes only need one cloud server. The two schemes are executed within the cyclic group G_p, where p is a large secure prime and g is the generator of G_p.

A. SECURE FIXED-BASE MODULAR EXPONENT EXPONENTIATION OUTSOURCING ALGORITHM

The fixed-base modular exponentiation scheme is aimed at securely computing fixed-base exponent exponentiations by

outsourcing them to cloud server, i.e., $(g^{x_1}, g^{x_2}, \ldots, g^{x_t}) \bmod p$. Due to that the base g is public, what we should protect is the exponents $x_1, x_2, \cdots, x_t$. To reach the purpose that the cloud server cannot get any information about the exponents and the client can easily verify the correctness of the returned data, we transform the exponents into some other numbers which seem to be irrelevant to protect the privacy of exponents. In the end, the exponents $(x_1, x_2, \ldots, x_t)$ are translated into a set $A = \{a_1, a_2, \ldots, a_n\}$ which can be sent to cloud server.

The cloud server makes some computations based on the information (g, p, A) and returns the result to the client. According to the internal relation between set A and exponents $\{x_1, x_2, \ldots, x_t\}$, the client can figure out g^{x_i} and verify the correctness of the returned values with little cost. Next, the fixed-base exponent exponentiation scheme can be formally described as follows.

- Given $(g, x_1), (g, x_2), \ldots, (g, x_t)$ as input data to compute $(g^{x_1}, g^{x_2}, \ldots, g^{x_t})$, where $x_i \in Z_p$ and $i \in [1, t]$. The client first transforms the exponents $(x_1, x_2, \ldots, x_t)$ into another set $A = \{a_1, a_2, \ldots, a_n\}$ where m elements of set A can be summed to x_i for each $i \in [1, t]$. The set A can be regarded as the combination of set B, C and R. Set B is composed of $m-1$ numbers which are randomly selected from Z_p. Let $B = \{b_1, b_2, \ldots, b_{m-1}\}$. Set C is the result of the set of exponent x_i minus the sum of set B, then c_i is $c_i = x_i - \sum_{j=1}^{m-1} b_j$, $1 \le i \le t$. The client needs to invoke the subroutine $Rand_1$ e times to get pairs (g, g^{y_i}). Next, the rest set R can be generated by the way like set C, r_i can be generated by $r_{b_i} = y_i - \sum_{j=1}^{m-1} b_j$, $1 \le i \le e$ and $r_{c_k} = y_k - \sum_{j=1}^{t} c_j$, $1 \le k \le e$. Finally, the client, using a pseudo-random permutation function, transforms the set of $\tilde{A} = \{b_1, b_2, \ldots, b_{m-1}, c_1, c_2, \ldots, c_t, r_{b_1}, r_{b_2}, \ldots, r_{b_e}, r_{c_1}, r_{c_2}, \ldots, r_{c_e}\}$ into permutate set $A = \{a_1, a_2, \ldots, a_n\}$ where $n = m + t + 2e - 1$, while the client needs to record the index of set B, C, R. Then, the client sends $\sigma_x = (A, g, P)$ to the cloud server.
- The server returns the result set $V = \{g^{a_1}, g^{a_2}, \ldots, g^{a_n}\}$ to the client.
- After receiving the result set V from the cloud server, the client needs to check the truth of the returned data. Firstly, the client needs to calculate S_{b_l} and S_{c_l} by the following formulas: $S_{b_l} = \prod g^{b_i} \cdot g^{r_{b_l}}$ and $S_{c_l} = \prod g^{c_i} \cdot g^{r_{c_l}}$. If $S_{b_l} = S_{c_l} = g^{y_l}$, the client concludes that the cloud server is honest. Then the client completes the exponentiations as: $g^{x_i} = \prod g^{b_l} \cdot g^{c_i}$, $1 \le i \le t$. Otherwise, the client concludes that the server is dishonest.

B. SECURE FIXED-EXPONENT MODULAR EXPONENTIATION OUTSOURCING ALGORITHM

This subsection focuses on securely computing fixed-exponent modular exponentiations by outsourcing them to cloud server, i.e., $({g_1}^x, {g_2}^x, \ldots, {g_t}^x) \bmod p$. Due to that the exponent x is public, what we should protect is the bases. To reach the purpose that the cloud server cannot get any information about the bases and the client can easily verify the correctness of the returned data, we process the bases into some other numbers which seem to be irrelevant to protect the privacy of bases. In the end, the bases $(g_1, g_2, \ldots, g_t)$ are translated into a set $A = \{a_1, a_2, \ldots, a_n\}$ which can be sent to cloud server.

The cloud server makes some computations based on the information (x, p, A) and returns the result to the client. According to the internal relation between set A and bases $\{g_1, g_2, \ldots, g_t\}$, the client can figure out ${g_i}^x$ and verify the correctness of the returned values with little cost. Next, the fixed-exponent base exponentiation scheme can be formally described as follows.

- Given $(g_1, x), (g_2, x), \ldots, (g_t, x)$ as input data to compute $({g_1}^x, {g_2}^x, \ldots, {g_t}^x)$, where $x \in Z_p$ and $g_i \in G_p$. The client first transforms the bases $(g_1, g_2, \ldots, g_t)$ into another set $A = \{a_1, a_2, \ldots, a_n\}$ where m elements of set A can be multiplied to g_i for each $i \in [1, t]$. The set A can be regarded as the combination of set B, C and R. Set B is composed of $m-1$ numbers which are randomly selected from G_p. Let $B = \{b_1, b_2, \ldots, b_{m-1}\}$. Set C is the set of base x_i divided by the multiplication of set B, then element c_i is $c_i = g_i / \prod_{j=1}^{m-1} b_j$, $1 \le i \le t$. The client needs to invoke the subroutine $Rand_2$ e times to get pairs (g_{y_i}, x). Similarly, the element r_i of set R can be generated by $r_{b_i} = y_i / \prod_{j=1}^{m-1} b_j$, $1 \le i \le e$ and $r_{c_k} = y_k / \prod_{j=1}^{t} c_j$, $1 \le k \le e$. Finally, the client, using a pseudo-random permutation function, transforms the set of $\tilde{A} = \{b_1, b_2, \ldots, b_{m-1}, c_1, c_2, \ldots, c_t, r_{b_1}, r_{b_2}, \ldots, r_{b_e}, r_{c_1}, r_{c_2}, \ldots, r_{c_e}\}$ into permutate set $A = \{a_1, a_2, \ldots, a_n\}$ where $n = m + t + 2e - 1$, while the client needs to record the index of set B, C, R. Then, the client sends $\sigma_x = (A, g, P)$ to the cloud server.
- The server returns the result set $V = \{{a_1}^x, {a_2}^x, \ldots, {a_t}^x\}$ to the client.
- After receiving the result set V from the cloud server, the client needs to check the truth of the returned data. Firstly, the client needs to calculate S_{b_l} and S_{c_l} by the following formulas: $S_{b_l} = \prod {b_i}^x \cdot {r_{b_i}}^x$ and $S_{c_l} = \prod {c_i}^x \cdot {r_{c_i}}^x$. If $S_{b_l} = S_{c_l} = {g_{y_i}}^x$, the client concludes that the cloud server is honest. Then the client can compute ${g_i}^x = \prod {b_l}^x \cdot {c_i}^x$, $1 \le i \le t$. Otherwise, the client concludes that the server is dishonest.

C. SECURITY ANALYSIS

Theorem 1: Fixed-base algorithm and fixed-exponent algorithm satisfy the the requirements of Correctness.

Proof: In the calculating part, the client can get the correct value $(g_1^x, g_2^x, \ldots, g_t^x)$ in fixed-base scheme and $(g_1^x, g_2^x, \ldots, g_t^x)$ in fixed-exponent scheme if the values returned by the cloud server are correct. The calculation

procedure is as follows:

$$S_{setB} = \prod_{i=1}^{m-1} g^{b_i} = g^{\sum_{i=1}^{m-1} b_i}, M_{setB} = (\prod_{i=1}^{m-1} b_i)^x,$$
$$S_k = S_k \cdot S_{setB} = g^{c_k + \sum_{i=1}^{m-1} b_i} = g^{x_k}, k \in [1, t],$$
$$M_k = M_k \cdot M_{setB} = \left(g_k \cdot \prod_{i=1}^{m-1} b_i\right)^x = g_k^x, k \in [1, t].$$

□

Theorem 2: The algorithm $(T_{fixed-base}, U)$ *is outsource-secure implementation, under the conditions that the inputs* $(x_1, x_2, \ldots, x_t, g)$ *are honest, secret; or honest, protected; or adversarial protected.*

Proof: In this section, we only focus on security. Let $A = (E, U')$ be a PPT adversary.

Fistly, we prove Pair One $EVIEW_{real} \sim EVIEW_{ideal}$:

The simulator S_1 behaves the same way as in the real execution under the input $(x_1, x_2, \ldots, x_t, g)$ anything other than honest, secret. If $(x_1, x_2, \ldots, x_t, g)$ are honest, secret inputs, then the simulator S_1 behaves as follows: S_1 makes n random queries of form $(x_1, x_2, \ldots, x_n, g)$ to U' when receiving the inputs in round i. S_1 tests two outputs from each program randomly. If an error is detected, S_1 checks the rest two outputs $(error, \varnothing, 1)$. If no error is detected, S_1 checks the rest two outputs. If all checks pass, S_1 outputs $(\varnothing, \varnothing, 0)$; Otherwise, S_1 selects a random element r and outputs $(r, \varnothing, 1)$. In other case, S_1 saves the appropriate states. In the real experiment, all three queries are computationally indistinguishable. In the ideal experiment, the inputs are chosen randomly and uniformly. Thus the input distributions to U' are computationally indistinguishable. In the ideal experiment, the inputs are chosen randomly and uniformly. Thus the input distributions to U' are computationally indistinguishable both in the real and ideal experiments. Next, it can be proved that $EVIEW^i_{real} \sim EVIEW^i_{ideal}$, which implies $EVIEW_{real} \sim EVIEW_{ideal}$.

Secondly, we prove Pair Two $UVIEW_{real} \sim UVIEW_{ideal}$:

The simulator S_2 behaves as follows: S_2 makes n random queries of form $(x_1, x_2, \ldots, x_n, g)$ to U' when receiving the inputs of round i. Then S_2 saves its states and the states of U'. Although, E can distinguish between these real and ideal experiments, he cannot communicate this information to U'. Thus, $UVIEW^i_{real} \sim UVIEW^i_{ideal}$ for each round i, proceed to the next step, and we have $UVIEW_{real} \sim UVIEW_{ideal}$. □

Theorem 3: The algorithm $(T_{fixed-exponent}, U)$ *is outsource-secure implementation, under the conditions that the inputs* $(g_1, g_2, \ldots, g_t, x)$ *are honest, secret; or honest, protected; or adversarial protected.*

Proof: In this section, we focus on the security of fixed-exponent algorithm. Because of the similarity between the two algorithms, we omit the specific proof process. The security proof can refer to the proof process of Theorem 2. □

Theorem 4: The checkability of fixed-base algorithm is $1 - \frac{C_t^2}{C_n^2} - \frac{2e}{C_n^{e+1}} \cdot \frac{1}{e+1}$, *if the server cheats the client.*

Proof: The checkability of a secure outsourcing scheme is that the malicious server cannot successfully persuade the client to trust that the returned values are exactly correct. In our schemes, the client has ability to verify the correctness of the result from cloud server, which prove the checkability of our schemes.

Then we will analyze the checkability of fixed-base scheme by probability. We split set A into B, C, R, however, what we care the returned data are $\prod g^{b_i} (1 \le i \le m)$, $\prod g^{c_i} (1 \le i \le t)$, $g^{r_{b_i}} (1 \le i \le e)$, $g^{r_{c_i}} (1 \le i \le e)$. Client needs to check the equations:

$$\prod_{i=1}^{m} g^{b_i} \cdot g^{r_{b_j}} = g^{y_j}, \quad \prod_{i=1}^{t} g^{c_i} \cdot g^{r_{c_j}} = g^{y_j}.$$

The server successfully cheats the client means that returned result pass the verification while it changes some right values. On the one hand, the server computes the correct values and chooses a random number w. Then the server returns the $w \cdot g^{b_i}$ or $w \cdot g^{c_i}$ and the corresponding $\frac{1}{w} \cdot g^{b_i}$ or $\frac{1}{w} \cdot g^{c_i}$. After that,

$$w \prod_{i=1}^{m} g^{b_i} \cdot \frac{1}{w} \cdot g^{r_{b_j}} = g^{y_j}, \quad w \prod_{i=1}^{t} g^{c_i} \cdot \frac{1}{w} \cdot g^{r_{c_j}} = g^{y_j}.$$

The probability that the server finds the right elements $(b_i, r_{b_1}, \ldots, r_{b_e})$ or $(c_i, r_{c_1}, \ldots, r_{c_e})$ is $\frac{m}{C_n^{e+1}} \cdot \frac{1}{e+1} + \frac{t}{C_n^{e+1}} \cdot \frac{1}{e+1}$. In this way, the probability that cloud server can cheat the client is $\frac{2e}{C_n^{e+1}} \cdot \frac{1}{e+1}$. Compared with another circumstance, the probability of this condition is much less. On the other hand, the server chooses a random number l and two elements c_i, c_j which belong to set C. Then the server returns g^{c_i+l}, g^{c_j-l} to the client. In this way, the probability is $\frac{C_t^2}{C_n^2}$.

Thus the checkability of client is $1 - \frac{C_t^2}{C_n^2} - \frac{2e}{C_n^{e+1}} \cdot \frac{1}{e+1}$. To improve the efficiency of our schemes, we suggest that e should be much smaller than t and m. In our experiment, we set $e = 5$ while $m = 100$. After the values of t and e have been determined, we can find that the checkability increases with the increase of n, but efficiency is getting lower and lower. Considering that users have different requirements for efficiency and checkability, users can adjust the value of n to balance checkability or efficiency according to their own circumstances. Finally, the user can choose a suitable n to determine all parameters in this scheme.

□

Theorem 5: The checkability of fixed-exponent algorithm is $1 - \frac{C_t^2}{C_n^2} - \frac{2e}{C_n^{e+1}} \cdot \frac{1}{e+1}$, *if the server cheats the client.*

Proof: Due to the similarity between the two algorithms, the specific proof of fixed-exponent algorithm can refer to the proof in Theorem 4. □

D. COMPLEXITY ANALYSIS

Compared with previous secure outsourcing modular exponentiation schemes, the advantages of our schemes in efficiency must be emphasized. Above all, our schemes do not need any complicated precomputations, which is more efficient than the existing schemes in [3], [6]. In addition, we will

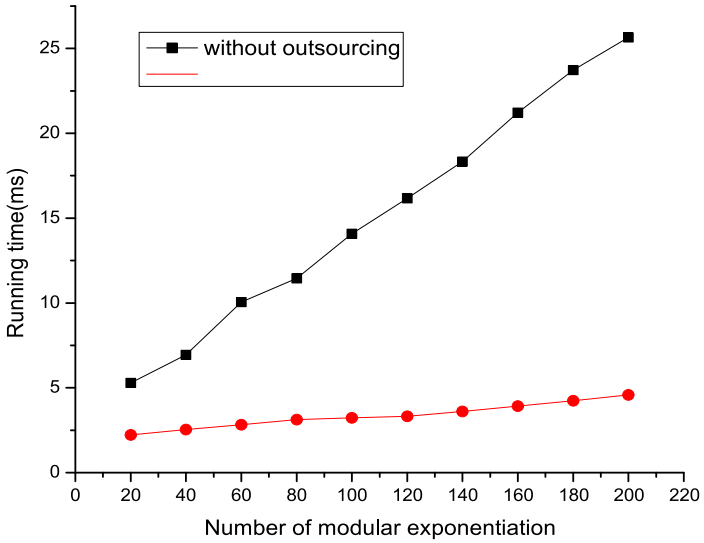

FIGURE 2. Comparison of time cost.

analyze the efficiency in computation, communication and storage complexity.

1) COMPUTATION COMPLEXITY

We compare the proposed schemes with the algorithms in Ma *et al.* [2], Chen *et al.* [6], Ding *et al.* [26], and Ren *et al.* [19] on outsourcing modular exponentiations. We denote by MM a modular multiplication and by MInv a modular inverse. We omit other operations such as modular additions in these algorithms. The comparison of the efficiency and the checkability of the algorithms are shown in Table 1.

Compared with the scheme of Ma *et al.* [2], our scheme is almost the same in terms of MM, MInv and Invoke (Server) than the other schemes. However, we invoke *rand* for e times and we use only one cloud server while Ma *et al.* [2] is based on the two severs model. What's more, the minimum checkability of our scheme is better than Ma *et al.* [2]. As for Chen *et al.* [6], our scheme is more excellent in both efficiency and checkability. In addition, our scheme is performed in single cloud server, which can save a lot expenses for client. Based on Ma *et al.* [2], Ding *et al.* [26] made improvements in checkability and the number of cloud server. However, they put part of modular exponentiations to the client, which is a burden for the client. On the contrary, our schemes are clearly superior to Ding *et al.* [26] in terms of the times of the invoking server while the client doesn't need to do any complicated computations. Although the checkability of Ren *et al.* [19] has reached 1, it seems to perform better than ours in the above tables. However, our schemes, considering MM and MInv, are more efficient. What's more, our schemes only need one cloud server.

We implement our experiment using C++ language and the GNU multiple precision arithmetic library (GMP). The experiment has been carried out on a LINUX machine with Intel Core i3-3320 processors running at 3.30GHz and 4G memory. The time cost is shown in Fig.1.

In our experiment, we randomly generate 180 random numbers in the range of 0 to 512 bit and invoke *Rand* 5 times. We also randomly select a 100-bit g. As shown in Fig.2, when computing $(g^{x_1}, g^{x_2}, \ldots, g^{x_t})$, the time cost of our outsourcing scheme is far less than without outsourcing. As the number of modular exponentiations increases, the rate of using time of the without outsourcing scheme grows clearly faster than the outsourcing scheme. The larger amounts of modular exponentiations need to be computed, the more resource and energy a device can be saved.

2) STORAGE COMPLEXITY

In fixed-base algorithm, the client needs to compute S_{setB} and S_{setC} and the g^{y_i}. Thus the client has to store $m + t + 2e - 1$ indexes, which is same as fixed-exponent algorithm. Therefore, the storage complexity is $m + t + 2e - 1$ elements in every scheme.

3) COMMUNICATION COMPLEXITY

In our schemes, the client sends set A to the cloud server. In fixed-base algorithm, the number of rounds between the client and cloud server is only one. Due to the uncertainty of the specific size of the number, we substitute the number of elements in communication set for communication complexity. Combined with the process of schemes, the communication complexity of fixed-base algorithm is $2n$, which is same as fixed-exponent algorithm.

V. SECURE PAILLIER ENCRYPTION OUTSOURCING SCHEME

Paillier encryption system [31] is a widely-used public key encryption scheme [25], [28], [30], which includes three algorithms as follows.

TABLE 1. Comparison of efficiency and checkability.

t Exps	[2]	[6]	[26]	[19]	Ours
MM	$m+t-2$	$7t$	$7t+9+1.5logrt_1t_2$	$13t$	$m+2t+2r-2$
MInv	0	$3t$	6	$3t$	0
Invoke ($Rand$)	0	5	6	6	e
Invoke (Server)	2	$2t$	$2n+2$	$4n$	1
Checkability	$\frac{3}{4}$	$\frac{2}{3}$	$1-\frac{n^2}{10(4n^2+6n+2)}$	1	$1-\frac{C_t^2}{C_n^2}-\frac{2e}{C_n^{r+1}}\cdot\frac{1}{r+1}$
Number of cloud server	Two	Two	One	Two	One

1) **Key generation.** Firstly, Two big enough primes p and q are selected. Then, set $\lambda = lcm(p-1, q-1)$ as the secret key sk. The public key pk is (n, g), where $n = pq$ and $g \in \mathbb{Z}_{n^2}^*$ so that $gcd(L(g^\lambda \bmod n^2), n) = 1$. Here, $L(x) = (x-1)/n$.
2) **Encryption.** Select a number m_0 from plaintext space $\mathbb{Z}_n$. Let a random $r \in \mathbb{Z}_n^*$ be the secret parameter, then the ciphertext c_0 of m_0 is $c_0 = g^{m_0} r^n \bmod n^2$.
3) **Decryption.** Make $c_0 \in \mathbb{Z}_{n^2}$ be a ciphertext. The plaintext hidden in c_0 is $m_0 = \frac{L(c_0^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$.

As we can see, both **Encryption** and **Decryption** algorithms contain fixed-base and fixed-exponent modular exponentiations. While we utilize Paillier encryption system to encryption a big number of values, it will cost much computation time, especially for resource-constrained clients. Thus, it is necessary to accelerate the **Encryption** and **Decryption** algorithms of Paillier encryption system by employing the strong computation ability of cloud. Here, we will propose a secure scheme for outsourcing **Encryption** and **Decryption** of Paillier algorithm based on our Protocols 1 and 2. We assume that the client want to encrypt or decrypt t plaintexts or ciphertexts simultaneously. Our general idea are as follows.

Based on Paillier algorithm, the parameter λ is $\lambda = lcm(p-1, q-1)$ where lcm means least common multiple. The logarithmic function $L(x)$ can be described as $L(x) = (x-1)/n$. According to the encryption formula $c = g^m r^n \bmod n^2$, the problem we want to solve is how to compute g^m while the random number r has been selected. Because what we concentrate on is encrypting t plaintexts simultaneously, the protocol 1 is right to compute g^{m_i}. Due to that the base g is public key, what we should protect in the process of outsourcing is the plaintext m_i. To reach the purpose that the cloud server cannot get any information about the plaintexts and the client can easily verify the correctness of the returned data, we divide the plaintexts into some other numbers which seem to be irrelevant to protect the privacy of plaintexts. In the end, the plaintexts $\{m_1, m_2, \ldots, m_t\}$ are translated into a set $A = \{a_1, a_2, \ldots, a_n\}$ which can be sent to cloud server. The cloud server does some computations based on the information (g, n, A) and returns the result to the client. According to the internal relation between set A and plaintexts $\{m_1, m_2, \ldots, m_t\}$, the client can figure out g^{m_i} and verify the correctness of the returned results with little cost.

In the process of decryption, the client should have decrypted by the formula $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$. We find that the main problem we need to solve is how to compute $c^\lambda \bmod n^2$ while $g^\lambda \bmod n^2$ is a fixed value. Because what we concentrate on is decrypting t ciphertexts simultaneously, the protocol 2 is right to compute ${c_i}^\lambda$. In order to avoid the cloud server learning anything about the process of decryption, what we should protect in this process of outsourcing is the ciphertext c_i while the exponent λ is public. To reach the purpose that the cloud server cannot get any information about the ciphertexts and the client can easily verify the correctness of the returned data, we divide the ciphertexts into some other numbers which seem to be irrelevant to protect the privacy of ciphertexts. In the end, the ciphertexts $\{c_1, c_2, \ldots, c_t\}$ are translated into a set $H_A = \{a_1, a_2, \ldots, a_n\}$ which can be sent to cloud server. The cloud server does some computations based on the information (λ, n, H_A) and returns the result to the client. According to the internal relation between set H_A and ciphertexts $\{c_1, c_2, \ldots, c_t\}$, the client can figure out ${c_i}^\lambda$ and verify the correctness of the returned results with little cost. Next, secure outsourcing of Paillier algorithm can be formally described as follows.

- **Key generation** Let p, q be large primes, $n = pq$. $g \in (Z/n^2Z)^*$ and $gcd(L(g^\lambda \bmod n^2), n) = 1$, public-Key is $\{n, g\}$, Secret-Key is $\{p, q\}$.
- **Encryption** On input the public key g, n, a message set $M = \{m_1, m_2, \ldots, m_t\}$ and a random number $r (r < n)$, the client first conceals the plaintext set M by dividing m_i in a manner that m elements of set $A = \{a_1, a_2, \ldots, a_n\}$ can be summed to m_i. In addition, set A is composed of set B, C and R. Then the client needs to randomly choose $m-1$ numbers from Z_n, which comprise set $B = \{b_1, b_2, \ldots, b_{m-1}\}$. Thus, the set C can be generated by the function $sc_i = m_i - \sum b_i$. Then the client needs to invoke $rand_1$ e times to get pairs (g, g^{y_i}). The set R can be generated by the formulas $r_{B_i} = y_i - \sum b_i, 1 \le i \le e$ and $r_{C_i} = y_i - \sum sc_i, 1 \le i \le e$. Next, the client sends g, n and set A to cloud server. After receiving the result set $V = \{g^{a_1}, g^{a_2}, \ldots, g^{a_n}\}$ from the cloud server, the client needs to check the truth of the results. Firstly, the client needs to calculate S_{b_l} and S_{c_l} by the following formulas: $S_{b_l} = \prod g^{b_i} \cdot g^{r_{b_l}}$ and $S_{c_l} = \prod g^{sc_i} \cdot g^{r_{c_l}}$. If $S_{b_l} = S_{c_l} = g^{y_l}$, the client concludes that the cloud server is honest. Then the client can compute $g^{x_i} = \prod g^{b_i} \cdot g^{sc_i}, 1 \le i \le t$. Otherwise, the client concludes

that the server is dishonest. In the end, the client can compute the ciphertext by $c_i = g^{m_i} r^n \bmod n^2$.

- **Decryption** Firstly, the client needs to hide the ciphertext set $\{c_1, c_2, \ldots, c_t\}$ by dividing c_i in a manner that m elements of set $H_A = \{a_1, a_2, \ldots, a_n\}$ can be multiplied to c_i. In addition, the set H_A can be composed of set H_B, H_C and H_R. The client randomly selects $m-1$ numbers from G_n, which make up $H_B = \{h_{b_1}, h_{b_2}, \ldots, h_{b_{m-1}}\}$. Therefore, set H_C can be generated by the formula $h_{c_i} = c_i / \prod h_{b_i}$. Then the client needs to invoke $rand_2$ e times to get pairs $(y_i, {y_i}^{\lambda})$. The set H_R can be generated by the formulas $hb_i = y_i / \prod h_{b_i}, 1 \le i \le e$ and $hc_i = y_i / \prod h_{c_i}, 1 \le i \le e$. Next, the client sends λ, n and set H_A to the cloud server. After receiving the result set $H_V = \{{a_1}^{\lambda}, {a_2}^{\lambda}, \ldots, {a_n}^{\lambda}\}$ from the cloud server, the client needs to check the truth of the results. Firstly, the client needs to calculate S_{b_l} and S_{c_l} by the following formulas: $S_{b_l} = \prod {b_i}^x \cdot {r_{b_i}}^x$ and $S_{c_l} = \prod {c_i}^x \cdot {r_{c_i}}^x$. If $S_{b_l} = S_{c_l} = {g_{y_i}}^x$, the client concludes that the cloud server is honest. Then the client can compute ${g_i}^x = \prod {b_l}^x \cdot {c_i}^x, 1 \le i \le t$. Otherwise, the client concludes that the server is dishonest. In the end, the client can compute the plaintext by $m_i = \frac{L(c_i^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n$.

The proposed algorithms can be used in both encryption and decryption. By using our schemes, we securely outsource the modular exponentiations that are burden for resource-constrained devices to cloud, which strikingly reduces the cost of time in encryption and decryption. Therefore, the efficiency of paillier algorithm is exceedingly improved. For space limitation, we will provide analysis detail of secure Paillier algorithm outsourcing scheme in extended version.

VI. CONCLUSION

In this paper, we presented two secure outsourcing schemes for outsourcing modular exponentiations to single cloud server, which are suitable for fixed-base modular exponentiation and fixed-exponent modular exponentiation, respectively. Based on secure modular exponentiation outsourcing schemes, we then proposed a secure outsourcing of the Paillier algorithm. Additionally, we theoretically analyzed our overheads and leveraged simulation experiments to evaluate our proposed schemes, which shows our solutions can achieve high efficiency.

REFERENCES

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.
[2] X. Ma, J. Li, and F. Zhang, "Outsourcing computation of modular exponentiations in cloud computing," *Cluster Comput.*, vol. 16, no. 4, pp. 787–796, Apr. 2013.
[3] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. 2nd Theory Cryptogr. Conf.*, Cambridge, MA, USA, 2005, pp. 264–282.
[4] K. M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. Annu. Cryptol. Conf.*, Berlin, Germany, 2010, pp. 483–501.
[5] M. Barbosa and P. Farshim, "Delegatable homomorphic encryption with applications to secure outsourcing of computation," in *Proc. Cryptographers' Track at RSA Conf.*, Berlin, Germany, 2012, pp. 296–312.
[6] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.
[7] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2010, pp. 48–59.
[8] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. 6th Annu. Conf. Privacy, Secur. Trust*, Oct. 2008, pp. 240–245.
[9] B. Applebaum, Y. Ishai, and E. Kushilevitz, "From secrecy to soundness: Efficient verification via secure computation," in *Proc. Int. Colloq. Automata, Lang., Program.*, Berlin, Germany, 2010, pp. 152–163.
[10] B. Chevallier-Mames, J.-S. Coron, bastien, N. Mccullagh, D. Naccache, M. Scott, "Secure delegation of elliptic-curve pairing," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, Berlin, Germany, 2010, pp. 24–35.
[11] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. Theory Cryptogr. Conf.*, Berlin, Germany, 2012, pp. 422–439.
[12] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Cryptol. Conf.*, Berlin, Germany, 2010, pp. 465–482.
[13] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. M. Chow, Z. Liu, and X. Tan, "Securely outsourcing exponentiations with single untrusted program for cloud storage," in *Proc. Eur. Symp. Res. Comput. Secur.*, Cham, Switzerland, 2014, pp. 326–343.
[14] E. Horowitz and S. Sahni, "Computing partitions with applications to the knapsack problem," *J. ACM*, vol. 21, no. 2, pp. 277–292, Apr. 1974.
[15] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany, 1998, pp. 221–235.
[16] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *Comput. Complex.*, vol. 2, no. 2, pp. 111–128, Jun. 1992.
[17] M. J. Coster, B. A. Lamacchia, A. M. Odlyzko, and C. P. Schnorr, "An improved low-density subset sum algorithm," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, Berlin, Germany, 1991, pp. 54–67.
[18] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, nos. 1–3, pp. 181–199, Aug. 1994.
[19] Y. Ren, N. Ding, X. Zhang, H. Lu, and D. Gu, "Verifiable outsourcing algorithms for modular exponentiations with improved checkability," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, 2016, pp. 293–303.
[20] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.
[21] J. Ye, X. Chen, and J. Ma, "An improved algorithm for secure outsourcing of modular exponentiations," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2015, pp. 73–76.
[22] P. Golle and I. Mironov, "Uncheatable distributed computations," in *Proc. Cryptographers' Track at RSA Conf.*, Berlin, Germany, 2001, pp. 425–440.
[23] L. Zhao, M. Zhang, H. Shen, Y. Zhang, and J. Shen, "Privacy-preserving outsourcing schemes of modular exponentiations using single untrusted cloud server," *Ksii Trans. Internet Inf. Syst.*, vol. 11, no. 2, pp. 826–845, 2017.
[24] J. Cai, Y. Ren, and T. Jiang, "Verifiable outsourcing computation of modular exponentiations with single server," *IJ Netw. Secur.*, vol. 19, no. 3, pp. 449–457, 2017.
[25] Y. Zhu, X. Li, J. Wang, and J. Li, "Cloud-assisted secure biometric identification with sub-linear search efficiency," *Soft Comput.*, vol. 24, no. 8, pp. 5885–5896, Apr. 2020.
[26] Y. Ding, Z. Xu, J. Ye, and K. K. R. Choo, "Secure outsourcing of modular exponentiations under single untrusted programme model," *J. Comput. Syst. Sci.*, vol. 90, pp. 1–13, Dec. 2017.
[27] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.
[28] Y. Zhu and X. Li, "Privacy-preserving k-means clustering with local synchronization in peer-to-peer networks," *Peer–Peer Netw. Appl.*, vol. 13, pp. 2272–2284, Mar. 2020.

[29] Q. Xue, Y. Zhu, and J. Wang, "Joint distribution estimation and Naïve Bayes classification under local differential privacy," *IEEE Trans. Emerg. Topics Comput.*, early access, Dec. 13, 2019, doi: 10.1109/TETC.2019.2959581.
[30] X. Li, Y. Zhu, and J. Wang, "Highly efficient privacy preserving location-based services with enhanced one-round blind filter," *IEEE Trans. Emerg. Topics Comput.*, early access, Jul. 5, 2019, doi: 10.1109/TETC.2019.2926385.
[31] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany, 1999, pp. 223–238.

DEZHI AN is currently a Professor with the School of Cyber Security, Gansu University of Political Science and Law, Lanzhou, China. His research interests include network security, public opinion analysis, and data mining.

YAN LI is currently an Associate Professor with the School of Cyber Security, Gansu University of Political Science and Law, China. She has published nearly 20 refereed journal and conference papers in these areas. Her research interests include data processing, multimedia security, computer vision, and information content security.

SHENGCAI ZHANG received the master's degree in signal and information processing from the Lanzhou University of Technology, Lanzhou, China, in 2009, where he is currently pursuing the Ph.D. degree in control theory and control engineering. He is also an Associate Professor with the School of Cyber Security, Gansu University of Political Science and Law. His research interests include information security, artificial intelligence, and intelligent optimization.

JUN LU is currently an Associate Professor with the School of Cyber Security, Gansu University of Political Science and Law. His research interests include computer network security, computational intelligence, and data mining.