Cluster Comput (2013) 16:787–796
DOI 10.1007/s10586-013-0252-0

Outsourcing computation of modular exponentiations in cloud computing

Xu Ma · Jin Li · Fangguo Zhang

Received: 12 December 2012 / Accepted: 14 March 2013 / Published online: 10 April 2013

Abstract Cloud computing is an emerging computing paradigm in which IT resources and capacities are provided as services over the Internet. Promising as it is, this paradigm also brings forth new challenges for security when users want to securely outsource the computation of cryptographic operations to the untrusted cloud servers. As we know, modular exponentiation is one of the basic operations among most of current cryptosystems. In this paper, we present the generic secure outsourcing schemes enabling users to securely outsource the computations of exponentiations to the untrusted cloud servers. With our techniques, a batch of exponentiations (e.g. t exponentiations) can be efficiently computed by the user with only $O(n + t)$ multiplications, where n is the number of bits of the exponent. Compared with the state-of-the-art algorithm, the proposed schemes are superior in both efficiency and verifiability. Furthermore, there are not any complicated pre-computations on the user side. Finally, the schemes are proved to be secure under the Subset Sum Problem.

Keywords Modular exponentiation · Outsourcing computation · Security and privacy · Cloud computing

X. Ma · F. Zhang (✉)
School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, P.R. China
e-mail: isszhfg@mail.sysu.edu.cn

X. Ma
e-mail: xumasysu@gmail.com

J. Li
Department of Computer Science, Guangzhou University, Guangzhou 510006, P.R. China
e-mail: lijin@gzhu.edu.cn

1 Introduction

Cloud computing [2, 3, 10] is a promising next-generation computing paradigm which primarily relies on technologies such as virtualization, utility computing, Service Oriented Architecture, and so forth. Cloud computing is capable to provide seemly unlimited "virtualized" resources to users as services across the Internet while hiding platform and implementation details from users. The services are invoked by users in a pay-per-use manner. With this paradigm computation/storage intensive tasks can be performed even by resource-constrained users through being outsourced to resource-abundant cloud servers. As compared to setting up their own infrastructures, cloud users can tremendously save their capital expenditures via the usage of cloud computing, not to mention other benefits such as reliability and high scalability of the system.

Promising as it is, this paradigm also brings forth new challenges and security concerns when users want to securely outsource the computation of cryptographic operations to the untrusted cloud servers. Firstly, the servers are not fully trusted and sometimes the computations outsourced to the cloud are so critical that it is imperative to rule out accidental errors during the outsourcing computation process. Therefore, the basic security requirements of outsourcing computation are verifiability and efficiency, which require that the client should be able to verify the correctness of the values returned by the worker, and the verification process should require substantially less computation efforts than doing the computation from scratch. Sometimes the input and output of the computation task contain sensitive information of the client, such as the client's private key or medical records, therefore, it is desirable to protect the secrecy of the client's input and output, which implies the security requirement of privacy of outsourcing computation.

The problem of secure outsourcing expensive computations has been well studied in the cryptography community. Matsumoto [26] firstly introduced the idea of speeding up secret computations using insecure auxiliary devices. Chaum and Pedersen [11] presented "wallets with observers" that allows a piece of hardware installed on the client's device to carry out some computation for each transaction. Golle and Mironov [19] introduced the concept of ringers to elegantly solved the problem of verifying computation completion for the "inversion of one-way function" class of outsourcing computation. In recent years, the outsourcing computation becomes the hot topic of research in the computer theory community and cryptography community due to the advancement of cloud computing, and many beautiful works have been done in this area, such as [6, 7, 13, 15].

As we know, modular exponentiation is one of the basic operations among most of current cryptosystems, such as DSS, RSA, and ElGmal et al. For an n-bit exponent a, the traditional square-and-multiply method requires $1.5n$ modular multiplications on average. Thus, it is a very time-consuming operation for limited computation resources. The applications of fast exponentiation include speeding up signature verification of the above mentioned schemes. For these scheme, the online generation of message signature can be done quickly because the part of the signature that requires exponentiation can be precomputed. However, the verification of the signature needs online exponentiation because the message signature is used in the exponentiation.

In this paper, we mainly focus on how to efficiently and securely outsource modular exponentiation computation to cloud servers.

Contributions Our contributions on outsourcing computation of modular exponentiation is multi-fold.

Firstly, we propose two secure modular exponentiation outsourcing schemes without any complicated computations utilizing two untrusted servers. One is fixed base-variable exponent exponentiation outsourcing and the other is fixed exponent-variable base exponentiation outsourcing. The variable exponent or variable base means that the exponent or the base is kept privacy against the cloud sever in the respective schemes. With the help of cloud servers, t exponentiations can be computed by the user within only $O(n+t)$ multiplications, where n is the number of bits of the exponent. Compared with [12, 21, 23], our schemes are more efficient. And the computational savings will increase when the batch size grows bigger.

Secondly, the first two schemes are provable secure and the security can be reduced to the Subset Sum Problem, which has been proven to be NP-complete problem. In addition, the schemes achieve approximately $3/4$ error detection probability, which means that if the cloud server returns a sequence of values that contains wrong ones, the client can detect it with probability at least $3/4$, which is greater than previous secure outsourcing computation schemes of modular exponentiations.

Finally but more importantly, in the third scheme, we present a new method to achieve the verifiability so that the scheme only needs one server. The model of the scheme is more practical and the security analysis shows that the error detection probability can reach approximately 1.

Related works In [15], Gennaro et al. showed how to delegate arbitrary computations by increasing the client's offline complexity and public-key size based on fully homomorphic encryption [16] an Yao's garbled circuit. In [13], Chung et al. proposed an improved generic outsourcing computation protocol without utilizing garbled circuit, whereas the client needs to pre-compute some values to verify the result. In [6], Barbosa and Farshim gave a modular construction of delegatable homomorphic encryption from fully homomorphic encryption, functional encryption and MAC, and showed how one can build a secure outsouring computation scheme generically from delegatable homomorphic encryption. However, the use of fully homomorphic encryption resulting in protocols of limited practical relevance.

In theoretical community, researchers have devoted considerable attention to the outsourcing computation of arbitrary functions. Indeed, after computing the delegated function f on input x and sending the result y, the server can use various types of proof systems to convince the client the correctness of the result. These works contain interactive proofs [5, 17], efficient arguments based on probabilistically checkable proofs [24, 25], CS proofs [27] and the muggles proofs [18]. However, utilizing proofs systems in outsourcing computation protocols can merely realize the security requirement of verifiability. The security requirement of input and output privacy cannot be satisfied solely by the proof systems.

For the outsourcing computation of specific functions, plenty of research works have been proposed. Benjamin and Atallah [4, 8] addressed the problem of secure outsourcing for widely applicable linear algebra computations. However, the proposed protocols required the expensive operations of homomorphic encryptions. Atallah and Frikken [1] further studied this problem and gave improved protocols based on the so-called weak secret hiding assumption. Benabbas et al. [7] presented the first practical outsourcing computation scheme for high degree polynomial functions based on the approach of [15]. Papamanthou et al. [28] further studied the problem and proposed a publicly verifiable outsourcing computation scheme. In 2011, Green et al. [20] proposed new methods for efficiently and securely outsourcing decryption of attribute-based encryption (ABE) ciphertexts. Based on this work, Parno et al. [29] showed a construction of a multi-function computation delegation scheme. In

2012, Waters [30] proposed an outsourcing computation for attribute-based encryption where the ciphertext update can be efficiently securely outsourced to the server.

Many works have been done towards speeding up the modular exponentiation computation by utilizing untrusted servers. In [23], Jakobsson showed how to generate secure server-aided signature by outsourcing the exponentiation computation task to the untrusted servers who may all be controlled by one and the same adversary. To blind the exponent and ensure the verifiability of the result. The protocol in [23] includes the algorithm of problem transformation which consists of replication, dependency, blinding and permutation. However, the user has to precompute a set of modular exponentiations in order to recover the result and check its correctness, which is what we want to perform at first place. And parameters have to be chosen to balance the efficiency and security requirements. Therefore, [23] is efficient for small batch size. Hohenberger and Lysyanskaya [21] presented the outsource-secure algorithm for modular exponentiation by utilizing two untrusted servers. In [12], Chen et al. gave a new algorithm for outsourcing computation of modular exponentiation, and the algorithm is superior in both efficiency and verifiability compared with [21]. Although both [21] and [12] ensure the exponent privacy and base privacy simultaneously, the protocols still require complicated precomputations (modular exponentiations) and the probabilities that the malicious server is detected are only $\frac{1}{2}$ and $\frac{2}{3}$, respectively.

Organization The rest of the paper is organized as follows: In Sect. 2, we present the system model and security definitions of our schemes. In Sect. 3, three schemes of secure outsourcing computation schemes of modular exponentiations are proposed. The security and complexity analysis in specified in Sect. 4. And Sect. 5 concludes the paper.

2 System model and security definitions

The system model and security definitions of our schemes are formally specified in this section.

2.1 System model

An outsourcing computation scheme is a two-party protocol between a client C a server S. The client chooses a function and an input which he provides to the server. Note that the input is always blinded into x' before sending it to the sever considering abut the input privacy. The server is expected to evaluate the function on the input and respond with the output together with a proof that the result is correct. The client then verifies that the output provided by the server is indeed the output of the function computed on the input provided. In the following description, the computation task is denoted as given f and an input x to compute $f(x)$. Unlike the previous outsourcing computation protocols [13, 15], since no public key operations are needed in our schemes, the $\mathsf{KenGen}(\cdot)$ algorithm is excluded in our scheme, and the random numbers used in our scheme are generated on the fly. Formally, our scheme consists of three algorithms, which are described as follows.

- $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau, f)$. The problem generation algorithm uses the secret input τ to encode the input x as a public value σ_x, and sends σ_x to the server S.
- $\sigma_y \leftarrow \mathsf{Compute}(f, \sigma_x)$. Given σ_x and the outsourcing function f, the server works out an encoded output σ_y and returns it to the client.
- $y \cup \perp \leftarrow \mathsf{Verify}(\sigma_y, \tau)$. Using the secret input τ, the verification algorithm converts the server's encoded output into the output of the function, e.g. $y = f(x)$ or outputs $\perp$ indicating that σ_y does not represent the valid output of f on x.

2.2 Security definitions

An outsourcing computation scheme should be correct, verifiable and private. In the following description, we will give the formal definition of these security definitions, respectively. Intuitively, an outsourcing scheme is correct if the problem generation algorithm produces values that allow an honest cloud server to compute values that will verify successfully and correspond to the function $f(\cdot)$ on those inputs.

Definition 1 (Correctness) An outsourcing computation scheme is correct if for the outsourcing function f and input x, satisfy that if $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau)$ and $\sigma_y \leftarrow \mathsf{Compute}(f, \sigma_x)$, then $y \leftarrow \mathsf{Verify}(\sigma_y, \tau)$.

An outsourcing computation scheme is secure if it satisfies the security definition of verifiability, which means that a malicious sever can not persuade the client to accept an incorrect output. We formalize this intuition with the following experiment.

Experiment $\mathbf{Exp}_A^{verif}[f, \kappa]$

Query and response:

- For $i = 1, \ldots, l = poly(\kappa)$, $x_i \leftarrow A(x_1, \sigma_{x_1}, \beta_1, \ldots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$.
- $\sigma_{x_i} \leftarrow \mathsf{ProbGen}(x_i, \tau_i)$.
- $\sigma_{y_i} \leftarrow A(x_1, \sigma_{x_1}, \beta_1, \ldots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{x_{i-1}}, \sigma_{x_i})$.
- $\beta_i = \mathsf{Verify}(\tau_i, \sigma_{y_i})$.

Challenge:

- $x \leftarrow A(x_1, \sigma_{x_1}, \beta_1, \ldots, x_l, \sigma_{x_l}, \beta_l)$.
- $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau)$.

– $\sigma_y \leftarrow A(x_1, \sigma_{x_1}, \beta_1, \ldots, x_l, \sigma_{x_l}, \beta_l, \sigma_x)$.
– $\hat{y} \leftarrow \mathsf{Verify}(\tau, \sigma_y)$.
– if $\hat{y} \neq f(x)$ and $\hat{y} \neq \perp$, output 1, else 0.

In the query phase, the malicious servers are given oracle access to generate the encoding of multiple problem instances, and also oracle access to the result of the verification algorithm on arbitrary strings on those instances. The adversary succeeds if he convince the client to output wrong result for a given input value. Our goal is to make the adversary succeed only with negligible probability.

Definition 2 (Verifiability) For an outsourcing computation scheme, we define the advantage of an adversary $\mathcal{A}$ in the experiment above as:

$$Adv_A^{verif}[f, \kappa] = Prob\big[\mathbf{Exp}_A^{verif}[f, \kappa] = 1\big].$$

An outsourcing computation scheme satisfies the security of verifiability if for any adversary A running in probabilistic polynomial time

$$Adv_A^{verif}[f, \kappa] < neg(\kappa)$$

where $neg(\kappa)$ is a negligible function of the input.

The security definition of privacy contains input privacy and output privacy. Below, we define the input privacy based on a typical indistinguishability argument that guarantees no information about the inputs is leaked. The security definition of output privacy can be described similarly, however, we omit the definition owning to that the output results are public values in our scheme. For more information, please refer to [15].

Intuitively, an outsourcing computation scheme is input private when the outputs of the problem generation algorithm $\mathsf{ProbGen}$ over two different inputs are indistinguishable. Formally, we define the input privacy with the following experiment.

Experiment $\mathbf{Exp}_A^{Ipriv}[f, \kappa]$

Query and response:

– For $i = 1, \ldots, l = poly(\kappa)$, $x_i \leftarrow A(x_1, \sigma_{x_1}, \ldots, x_{i-1}, \sigma_{x_{i-1}})$.
– $\sigma_{x_i} \leftarrow \mathsf{ProbGen}(x_i, \tau_i)$.

Challenge:

– $(x_{c0}, x_{c1}) \leftarrow A(x_1, \sigma_{x_1}, \ldots, x_l, \sigma_{x_l})$.
– The client randomly selects a bit $b \leftarrow_R \{0, 1\}$ and sends $\sigma_{x_{cb}} \leftarrow \mathsf{ProbGen}(x_{cb}, \tau)$ to the adversary.
– $\hat{b} \leftarrow A(x_1, \sigma_{x_1}, \ldots, x_l, \sigma_{x_l}, x_c, \sigma_{x_c})$.
– If $\hat{b} = b$, output1, else 0.

In the above experiment, the adversary is given the oracle access of the problem generation algorithm in the query and challenge phase, the adversary succeeds if he can distinguish the output of the problem generation algorithm in the challenge phase.

Definition 3 (Input privacy) For an outsourcing computation scheme, we define the advantage of an adversary in the above experiment as

$$Adv_A^{Ipriv}[f, \kappa] = Prob\big[\mathbf{Exp}_A^{Ipriv}[f, \kappa] = 1\big] - \frac{1}{2}.$$

An outsourcing computation scheme is input private if for any adversary A running in probabilistic polynomial time

$$Adv_A^{Ipriv}[f, \kappa] < neg(\cdot)$$

where $neg(\cdot)$ is a negligible function of its input.

In this paper, we use a weak input privacy definition considering the applications of our schemes. As we know, in the cryptosystem, the exponents are always the private keys of the user. Thus, the adversary is not able to query the problem generation algorithm arbitrarily in the real world. Therefore, the security definition of input privacy can be formalized as follows.

Definition 4 (Weak input privacy) Let T be a computational task and $l(\cdot)$ an arbitrary function. We say that the outsourcing of T is ϵ-private with respect to l if the adversary has only negligible advantage ϵ in computing $l(i)$ for some private input i if performing the outsourcing work and seeing the public input and output of the user, compared to a setting where he only sees the public input and output of the user. In some papers, the privacy security requirement also expressed as input privacy.

3 Construction

In this paper, we propose three generic constructions for secure outsourcing of exponentiations. The first two protocols needs two non-collusion servers, while the third only requires one server. All the computations are executed within the cyclic group G_p, where p is a secure prime, and g is the generator of G_p. The schemes consist of three subprotocols: **ProbGen**, **Compute**, and **Verify**.

3.1 Protocol 1. Fixed base-variable exponent exponentiation secure outsource

As described in Fig. 1, the fixed base-variable exponent scheme works as follows:

– $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau, f)$. Given $(g, x_1), (g, x_2), \ldots, (g, x_t)$ as input that corresponding to an implicit request to compute $(g^{x_1}, g^{x_2}, \ldots, g^{x_t})$, where $x_i \in \mathbf{Z}_p$, $i \in [1, t]$, U first blinds the exponent vector $(x_1, x_2, \ldots, x_t)$ by generating a set of n pseudo-random numbers $A = \{a_1, a_2, \ldots, a_n\}$ in a manner that m of which can be summed to x_i, for

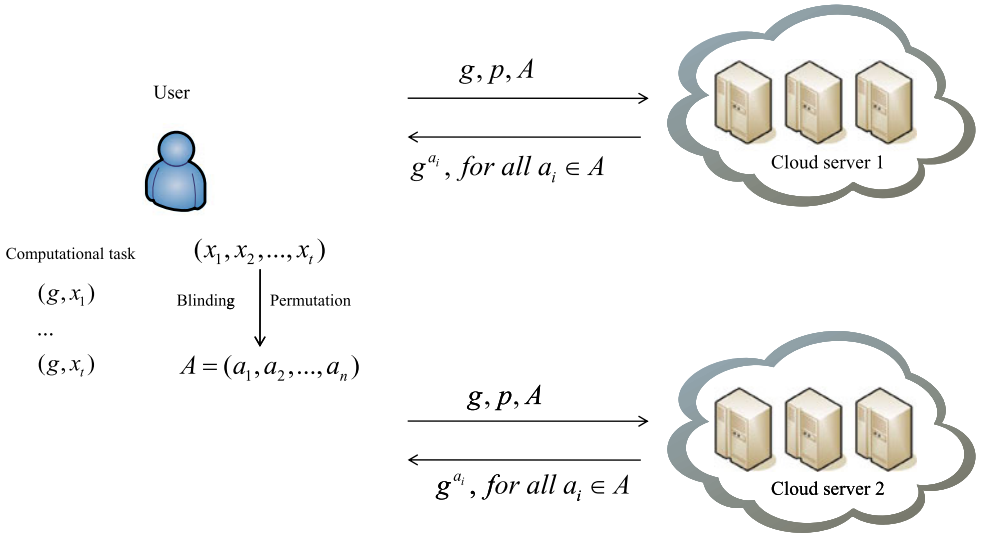

Fig. 1 Secure fixed base-variable exponent exponentiation outsourcing

each $i \in [1, t]$. The set can be constructed as randomly selecting $m-1$ numbers $b_1, b_2, \ldots, b_{m-1}$ from $\mathbf{Z}_p$, and then set $c_k = x_k - \sum_{j=1}^{m-1} b_j$, $1 \le k \le t$. Let $B = \{b_1, b_2, \ldots, b_{m-1}\}$. Then rest $m-1$ redundant numbers $r_1, r_2, \ldots, r_{m-1}$ are randomly selected from $\mathbf{Z}_p$. Finally, U re-randomizes the set of these numbers $A' = (b_1, b_2, \ldots, b_{m-1}, c_1, c_2, \ldots, c_t, r_1, r_2, \ldots, r_{m-1})$ using a pseudo-random permutation function $\pi : [1, n] \to [1, n]$ to a permutated set $A = (a_1, a_2, \ldots, a_n)$, where $n = 2(m-1) + t$. Then, U sends $\sigma_x = (A, g, p)$ to two independent cloud servers CS_1 and CS_2.

- $\sigma_y \leftarrow \mathsf{Compute}(f, \sigma_x)$. CS_1 and CS_2 return the corresponding exponentiation pair $\sigma_y = (v_{i1}, v_{i2}) = (g^{a_i}, g^{a_i})$, $i \in [1, n]$ to U one by one. $v_1 = v_2$ will not hold with overwhelming probability if either of them is dishonest.
- $y \cup \perp \leftarrow \mathsf{Verify}(\sigma_y, \tau)$. Upon receiving the value pair $\sigma_y = (v_{i1}, v_{i2})$, U first checks whether they are equal or not. If $v_{i1} \neq v_{i2}$, U concludes that the cloud servers are dishonest and aborts the protocol. Otherwise, U does the computation as follows, for $1 \le i \le n$, and $k \in [1, t]$:
  - $S_{base} = S_{base} \cdot v_{i1}$, if $a_{\pi^{-1}(i)} \in B$
  - $S_j = S_j \cdot v_{i1}$, if $a_{\pi^{-1}(i)} = c_j$, $j \in [1, t]$.

  At the end of the protocol, U completes the exponentiations as:

$$g^{x_i} = S_i \cdot S_{base}, \quad \text{for all } 1 \le i \le t.$$

3.2 Protocol 2. Fixed exponent-variable base exponentiation secure outsource

As described in Fig. 2, the fixed exponent-variable base scheme works almost the same as fixed base-variable exponent scheme.

- $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau, f)$ Given $(g_1, x), (g_2, x), \ldots, (g_t, x)$ as input that corresponding to an implicit request to compute $(g_1^x, g_2^x, \ldots, g_t^x)$, where $x \in \mathbf{Z}_p$, $g_1, g_2, \ldots, g_t \in G_p$, U first blinds the base vector $(g_1, g_2, \ldots, g_t)$ by generating a set of n pseudo-random numbers $A = \{h_1, h_2, \ldots, h_n\}$ in a manner that m of which can be multiplied to g_i, for each $i \in [1, t]$. The set can be constructed as choosing $m-1$ numbers $h_{b_1}, h_{b_2}, \ldots, h_{b_{m-1}}$ randomly from G_p, and then set $h_{c_k} = g_k / \prod_{j=1}^{m-1} h_{b_j}$, for $1 \le k \le t$. Let $H_B = \{h_{b_1}, h_{b_2}, \ldots, h_{b_{m-1}}\}$. The rest $m-1$ elements $h_{r_1}, h_{r_2}, \ldots, h_{r_{m-1}}$ can be randomly selected from G_p. Finally, U re-randomizes the set of these numbers $A' = (h_{b_1}, h_{b_2}, \ldots, h_{b_{m-1}}, h_{c_1}, h_{c_2}, \ldots, h_{c_t}, h_{r_1}, h_{r_2}, \ldots, h_{r_{m-1}})$ using a pseudo-random permutation $\pi : [1, n] \to [1, n]$ to the set $A = (h_1, h_2, \ldots, h_n)$, where $n = 2(m-1) + t$. Then, U sends $\sigma_x = (A, x, p)$ to the cloud servers CS_1 and CS_2.
- $\sigma_y \leftarrow \mathsf{Compute}(f, \sigma_x)$. CS_1 and CS_2 return the corresponding values $(v_{i1}, v_{i2}) = (h_i^x, h_i^x)$ to U one by one. $v_1 = v_2$ will not hold if either of them is dishonest.
- $y \cup \perp \leftarrow \mathsf{Verify}(\sigma_y, \tau)$. Upon receiving the value pair (v_{i1}, v_{i2}) from the cloud servers, U first checks whether they are equal or not. If $v_{i1} \neq v_{i2}$, U concludes that the cloud servers are dishonest and aborts the protocol. Otherwise, U does the computations as follows, for $1 \le i \le n$ and:
  - $M_{base} = M_{base} \cdot v_{i1}$ if $a_{\pi^{-1}(i)} \in B$
  - $M_j = M_j \cdot v_{i1}$, if $a_{\pi^{-1}(i)} = c_j$, $j \in [1, t]$.

  At the end of the protocol, U completes the exponentiations as:

$$g_i^x = M_i \cdot M_{base}, \quad \text{for all } 1 \le i \le t.$$

3.3 Protocol 3. Outsourcing computation of exponentiation with one server

In this subsection, we present a new outsourcing computation protocol in one sever model. The protocol is superior in both the verifiability and security compared with the

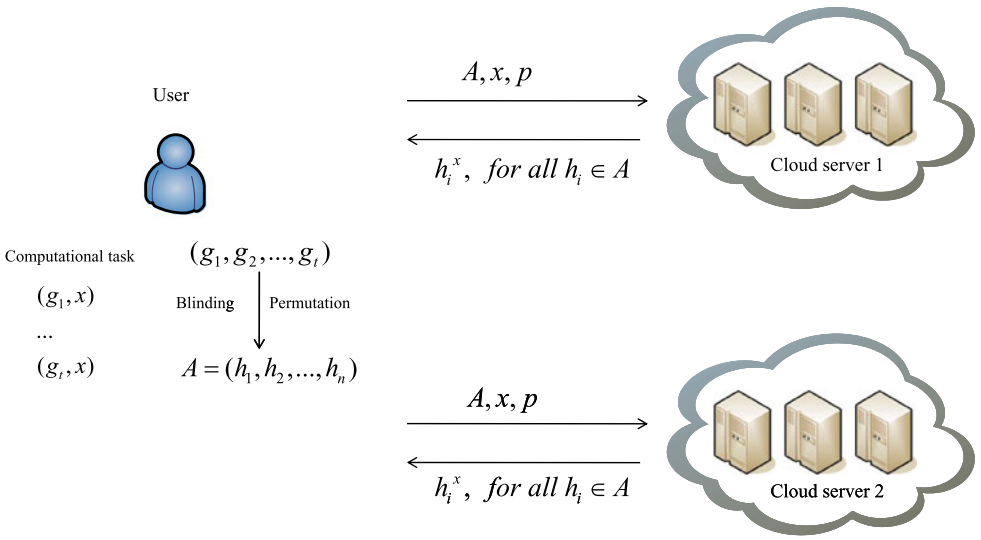

Fig. 2 Secure fixed exponent-variable base exponentiation outsourcing

above schemes. In the previous constructions, verifiability of the result is realized through the comparison of the values returned by the two cloud servers. Although verification process is quite efficient, we can not distinguish the honest server from the malicious server when there are inequalities. Below, we present the protocol for fixed base-variable exponent computations, and the protocol for fixed exponent-variable base computations can be constructed similarly.

- $\sigma_x \leftarrow \mathsf{ProbGen}(x, \tau, f)$. Given $(g, x_1), (g, x_2), \ldots, (g, x_t)$ as input that corresponding to an implicit request to compute $(g^{x_1}, g^{x_2}, \ldots, g^{x_t})$, where $x_i \in \mathbf{Z}_p$, $i \in [1, t]$, U first blinds the exponent vector $(x_1, x_2, \ldots, x_t)$ by generating a set of n pseudo-random numbers $A = \{a_1, a_2, \ldots, a_n\}$. The set can be constructed as randomly selecting $m-1$ numbers $b_1, b_2, \ldots, b_{m-1}$ and $m-1$ numbers $b'_1, b'_2, \ldots, b'_{m-1}$ from $\mathbf{Z}_p$, and then set $c_k = x_k - \sum_{j=1}^{m-1} b_j$, $1 \le k \le t$ and $c'_k = x_k - \sum_{j=1}^{m-1} b'_j$, $1 \le k \le t$. Let $B = \{b_1, b_2, \ldots, b_{m-1}\}$, $B' = \{b'_1, b'_2, \ldots, b'_{m-1}\}$. Finally, U re-randomizes the set of these numbers using a pseudo-random permutation function $\pi : [1, n] \rightarrow [1, n]$ to a permutated set $A = \{a_1, a_2, \ldots, a_n\}$, where $n = 2(m + t - 1)$. Then, U sends $\sigma_x = (A, g, p)$ to the server S.
- $\sigma_y \leftarrow \mathsf{Compute}(f, \sigma_x)$. S returns the corresponding exponentiation pair $\sigma_y = (v_1, \ldots, v_{2n}) = (g^{a_i})$, for all $i \in [1, 2n]$ to U.
- $y \cup \perp \leftarrow \mathsf{Verify}(\sigma_y, \tau)$. Upon receiving σ_y, U does the following verification to verify the validity of the values.
  - $S_{base} = S_{base} \cdot v_i$ if $a_{\pi^{-1}(i)} \in B$
  - $S_k = S_k \cdot v_i$ if $a_{\pi^{-1}(i)} = c_j$, $j \in [1, t]$.
  - $S'_{base} = S'_{base} \cdot v_i$ if $a_{\pi^{-1}(i)} \in B'$
  - $S'_j = S'_j \cdot v_i$ if $a_{\pi^{-1}(i)} = c'_j$, $j \in [1, t]$.
  - Then U verifies $S_i \cdot S_{base} \stackrel{?}{=} S'_i \cdot S'_{base}$, for all $i \in [1, t]$. If all the values are correct, U outputs the result as $g^{x_i} = S_i \cdot S_{base}$, for all $1 \le i \le t$.

4 Security and complexity analysis

4.1 Security analysis

Unlike previous cryptographic protocols of which the security is based on hard problem assumptions, such as factoring, DLP. We capture the security of our schemes through a reduction to Subset Sum Problem (SSP), which is an important problem in complexity theory and cryptography. The Subset Sum Problem is a NP-complete problem and can be defined as follows:

Definition 5 (Subset Sum Problem) Given a weight set of integers $(a_1, a_2, \ldots, a_n)$ and sum s, find a subset of which the elements sum to s, ie., determine the variables $x_1, x_2, \ldots, x_N \in \{0, 1\}$, such that $s = \sum_1^N x_i a_i$.

SSP is a special case of the knapsack problem [22]. We have to mention that not all of the SSP problems are difficult. The difficulty of SSP depends on the density of the set $(a_1, a_2, \ldots, a_n)$, which is defined as

$$d = \frac{n}{\log_2 \max_{1 \le i \le n} a_i}.$$

In our schemes, we set the density of the set to be

$$d = \frac{n}{\log_2 \max_{1 \le i \le n} a_i} \approx 1.$$

Modular Subset Sum Problem (MSSP) is a transformation of traditional SSP, in which $M, a_1, a_2, \ldots, a_N, b \in \{0, 1\}^l$ and find $x_1, x_2, \ldots, x_N \in \{0, 1\}$, such that $b = \sum_1^N x_i a_i \bmod M$. It has been proved that MSSP is equivalent to the traditional SSP in [9].

The most powerful method known to date to solve the Subset Sum Problem is lattice-based [14] method, which reduces the problem to the problem of finding a shortest vector

in a lattice. In practice, lattice basis reduction methods such as LLL algorithm or the Block-Korkine-Zolotarev (BKZ) reduction algorithm [14, 31] provide suitable approximations to the shortest lattice vector. They perform well for a small number of weights but breaks down for $N > 200$. For an increasing number of weights the quality of the approximation becomes insufficient for proving a solution to the Subset Sum Problem.

Theorem 1 *The protocols satisfy the security requirement of input privacy.*

Proof Semantically, for the fixed base-variable exponent exponential outsource scheme, the security requirement of input privacy means that the malicious adversary (server) knows nothing about the exponent. This security requirement is very important because the exponent is always the secret key of the user. In the following specification, we give a formal proof for Protocol 1. The security proof of input privacy for Protocols 2 and 3 can be constructed similarly, so we omit the details.

- Firstly, the malicious adversary obtains the set $A = \{a_1, a_2, \ldots, a_n\}$. Later, the user may publish an exponentiation g^{x_i} as public parameter of the signature schemes or encryption scheme. Assume that there exist an adversary $\mathcal{A}$ can successfully break the input privacy of our scheme, which means that $\mathcal{A}$ finds out the subset S that satisfies $g^{x_i} = g^{\sum a}, a \in S$. It is obvious that this is also a solution to the SSP with respect to A and sum x_i.
- For security analysis, we have to take the following attack into account. After the outsourcing computation, U obtains the exponentiations $g^{x_1}, g^{x_2}, \ldots, g^{x_t}$. Later, U may publish all of these exponentiations. Therefore, on inputs $(G_p, p, a_1, a_2, \ldots, a_n, g^{x_1}, g^{x_2}, \ldots, g^{x_t})$, the adversary $\mathcal{A}$ can compute $\frac{g^{x_j}}{g^{a_i}}$, for $1 \le i \le n$ and $1 \le j \le t$. In our scheme, there exist a base value $S_{base} = g^{\sum_{i=1}^{m-1} b_i}$ (or M_{base}) and $g^{x_k} = S_{base} g^{a_j}$ for $k \in [1, t]$, $a_j \in A$ and $a_j \notin B$. Therefore, among t sets $\{\frac{g^{x_i}}{g^{a_1}}, \frac{g^{x_i}}{g^{a_2}}, \ldots, \frac{g^{x_i}}{g^{a_n}}\}$, $1 \le i \le t$, there must exist the same value $v = S_{base} = g^{\sum_{i=1}^{m-1} b_i}$. Suppose that $\mathcal{A}$ has detected $v = \frac{g^{x_k}}{g^{a_l}}$, then $\mathcal{A}$ can conclude that $a_l \notin B$. Similarly, $\mathcal{A}$ can detect the rest $t - 1$ numbers $a_j \notin B$. Therefore, the problem that the adversary $\mathcal{A}$ has to settle now is transformed to another SSP, that is, given a set of $n - t$ elements $A' = \{a'_1, a'_2, \ldots, a'_{n-t}\}$, determine which subset B of A', satisfies $S_{base} = g^{\sum a_i}, a_i \in B$, which is also a modular SSP problem. For security, we set $n - t > 200$ in our scheme, and all the known attacks break down in this case.

In the end, we have proved that for any PPT adversary $\mathcal{A}$, the probability

$$\Pr\{\mathcal{A}(g, G_p, g^{x_1}, g^{x_2}, \ldots, g^{x_t}, a_1, a_2, \ldots, a_n) = x_i\}$$

is negligible in k. Similarly, we can prove the security requirement for Protocols 2 and 3. And we can conclude that for any PPT adversary $\mathcal{A}$, the probability

$$\Pr\{\mathcal{A}(G_p, x, g_1^x, g_2^x, \ldots, g_t^x, g'_1, g'_2, \ldots, g'_n) = g_i\}$$

is negligible in k. □

So far, the best algorithm for discrete logarithm problem (DLP) $y = g^x$ is $O(\sqrt{q})$, where q is the order of g. To achieve an approximately equal computation complexity of DLP we have to carefully set the parameters n, m, and t. The formula that n, m and t have to satisfy is based on the above attack game. In our scheme, we set $2(m - 1) > 200$. In this case, the successful probability of the adversary is less than $\frac{1}{2^{80}}$.

Theorem 2 *The protocols satisfy the security requirement of verifiability.*

Proof The security requirement of soundness means that whenever the untrusted cloud servers return wrong values, the user can detect its dishonesty with high probability.

In Protocols 1 and 2, two non-collusion untrusted cloud servers are incorporated into the system, which can be realized in the real world easily. The technique used in our scheme greatly reduces the redundancies that have to be inserted into the input $(x_1, x_2, \ldots, x_n)$ and thus saves the bandwidth for interactive communication. The error detection is simplified as a equality verification of the return values rather than complicated computations. As long as the cloud servers return the same values, the user concludes that they are honest and vice versa.

In both of our schemes, we assume that the cloud servers do not collude with each other, which is a practical assumption in the real world. In the security definition of verifiability, we defined a query and response phase. However, for each problem generation of our scheme, a new random set $s_1, \ldots, s_{m-1}$ is generated. Thus, nothing useful information is leaked to $\mathcal{A}$ for the challenge phase.

For each cloud server, the probability that each returned value is computed correctly or wrongly is $1/2$. In both of our schemes, the probability that the user can detect the malicious cloud server's dishonesty is $P = \frac{3}{4}$. The probability is obtained from the analysis of three independent cases. Specifically, the inconsistency occurs when both of the cloud servers are dishonest or either of them is dishonest.

For Protocol 3, a malicious adversary can pass the verification with wrong values only if he is able to find the solutions for the two distinct SSPs. Whereas, given a randomly permuted sequence $\{a_1, \ldots, a_n\}$, the probability that a malicious adversary $\mathcal{A}$ can find the solutions of the SSPs for

any exponent x_i is less than $\frac{1}{\binom{n}{m-1}}\frac{1}{\binom{n-m+1}{m-1}}$. In our scheme, the parameters are set to satisfy $n > 200$, $m \approx n/2$. Thus, the success probability of the adversary is negligible. □

Theorem 3 *The protocols satisfy the security requirement of correctness.*

Proof If all the values returned by the cloud servers are correct, then at the **Compute** stage, U gets the correct value $g^{x_1}, g^{x_2}, \ldots, g^{x_t}$ in Scheme 1 and $g_1^x, g_2^x, \ldots, g_3^x$ in Scheme 2. The procedure can be verified as:

$$S_{base} = \prod_{i=1}^{m-1} g^{b_i} = g^{\sum_{i=1}^{m-1} b_i}, \qquad M_{base} = \prod_{i=1}^{m-1} h_{b_i}^x,$$

$$S_k = S_k \cdot S_{base} = g^{c_k} g^{\sum_{i=1}^{m-1} b_i} = g^{c_k + \sum_{i=1}^{m-1} b_i}$$

$$= g^{c_k - \sum_{i=1}^{m-1} b_i + \sum_{i=1}^{m-1} b_i} = g^{x_k}, \quad k \in [1, t],$$

$$M_k = M_k \cdot M_{base} = h_{c_k}^x \prod_{i=1}^{m-1} h_{b_i}^x$$

$$= \left(\frac{g_k}{\prod_{i=1}^{m-1} h_{b_i}}\right)^x \prod_{i=1}^{m-1} h_{b_i}^x = g_k^x, \quad k \in [1, t].$$

□

4.2 Complexity analysis

We have to emphasize that our schemes are more efficient compared with any previous server-aided fast exponentiation scheme in both the storage complexity and computation complexity. First of all, our schemes do not require any complicated precomputations, which is a great progress compared with the previous fast exponentiation schemes [12, 21]. Below, we will formalize the efficiency analysis in computational, communication and communication complexity.

Computation complexity We split our schemes into offline and phase. In the offline phase, the user generates the set A. For Protocol 1, the generation of the set A requires t modular additions, which are so efficient so that we can omit them. For Protocol 2, the generation of the set A requires 1 inversion and t modular multiplications. In the online phase, the overall computation costs result from the **Verify** stage of the protocol. Moreover, a batch of t $(1 \le t \le n - m)$ exponentiations can be computed in just one implementation of our scheme. In our schemes, there are base values, S_{base} and M_{base}, that used for batch exponentiation computations. $m - 2$ modular multiplications are needed to compute the base value. Then the base value can be used repeatedly to compute as many as t exponentiations. In Protocols 1 and 2, for one more exponentiation, just one more modular multiplication is need. Therefore, for t exponentiations, all the computations that U has to do are $m + t - 2$ modular multiplications, which is very efficient. For Protocol 3, due to the verification algorithm, the computation cost of the user is double that of Protocols 1 and 2, but still very efficient.

We have to mention that for small batches, for example just one modular exponentiation is outsourced, our schemes are still efficient for long exponent, such as the exponent d of RSA signature is about 1024 bits. As we know, for τ-bit long exponent, 1.5τ multiplications are needed to complete the computation if the user does it by himself. Whereas, only approximately 100 multiplications are needed if the user utilizes our outsourcing scheme.

Table 1 Efficiency comparison

t Exps	[21]	[12]	Protocol 1	Protocol 3
MM	$9t$	$7t$	$m + t - 2$	$2(m + t - 2)$
MInv	$5t$	$3t$	0	0
Verifiability	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{3}{4}$	≈ 1

Communication complexity In our schemes, the user has to send the set A to the servers. For Protocol 1, A consists of n elements from Z_p. As there are two servers, thus the overall communication complexity is $2n$ elements from Z_p. For Protocol 2, the communication complexity is $2n$ elements from G_p. And the communication complexity of Protocol 3 is n elements from Z_p. For the security of our schemes, n is usually set to be greater than 200.

Storage complexity In Protocols 1 and 2, the user just needs to store the temporary value $S_{base}(M_{base})$, and $S_k(M_k)$, $k = 1, \ldots, t$. And for Protocol 3, the storage complexity is double that of the previous schemes. And for all the schemes, the user has to store the permutation π to correctly compute the final modular exponentiations.

Table 1 presents the efficiency comparison between our scheme and [12, 21]. We denote by MM a modular multiplication, by MInv a modular inverse. Note that Protocol 2 is eliminated from the table, because Protocol 2 is designed form variable base exponentiations. Therefore, our schemes are practical and very efficient, and can be widely used in the resource limited application scenarios.

5 Conclusion

In this paper, we presented secure outsourcing schemes enabling users to securely outsource the computations of exponentiations to servers. The first two schemes require two non-collusion servers, while the third scheme only needs one server. In our schemes, a batch of exponentiations (e.g. t exponentiations) can be efficiently computed by the user with only $O(n + t)$ multiplications, where n is the number of bits of the exponent. Furthermore, there is not any complicated pre-computations on the user side. We also showed

that the proposed schemes are provably secure under the Subset Sum Problem.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Nos. 61070168, 61100224 and U1135001), the National Basic Research Program of China (973 Program, No. 2012CB316100), and the Specialized Research Fund for the Doctoral Program of Higher Education, and the Foundation for Distinguished Young Talents in Higher Education of Guangdong Province (No. LYM10106).

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: efficient verification via secure computation. In: ICALP 2010, pp. 152–163 (2010)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the cloud: a Berkeley view of cloud computing. Berkeley University (2009)
3. Assuncao, M.D., Costanzo, A., Buyya, R.: A cost-benefit analysis of using cloud computing to extend the capacity of clusters. Clust. Comput. **13**(3), 335–347 (2010)
4. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: AISACCS 2010, pp. 48–59 (2010)
5. Babai, L.: Trading group theory for randomness. In: Proceeding of ACM Symposium on Theory of Computing (STOC), pp. 421–429 (1985)
6. Barbosa, M., Farshim, P.: Delegatable homomorphic encryption with applicatoins to secure outsourcing of computation. In: CT-RSA 2012. LNCS, vol. 7178, pp. 296–312. Springer, Heidelberg (2012)
7. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
8. Benjamin, D., Atallah, M.J.: Private and cheating-free outsourcing of algebraic computations. In: PST 2008, pp. 240–245 (2008)
9. Boyko, V., Peinado, M.: Speeding up discrete log and factoring based schemes via precomputation. In: EUROCRRYPTO 1998. LNCS, vol. 1403, pp. 221–232. Springer, Heidelberg (1998)
10. Chapman, C., Emmerich, W., Marquez, F.G., Clayman, S., Galis, A.: Software architecture definition for on-demand cloud provisioning. Clust. Comput. **15**(2), 79–100 (2012)
11. Chaum, D., Pedersen, T.P.: Wallet database with observers. In: CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
12. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. In: ESORICS 2012. LNCS, vol. 7459, pp. 541–556. Springer, Heidelberg (2012)
13. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved delegation of computation using fully homomorphic encryption. In: CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
14. Coster, M.J., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. Comput. Complex. **2**(2), 111–128 (1992)
15. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceeding of ACM Symposium on Theory of Computing (STOC), pp. 169–178 (2009)
17. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
18. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceeding of ACM Symposium on Theory of Computing (STOC), pp. 113–122 (2008)
19. Golle, P., Mironov, I.: Uncheatable distributed computation. In: CT-RSA 2001. LNCS, vol. 2020, pp. 425–550. Springer, Heidelberg (2001)
20. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: Proceeding of the USENIX Security Symposium (2011)
21. Hohenbergera, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005)
22. Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. J. ACM **21**(2), 277–292 (1974)
23. Jakobsson, M., Wetzel, S.: Secure server-aided signature generation. In: PKC 2001. LNCS, vol. 1992, pp. 383–401. Springer, Heidelberg (2001)
24. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceeding of ACM Symposium on Theory of Computing (STOC), pp. 723–732 (1992)
25. Kilian, J.: Improved efficient arguments. In: CRYPTO 1995. LNCS, vol. 963, pp. 311–324. Springer, Heidelberg (1995)
26. Matsumoto, T., Kato, K., Iami, H.: Speeding up secret computations with insecure auxiliary devices. In: CRYPTO 1990. LNCS, vol. 403, pp. 497–506. Springer, Heidelberg (1990)
27. Micali, S.: CS proofs (extended abstract). In: Proceeding of the 35th IEEE Symposium on Foundations of Computer Science, pp. 436–453 (1994)
28. Papamanthou, C., Shi, E., Tamassia, R.: Publicly verifiable delegation of computation. http://eprint.iacr.org/2011/587
29. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: TCC 2012. LNCS, vol. 7194, pp. 422–439 (2012)
30. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012)
31. Schnorr, C.P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**(1–3), 181–199 (1994)

Xu Ma is a Ph.D. candidate of School of Information Science and Technology, Sun Yat-sen university, China. His research mainly focuses on cryptography and its applications, especially on secure outsourcing computation.

Jin Li received his B.S. (2002) and M.S. (2004) from Southwest University and Sun Yat-sen University, both in Mathematics. He got his Ph.D. degree in information security from Sun Yat-sen University at 2007. Currently, he works at Guangzhou University. His research interests include Applied Cryptography and Security in Cloud Computing (secure outsourcing computation and cloud storage).

Fangguo Zhang received his Ph.D. from the School of Communication Engineering, Xidian University in 2001. He is currently a Professor at the School of Information Science and Technology of Sun Yat-sen University, China. He is the co-director of Guangdong Key Laboratory of Information Security Technology. His research mainly focuses on cryptography and its applications. Specific interests are elliptic curve cryptography, secure multi-party computation, anonymity and privacy.