Verifiable and Secure Outsourcing Schemes of Modular Exponentiations Using One Untrusted Cloud Server and Their Application

Can Xiang [1,2] Chun-ming Tang[1,2]
[1]School of Mathematics and Information Science, Guangzhou University, guangzhou 510006, P.R.China
[2]Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes, Guangzhou University, Guangzhou, P.R. China
[Email: xiangcan1987@sina.com]

Abstract: Modular exponentiation is one of basic operations among most of current cryptosystems. Under some algebraic assumptions or cryptography assumptions, it can construct outsourcing schemes for modular exponentiations by using two untrusted cloud servers, which cannot resist the collusive attack of two untrusted cloud servers. However, it doesn't exist an efficient outsourcing of modular exponentiations using one untrusted cloud server. In this paper, we present three secure outsourcing schemes using one untrusted cloud server, which can enable user to securely outsource exponentiations to single cloud server. The first one is a secure outsourcing scheme for fixed base-variable exponent modular exponentiations, the second is for variable base-variable exponent modular exponentiations, and the third is a secure outsourcing scheme for simultaneous modular exponentiations. Compared with other proposed schemes, our proposed schemes are superior in both efficiency and checkability. Moreover, our schemes are secure without any cryptographic assumptions. Finally, we give two applications for our outsourcing schemes, one is to construct an outsourcing scheme for Cramer-Shoup encryptions, and the other is to design an outsourcing scheme for Schnorr signatures.
Keywords: Cloud computing, Outsourcing computation, Single untrusted cloud server, Modular exponentiation.

1. Introduction

Cloud computing is a new calculation method, in which user can get the required service and the required resources via internet. It means that the computing power is considered as a commodity which can circulate via internet. Cloud computing services generally offer the online commerce application which can be accessed by browser, but the software and the data can be stored in the data center. In the cloud computing environment, user generally has a device with inputting and outputing, and has practically no capacity to storage, compute and deal with data. However, user can get the desired services via internet. For example, user needs to perform a computing task and has no capacity to complete the task by himself, then he has to ask for help from cloud server.

Outsourced computation is one of the advantages of cloud computing model, and it can be able to make the computing power of cloud users no longer limited to their respective resource-constrained equipment. By outsourcing workload to cloud servers, cloud users can use the infinite resources of the cloud server to finish computing with high cost. Because the operation of internal cloud is not transparent to users[1], there are a variety of motives which make the behavior of the cloud server dishonest. For example, for computing that needs a large amount of computing resources, if the user cannot judge the correctness of cloud server's output, the cloud may be "idle" in order to save resources. Hence, this paradigm also brings forth new challenges for security. The problem that researchers face is how to construct secure outsourcing scheme, which has become a hot topic in the cloud computing. As far as applications are concerned, an efficient outsourced computation scheme should satisfy three

This work is supported by National Science Foundation of China(11271003), and Guangdong Provincial Natural Science Foundation(No.52012010009950),Higher Level Talents Project of Guangdong.

basic conditions as follows.

(1) The privacy of the user's input and output.

(2) The correctness of the cloud computing output.

(3) The workload that user end needs under this scheme (including verification of correctness) is less than workload that the user computed by himself, otherwise, it is not necessary for user to seek assistance of the cloud.

It is generally known that modular exponentiation u^a is one of basic operations in cryptosystems, such as RSA, DSS, and ElGmal et al. However, user requires $O(n)$ modular multiplications by himself to compute modular exponentiation u^a, where n is the bit length of exponent a. The computation cost is extremely large, which hampers the application of exponentiations for limited resource devices because that the device has less computation power. In order to reduce the computation cost of user, user needs to ask for help from cloud server CS who has enough computing power. In this paper, we research how to outsource modular exponentiations.

Related Work. The outsourcing scheme for exponentiation was firstly proposed in [2]. Later, many outsourcing schemes were proposed in [3~6]**.** In [7], Jakobsson described how to securely outsource signature by using outsourcing modular exponentiations. In this scheme, the user had to blind the exponent before outsourcing, and the scheme can support batch exponentiations, but it was only efficient for batch exponentiations with small size. Without the outsourcing technique, some methods were proposed in the terms of how to speed up the modular exponentiations on local side in [8~13]. However, the overall computation complexity of user required $O(n)$, which was still expensive for limited resource devices with less computation power. In [14], Clarke proposed protocols for speeding up fixed-base exponentiation and variable base exponentiation using an untrusted computation resource, which provided a speedup of $3(\log_2^n - 1)/2$ over the square-and-multiply algorithm (where n is the bit length of exponent). Later, Ma et al. [15] proposed two generic secure outsourcing schemes for modular exponentiation by using two non-colluding untrusted cloud servers, which enabled a user to securely outsource modular exponentiations to two non-colluding untrusted cloud servers, but a batch (eg. t exponentiations) of modular exponentiations can be efficiently computed by the user with $O(\mathrm{n}+\mathrm{t})$ multiplications (where n is the bit length of the exponent), and these schemes were secure under the hidden subset sum problem. Secure outsourcing scheme of modular exponentiations is a popular topic [5,10,16~22], but past methods focus on fixed-base (or fixed-exponent) modular exponentiations, which were suit for a weaker notion of security. Hohenberger and Lysyanskaya [6] proposed a secure outsourcing scheme for variable base-variable exponent exponentiation, which outsources modular exponentiations to two non-colluding untrusted servers. With their techniques, modular exponentiations can be efficiently computed by the user with $O(\log^2 n)$ multiplications. The algorithm was improved in [23], in which Chen et al. proposed a new secure outsourcing algorithm for exponentiation modular a prime by using two non-colluding untrusted servers. Compared with the scheme [6], their scheme was superior in both efficiency and checkability, but exponentiations can be efficiently computed by the user with $O(\log^2 n)$ multiplications and the scheme has checkability with only 2/3. Then, Chen et al. utilized this algorithm as a subroutine to complete outsource-secure Cramer-Shoup encryptions and Schnorr signatures.

In addition, simultaneous modular exponentiations $u_1^a u_2^b$ plays a significant role in many cryptographic primitives such as chameleon hashing [24~29] and trapdoor commitment [30~34]. In general, a simultaneous modular exponentiation operation can be implemented by invoking 2 modular exponentiations. This requires roughly *3n* MM, where *n* is the bit length of *a* and *b*. However, the computation cost is only *1.75n* MM if the Algorithm *14.88* of [35] is adopted. In [23], Chen et al. firstly presented a secure outsourcing algorithm for simultaneous

exponentiation modular a prime by using two non-colluding untrusted servers, which required 5 invocations of *Rand1*. Although the computation complexity of user is $O(\log^2 n)$, the computational cost of user was still extremely large and the scheme had checkability with only 1/2. In order to improve efficiency, we propose a new secure outsourcing scheme (**Sexp**) for simultaneous modular exponentiations $u_1^a u_2^b$ by using one untrusted cloud server.

Contribution. In order to reduce user's cost and avoid the collusive attack of multiple untrusted cloud servers, we firstly present three efficient and provably secure modular exponentiation outsourcing schemes using one untrusted cloud server. The first is fixed base-variable exponent exponentiation outsourcing scheme (**FBVE**), the second is variable base-variable exponent exponentiation outsourcing scheme (**Exp**), and the third is a secure outsourcing scheme for simultaneous exponentiations (**Sexp**).

Under this assumption that there only exists one untrusted cloud server, modular exponentiation can be computed by the user within only $O(1)$ multiplications in our scheme **FBVE**, and only $O(1)$ multiplications and two operations for modular inverse in other two schemes. In addition, our schemes are secure without any cryptographic assumption. Compared with the proposed schemes [6] and [23], our schemes realize modular exponentiations using only one cloud server, and our schemes are superior in both efficiency and checkability.

Finally, we give two applications for our proposed outsourcing schemes, one is to construct an outsourcing scheme for Cramer-Shoup encryptions, and the other is to design an outsourcing scheme for Schnorr signatures.

Organization . The rest of the paper is organized as follows. The security model of our schemes is given in Section 2. Section 3 details our three modular exponentiations outsourcing schemes and proves security and high efficiency of our schemes. Two applications for our outsourcing schemes are given in section 4. And section 5 concludes the paper.

2. Model

In this paper, two parties are involved in our schemes, that is, the user T and one untrusted cloud server CS. In the following, our model can be described.

(1)Given modular exponentiations which will be computed, the user T does some transformations on these values and sends them to CS.

(2) After receiving these transformed values, CS computes and returns the results to T.

(3) After receiving all the values from CS, T can complete modular exponentiations with another transformation. At the same time, the user T can verify the correctness of the results of calculations by CS.

The security threats our model face mainly come from the dishonesty of the cloud server CS. The type of the dishonesty includes active and passive type. The passive action means that the adversary controls the cloud server and intends to obtain some secret information of cloud users through faithfully executing protocols. The active action means that the adversary, who controls the cloud server, has strong attacking power, can tamper data, and can not follow the requirement of the protocols in performing the protocols, even unilaterally terminate the protocols. In this paper, we suppose that the action of cloud server CS is active.

An efficient and secure outsourcing scheme should satisfy five properties.

Correctness. Outputs generated by any honest cloud server are acceptable to users.

Soundness. Any error output generated by the dishonest cloud server can be accepted by the user with negligible probability.

Privacy. During the period of implementation protocol of cloud server, any sensitive information about the user's private data can not be deduced by the server.

Verifiability. Users can verify the correctness of the honest cloud server output with great

probability, and can also verify the incorrectness of dishonest cloud server output with great probability.

Efficiency. The computation cost that user needs under this scheme (including the correctness verification) is less than computation cost that the user computes by himself.

3. Secure outsourcing schemes of modular exponentiation using one untrusted cloud server

In this section, we give three efficient, verifiable and secure outsourcing schemes for modular exponentiations by using one untrusted cloud server. That is, the secure outsourcing scheme for the fixed base-variable exponent modular exponentiations (**FBVE**), for the variable base -variable exponent exponentiation (**Exp**), and for simultaneous modular exponentiations (**Sexp**). Compared with other proposed schemes, our schemes realize modular exponentiations by using only one cloud server, and our schemes are superior in both efficiency and checkability.

3.1 Secure Outsourcing Scheme of Fixed Base-Variable Exponent Exponentiations

Let p,q be two large primes and $q \mid (p-1)$. Let $a \in z_q^*$ and $u \in z_p^*$, such that $u^q = 1 \bmod p$ .Suppose that the user T wants to compute the exponentiation u^a for some base $u \in z_p^*$ and exponent a. In general, u^a can be computed by the user with $O(\mathrm{n})$ modular multiplications, where n is the bit length of exponent a. The goal of the user T is to compute u^a with less computation cost. An untrusted cloud server CS who has enough computing power, T needs to ask for help from cloud server. The fixed base-variable exponent exponentiation scheme (**FBVE**) is described in the following section (where a is arbitrary and blinded, u is fixed and public).

Step1. T randomly selects seven integers $(a_1, a_3, a_5, \varepsilon_1, \varepsilon_3, \varepsilon_5, s)$, and computes $a_2 = a + 1 - (a_1 \varepsilon_1^{\,2})^2, a_4 = a - s - (a_3 \varepsilon_3^{\,2})^2$ and $a_6 = s - (a_5 \varepsilon_5^{\,2})^2$. For each $i \in \{i \mid 1 \le i \le 6, i \in Z\}$, a_i is secret.

Step2. T randomly chooses six integers $(k_1, k_3, k_5, r_1, r_3, r_5)$. For each $i \in \{1,3,5\}$, T computes $a_i' = k_i \varepsilon_i \mathrm{r}_i + a_i \varepsilon_i^{\,2}, a_i'' = k_i \mathrm{r}_i^{\,2} + 2 a_i \mathrm{r}_i \varepsilon_i$, and $\mu_i = -k_i \varepsilon_i^{\,2}$. Then, the user T puts nine numbers $\{a_i', a_i'', a_2, a_4, a_6\} (i = 1,3,5)$ into a matrix $B_{m_1 \times m_2}$ (suppose $m_1 = m_2 = m$). Meanwhile, T puts three numbers $\{\mu_1, \mu_3, \mu_5\}$ into a matrix $C_{m \times m}$. In the matrices B and C , the positions of these numbers are determined by the user T . These positions are the privacy of the user T . For example, suppose $b_{11} = a_1', b_{42} = a_2, b_{63} = a_3', b_{24} = a_4, b_{36} = a_5', b_{71} = a_6$, $b_{56} = a_1''$, $b_{87} = a_3'', b_{68} = a_5'', c_{56} = \mu_1$, $c_{87} = \mu_3, c_{68} = \mu_5$, that

$$B = \begin{pmatrix} a_1' & \# & \# & \# & \# & \# & \# & \# & \cdots & \# \\ \# & \# & \# & a_4 & \# & \# & \# & \# & \cdots & \# \\ \# & \# & \# & \# & \# & a_5' & \# & \# & \cdots & \# \\ \# & a_2 & \# & \# & \# & \# & \# & \# & \cdots & \# \\ \# & \# & \# & \# & \# & a_1'' & \# & \# & \cdots & \# \\ \# & \# & a_3' & \# & \# & \# & \# & a_5^* & \cdots & \# \\ a_6' & \# & \# & \# & \# & \# & \# & \# & \cdots & \# \\ \# & \# & \# & \# & \# & \# & a_3'' & \# & \cdots & \# \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \# & \# & \# & \# & \# & \# & \# & \# & \cdots & \# \end{pmatrix}_{m \times m}, \mathrm{C} = \begin{pmatrix} \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \mu_1 & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \# & \# & \mu_5 \cdots & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \\ \# & \# & \# & \# & \# & \# & \mu_3 & \# \cdots & \# \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \# \cdots & \vdots \\ \# & \# & \# & \# & \# & \# & \# & \# \cdots & \# \end{pmatrix}_{m \times m}.$$

Then, T sends B , C and u to the cloud server CS .

Step3. CS computes $D=(u^{\mathrm{b}_{ij}^2})_{m\times m}$, $F=(u^{\mathrm{b}_{ij}})_{m\times m}$, and $E=(u^{b_{ij}c_{ij}})_{m\times m}$. Then, CS returns D, E and F to the user T.

Step4. T computes $m_1=(\mathrm{d}_{11}\cdot e_{56})\cdot \mathrm{f}_{42}$, $m_2=(\mathrm{d}_{63}\cdot e_{87})\cdot \mathrm{f}_{24}$, and $m_3=(\mathrm{d}_{36}\cdot e_{68})\cdot \mathrm{f}_{71}$. If $m_1\neq u\cdot m_2m_3$, then T concludes that the cloud server CS is dishonest and aborts the protocol. Otherwise, T completes the exponentiation as $u^a=m_2m_3$.

Theorem 1. In a single-cloud server model, our scheme **FBVE** is verifiable and secure.

Proof.

(1)Correctness. The correctness of the scheme FBVE is obvious. If the cloud server CS honestly implements the scheme, it will certainly make users accept its output.

(2)Soundness. The cloud server CS may dishonestly act as follows. Firstly, CS changes the values of certain elements in matrices D, E ,and F. In this case, the correctness of u^a is not affected as long as CS does not change the values of $(d_{11},d_{63},d_{36},e_{56},e_{87},e_{68},f_{42},f_{24},f_{71})$. Otherwise, if CS wants to pass the verification equation $\mathrm{d}_{11}\,e_{56}\,\mathrm{f}_{42}=u\cdot(\mathrm{d}_{36}\,e_{68}\,\mathrm{f}_{71})(\mathrm{d}_{63}\,e_{87}\,\mathrm{f}_{24})$ in Step4,then he needs to guess the positions of (d_{11},d_{63},d_{36}) in the matrix D ,the positions of (e_{56},e_{87},e_{68}) in the matrix E, and the positions of (f_{42},f_{24},f_{71}) in the matrix F. The probability that CS finds all the right positions of nine numbers $\{\mathrm{d}_{11},e_{56},\mathrm{f}_{42},\mathrm{d}_{36},e_{68},\mathrm{f}_{71},\mathrm{d}_{63},e_{87},\mathrm{f}_{24}\}$ is $1/(m^2(m^2-1)(m^2-2))^3$. Secondly, CS changes the values of certain elements in matrices B and C. In this case, if CS wants to pass the verification equation $\mathrm{d}_{11}\,e_{56}\,\mathrm{f}_{42}=u\cdot(\mathrm{d}_{36}\,e_{68}\,\mathrm{f}_{71})(\mathrm{d}_{63}\,e_{87}\,\mathrm{f}_{24})$ with affecting the correctness of u^a, then CS needs to guess the right positions of $\{a_1',a_1'',a_3',a_3'',a_5',a_5'',a_2,a_4,a_6\}$ in the matrix B and the positions of $\{\mu_1,\mu_3,\mu_5\}$ in the matrix C. The probability that CS finds all the right positions of these 12 numbers is $\dfrac{(\mathrm{m}^2-9)!}{m^2!\mathrm{m}^2(\mathrm{m}^2-1)(\mathrm{m}^2-2)}$. Thirdly, CS calculates $D=((u')^{\mathrm{b}_{ij}^2})_{m\times m}$, $F=((u')^{\mathrm{b}_{ij}})_{m\times m}$ and $E=((\mathrm{u}')^{b_{ij}c_{ij}})_{m\times m}$ by randomly selecting a number u' (where $u'\neq u$), and returns these three matrices to the user T. In this case, if CS wants to pass the verification equation $\mathrm{d}_{11}\,e_{56}\,\mathrm{f}_{42}=u\cdot(\mathrm{d}_{36}\,e_{68}\,\mathrm{f}_{71})(\mathrm{d}_{63}\,e_{87}\,\mathrm{f}_{24})$, CS needs to guess the positions of (d_{11},d_{63},d_{36}) in the matrix D, (e_{56},e_{87},e_{68}) in the matrix E, and (f_{42},f_{24},f_{71}) in the matrix F. The probability that the positions of nine number $\{\mathrm{d}_{11},e_{56},\mathrm{f}_{42},\mathrm{d}_{36},e_{68},\mathrm{f}_{71},\mathrm{d}_{63},e_{87},\mathrm{f}_{24}\}$ are guessed successfully, is $1/(m^2(m^2-1)(m^2-2))^3$.

(3) Privacy. Our outsourcing scheme FBVE does not leak any valuable information for both a and u^a. If an attacker want to recover a, then he needs to get the values of a_1, a_2 and ε_1 (or a_3, a_4, ε_3 and s)at the same time. The probability of retrieving a_2 by the matrix B is $1/m^2$, and the probability of retrieving a_1 and ε_1 by the equation $a_1'=k_1\varepsilon_1\,\mathrm{r}_1+a_1\varepsilon_1^2$ and $a_1''=k_1\,\mathrm{r}_1^2+2a_1\,\mathrm{r}_1\,\varepsilon_1$ is negligible because k_1, ε_1 ,and r_1 are randomly selected and are secret. In addition, if the attacker wants to obtain u^a, then he needs to get the values of d_{11},e_{56} ,and f_{42} (or d_{63},e_{87}, f_{24}, d_{36}, e_{68}, f_{71}) at the same time. The probability of retrieving the values of d_{11},e_{56} ,and f_{42} at the same time is $1/m^6$.

Let X be the event of "attackers successfully recover some private data". Thus, the probability of X is at most $1/m^6$.

(4) Verifiability. It is assumed that cloud servers are bribed by a *PPT* attacker, and then the *PPT* attacker will access all information that the cloud servers receives and computes. Let X be the event of " CS can pass the verification equation $\mathrm{d}_{11}\,e_{56}\,\mathrm{f}_{42}=u\cdot(\mathrm{d}_{36}\,e_{68}\,\mathrm{f}_{71})(\mathrm{d}_{63}\,e_{87}\,\mathrm{f}_{24})$ and affect the correctness of u^a ". When selecting the first kind of attack behavior described in

soundness, the probability of X is $\Pr[X]=1/(m^2(m^2-1)(m^2-2))^3$ for the *PPT* attacker. When selecting the second kind of attack behavior described in soundness, the probability of X is $\Pr[X]=\frac{(m^2-9)!}{m^2!m^2(m^2-1)(m^2-2)}$ for the *PPT* attacker. When selecting the third kind of attack behavior described in soundness, the probability of X is $\Pr[X]=1/(m^2(m^2-1)(m^2-2))^3$ for the *PPT* attacker.

Let Y be the event of "attackers forge result and successfully cheat users, and the correctness of u^a is affected ". Thus, the probability of Y is $\Pr[Y]\le\frac{(m^2-6)!}{m^2!m^2(m^2-1)}+2(\frac{1}{m^2(m^2-1)(m^2-2)})^3$ for the *PPT* attacker. In other words, the user T will detect the error output u^a with probability at least $(1-\frac{(m^2-6)!}{m^2!m^2(m^2-1)}-2(\frac{1}{m^2(m^2-1)})^3$. We noticed that the checkability is approximately equal to one when the matrix size m is a big number.

(5) Efficiency. In general, u^a can be computed by the user with $O(n)$ multiplications, where n is the bit lengths of exponent a. Although these methods[8-13] can speed up the modular exponentiation, the overall computation complexity of offline stage (pre-computation) and online stage still required $O(1.5n)$ multiplications, which was still a huge burden on resource-limited devices. In [14], Clarke proposed protocols for speeding up fixed base-variable exponent exponentiation and variable base-fixed exponent exponentiation using an untrusted computation resource, the protocols provided a speedup of $3(\log_2^n-1)/2$ over the square-and-multiply algorithm. In [6,23], exponentiations can be efficiently computed by the user with $O(\log^2 n)$ multiplications. In [15], Ma et al. proposed two generic secure outsourcing schemes for modular exponentiations by using two untrusted cloud servers, these schemes enabled users to securely outsource the computations of exponentiations to two untrusted cloud servers, but a batch of exponentiations (eg. t exponentiations) can be efficiently computed by the user with $O(n+t)$ multiplications and their schemes were proved to be secure under the Hidden Subset Sum Problem. Our scheme **FBVE** only needs to rent one untrusted cloud server, and the computation cost of the user requires only 50 multiplications, which means that computation complexity of the user only has $O(1)$. In addition, our scheme don't require any cryptographic assumption.

We now compare our FBVE scheme with other existing schemes in the following table 1.

Table 1. Comparison of other proposed protocols vs. ours

Scheme derived from.	Complexity (User)	The number of cloud server	Necessary for cryptographic assumptions
Ref.[6]	$O(\log^2 n)$	2	No
Ref.[14]	$O(3(\log n-1)/2)$	1	No
Ref.[15]	$O(n+t)$	2	Yes
Ref.[23]	$O(\log^2 n)$	2	No
Ours.	$O(1)$	1	No

In summary, our scheme FBVE is efficient, verifiable, and secure outsourcing scheme of fixed base-variable exponent modular exponentiation.

3.2 Secure Outsourcing Scheme of Variable Base-Variable Exponent Exponentiations

In this section, we propose a new secure outsourcing scheme (**Exp**) for modular exponentiations using one untrusted cloud server. In **Exp**, we implement a secure outsourcing

of variable base-variable exponent modular exponentiations by invoking our scheme **FBVE** (on each invocation **FBVE**, the user T requires $O(1)$ multiplications).

Let $a \in z_q^*$ and $u \in z_p^*$ such that $u^q = 1 \bmod p$. Suppose that T wants an untrusted cloud server CS to compute the exponentiation $u^a \bmod p$ for some base $u \in z_p^*$ and exponent a . The goal of the user T is to compute u^a with less computation cost. T needs to ask for help from cloud server who has enough computing power. The secure outsourcing scheme for variable base-variable exponent exponentiations (**Exp**) is described as follows:

Step1. T runs **FBVE** twice to obtain two pairs (α, g^α) and (β, g^β) . α, g^α and β are secret, g^β is public.

Step2. T computes $t = \alpha / \beta$ and $w = u / g^\alpha$. Thus, we have

$$u^a = (g^\alpha w)^a = g^{\alpha a} w^a = g^\alpha \bullet (g^\beta)^{ta-t} \bullet (w^a) .$$

Step3. T requests CS to compute $(g^\beta)^{ta-t}$ and (w^a) by invoking our scheme **FBVE**. That is, T runs $FBVE(ta-t, g^\beta) \to (g^\beta)^{ta-t}$ and $FBVE(a, w) \to (w^a)$.

Step4. T concludes that the cloud server CS is dishonest and aborts the protocol if FBVE scheme could not be performed correctly in step3. Otherwise, T computes

$$u^a = g^\alpha \bullet FBVE(ta-t, g^\beta) \bullet FBVE(a, w) .$$

Theorem 2. In a single-cloud server model, our scheme **Exp** is efficient and provably secure outsourcing scheme of variable base-variable exponent modular exponentiations.

Proof.

(1)Correctness. The correctness of the scheme Exp is obvious. If the cloud honestly implements the scheme, it will certainly make users to accept its output.

(2)Soundness. The cloud server CS involved in computing only in step3. The correct execution of Setp3 is completely dependent on our scheme **FBVE**. We have analyzed the soundness of our scheme **FBVE** in section **3.1**. Thus, the scheme **Exp** meets requirements of the soundness.

(3) Privacy. The outsourcing scheme does not leak a , u and u^a . We have proved that **FBVE** can hide exponent effectively in section 3.1. It is impossible that an attacker wants to recover a during computing (w^a) and $(g^\beta)^{ta-t}$ by invoking FBVE. If the attacker wants to obtain u by the equation $u = wg^\alpha$, it needs to get w and g^α at the same time. The probability of retrieving g^α is negligible because g^α is secret. In addition, if the attacker wants to obtain u^a , it needs to obtain $FBVE(ta-t, g^\beta)$, $FBVE(a, w)$ and g^α at the same time. This is infeasible because our scheme FBVE can ensure the privacy of $FBVE(ta-t, g^\beta)$ and $FBVE(a, w)$. Meanwhile, g^α is secret.

Let X be the event of "attackers successfully recover some private data". Thus, the probability of X is negligible.

(4) Verifiability. It is assumed that cloud servers are bribed by a *PPT* attacker, and then the *PPT* attacker will access all information that the cloud servers receives and the final result of computing by the cloud servers. Similar to our scheme **FBVE**, X denotes that the event of "attackers forge $FBVE(ta-t, g^\beta) \to (g^\beta)^{ta-t}$ and successfully cheat users", Y denotes that the event of "attackers forge $FBVE(a, w) \to (w^a)$ and successfully cheat users", Z denotes that the event of "attackers forge $u^a = g^\alpha \bullet FBVE(ta-t, g^\beta) \bullet FBVE(a, w)$ and successfully cheat users". According to our scheme **FBVE**, we have

$$\Pr[X] \le \frac{(m^2-6)!}{m^2!m^2(m^2-1)} + 2(\frac{1}{m^2(m^2-1)})^3 \text{ and } \Pr[Y] \le \frac{(m^2-6)!}{m^2!m^2(m^2-1)} + 2(\frac{1}{m^2(m^2-1)})^3 .$$

Thus, $\Pr[Z] \le \{\frac{(m^2-6)!}{m^2!m^2(m^2-1)} + 2(\frac{1}{m^2(m^2-1)})^3\}^2 = 1/O(m^{24})$. In other words, the user T will detect the error output u^a with probability no less than $1-\{\frac{(m^2-6)!}{m^2!m^2(m^2-1)} + 2(\frac{1}{m^2(m^2-1)})^3\}^2$. We noticed that the checkability is approximately equal to one when the matrix size m is a big number.

(5) Efficiency. In general, u^a can be computed by user with O(n) multiplications, where n is the bit length of exponent a. In [6], Hohenberger and Lysyanskaya firstly proposed a secure outsourcing scheme for variable base-variable exponent modular exponentiation, which outsourced modular exponentiations to two non-colluding untrusted cloud servers. With their techniques, *Rand1* was called many times (each invocation *Rand1*, the user T requires $O(\log^2 n)$ multiplications). Although exponentiations can be computed by the user with $O(\log^2 n)$ multiplications and the computation complexity is not changed, but the computation cost of the user T was still large. In [23], Chen et al. proposed a new secure outsourcing algorithm for modular exponentiations by using two non-colluding untrusted cloud servers. Compared with the algorithm [6], their algorithm was superior in both efficiency and checkability, but exponentiations can be computed by the user with $O(\log^2 n)$ multiplications, and the scheme had checkability with only 2/3. In our scheme **Exp**, we only need one cloud server. Compared with [6] and [23], our scheme is more efficient. Similar to [23], we denote by **MM** a modular multiplication, by **MInv** a modular inverse, and by **Invoke(*Rand1*)** an invocation of the subroutine *Rand1*. We omit other operations such as modular additions. In the following table2, we compare our scheme **Exp** with other existing schemes.

Table 2. Comparison of other proposed protocols vs. ours. Exp

Scheme derived from.	Ref.[6]. Exp	Ref.[23]. Exp	Ours.
MM	9	7	$O(1)$
MInv	5	3	2
Invoke(*Rand1*)	6	5	0
The number of cloud server	2	2	1
Checkability	1/2	2/3	$\geqslant$1-1/O(m^{24})

In summary, our scheme **Exp** is efficient and secure outsourcing scheme of variable base-variable exponent exponentiation.

3.3 Secure Outsourcing Scheme of Simultaneous Exponentiation

In order to improve efficiency, we propose a new secure outsourcing scheme (**Sexp**) for simultaneous modular exponentiations $u_1^a u_2^b$ using one untrusted cloud server. Compared with the scheme in[23], our scheme **Sexp** is superior in both efficiency and checkability.

Let $a, b \in z_q^*$ and $u_1, u_2 \in z_p^*$, such that $u_1^q = 1 \bmod p, u_2^q = 1 \bmod p$. Suppose that T requests an untrusted cloud server CS to compute the exponentiations $u_1^a u_2^b \bmod p$. The untrusted cloud server CS does not know anything about a, b, u_1, u_2 and $u_1^a u_2^b \bmod p$. The goal of the user T is to compute $u_1^a u_2^b \bmod p$ with less computation cost. T needs to ask for help from the untrusted cloud server CS who has enough computing power. The secure outsourcing scheme of simultaneous modular exponentiations $u_1^a u_2^b$ (**Sexp**) is described as follows.

Step1. T runs our scheme **FBVE** twice to obtain two pairs (α, g^α) and (β, g^β). α, g^α and β

are secret, and g^{β} is public.

Step2. T computes $t = \alpha / \beta$, $w_1 = u_1 / g^{\alpha}$ and $w_2 = u_2 / g^{\alpha}$. Thus, we have

$u_1{}^a u_2{}^b = (g^{\alpha} w_1)^a (g^{\alpha} w_2)^b = (g^{a\alpha+b\alpha}) w_1{}^a w_2{}^b = g^{\alpha} \cdot (g^{\beta})^{(a+b)t-t} \cdot w_1{}^a \cdot w_2{}^b$.

Step3. T requests CS to compute $(g^{\beta})^{t(a+b)-t}$, $w_1{}^a$ and $w_2{}^b$ by invoking our proposed FBVE scheme. That is, T runs

$FBVE(t(a+b)-t, g^{\beta}) \to (g^{\beta})^{t(a+b)-t}$, $FBVE(a, w_1) \to (w_1{}^a)$ and $FBVE(b, w_2) \to (w_2{}^b)$.

Step4. T concludes that the cloud server CS is dishonest and aborts the protocol if FBVE scheme could not be correctly performed in step3. Otherwise, T computes

$$u_1{}^a u_2{}^b = g^{\alpha} \bullet FBVE(t(a+b)-t, g^{\beta}) \bullet FBVE(a, w_1) \bullet FBVE(b, w_2).$$

Theorem 3. In a single-cloud server model, our scheme **Sexp** is efficient and provably secure.

Proof. Similar to theorem 1 and theorem 2, we can easy prove theorem 3, so we won't go into detail here. We compare our scheme **Sexp** with the existing scheme [23] in the following table3.

Table 3. Comparison of other proposed protocols vs. ours. Sexp

Scheme derived from.	Ref.[6].Exp	Ref.[23].Sexp	Ours.
MM	9	10	$O(1)$
MInv	5	4	2
Invoke(*Rand1*)	6	5	0
The number of CS	2	2	1
Checkability	1/2	1/2	1-1/$O(m^{36})$

Compared with [23], the weakness of our scheme **Sexp** is increased MM, but our scheme **Sexp** requires only *2* MInv, *0* invocation of *Rand1*, and one untrusted cloud server. Moreover, the checkability is approximately equal to one when the matrix size m is a big number in our scheme **Sexp**.

In summary, our scheme **Sexp** is efficient and provably secure outsourcing scheme of variable base-variable exponent simultaneous modular exponentiations using one untrusted cloud server.

4. Application

In this section, we will give two applications for our proposed outsourcing schemes, one is a secure outsourcing scheme for Cramer-Shoup encryption (**CSES**), and the other is a secure outsourcing scheme for Schnorr signature (**SSS**).

4.1 Secure Outsourcing Scheme for Cramer-Shoup Encryption

The secure outsourcing scheme for Cramer-Shoup encryption(**CSES**) consists of System Parameters Generation, Key Generation, Encryption and Decryption.

System Parameters Generation: Let G be an abelian group of a large prime order q, g be a generator of G. Define a cryptographic secure hash function $H: G^3 \to Z_q$. The system parameters are $SP = (\mathrm{G}, \mathrm{g}, \mathrm{q}, \mathrm{H})$.

Key Generation: Randomly selected three numbers $w, y, z \in Z_q^*$. The secret key is $SK = (\mathrm{w}, \mathrm{y}, \mathrm{z})$, and the public key is $PK = (\mathrm{W}, \mathrm{Y}, \mathrm{Z}) = (\mathrm{g}^w, \mathrm{g}^y, \mathrm{g}^z)$.

Encryption: T inputs a message m and the public key $PK = (\mathrm{W}, \mathrm{Y}, \mathrm{Z})$. Then, T generates the ciphertext C as follows:

Step1. T requests CS to compute $FBVE(\mathrm{k}, g) \to r = g^k$ by invoking our scheme **FBVE**.

Step2. T requests CS to compute $FBVE(\mathrm{k},\mathrm{W}) \to s = W^k$ and $FBVE(\mathrm{k},\mathrm{Z}) \to t = \mathrm{Z}^k$ by invoking our scheme **FBVE**.

Step3. T computes $e = mt, h = H(r,s,e)$.

Step4. T requests CS to compute $FBVE(\mathrm{kh},\mathrm{WY}) \to \gamma = (WY)^{kh}$ by invoking our scheme **FBVE**.

Step5. T outputs the ciphertext $C = \{r,s,e,\gamma\}$.

Decryption: T inputs the ciphertext $C = \{r,s,e,\gamma\}$ and the secret key $SK = (\mathrm{w},\mathrm{y},\mathrm{z})$, and generates the plaintext m as follows:

Step1. T computes $h = H(r,s,e)$.

Step2. T requests CS to compute $Exp(wh + yh, r) \to \varphi$ by invoking our scheme **Exp**.

Step3. If and only if $\varphi = \gamma$, T requests CS to compute $Exp(z,\gamma) \to t$ by invoking our scheme **Exp**, then T computes $et^{-1} \to m$ and outputs the plaintext m.

Comparing to the outsourcing schemes for Cramer-Shoup encryption in [6] and [23], our scheme **CSES** has many advantages which are described in the following table4.

Table 4. Comparison of other proposed protocols vs. ours

Scheme derived from.	Ref.[6]	Ref.[23]	Ours.
Invoke(Exp)	6	7	2
The number of cloud server	2	2	1
Invoke(*Rand1*)	1	1	0

To be emphasized, we do not use the scheme **Exp** in the encryption stage (note that the user not only requires $O(\mathrm{l})$ multiplications, but also requires 2 MInv in **Exp**), while our scheme requires 3 invocation of **FBVE** (note that the user requires only $O(1)$ multiplications in **FBVE**) for encryption.

4.2 Secure outsourcing scheme for Schnorr signature

The secure outsourcing scheme for schnorr signature consists of System Parameters Generation, Key Generation, Signature Generation and Signature Verification.

System Parameters and Key Generation: The system parameters are $SP = (\mathrm{p},\mathrm{g},\mathrm{q},\mathrm{H})$, where p and q are two large primes such that $q \mid (p-1)$, $g \in Z_q^*$ such that $g^q = 1 \bmod p$. Define a cryptographic secure hash function $H : G^3 \to Z_q$. The signing/verification key pair(x, y), where $y = g^x \bmod p$.

Signature Generation: T inputs a message m and the signing key x. Then, T generates the Signature σ as follows:

(1) T requests CS to compute $FBVE(\mathrm{k},g) \to r = g^k$ by invoking our scheme **FBVE**.

(2) T computes $e = H(m \| r)$ and $s = k + xe \bmod q$.

(3) T outputs the signature $\sigma = (r,s)$.

Signature Verification: T inputs $\sigma = (e,s)$, y and m. Meanwhile, T verify the signature $\sigma = (e,s)$ by invoking our scheme **FBVE** as follows:

(1) T requests CS to compute $e = H(m \| r)$, $FBVE(s,g) \to \varphi_1$ and $FBVE(e,y) \to \varphi_2$.

(2) T accepts the signature σ if and only if $\varphi_1 = r\varphi_2$.

Comparing to the outsourcing schemes for schnorr signature in [6] and [23], our scheme **SSS** has many advantages which was described in the following table5.

Table 5. Comparison of other proposed protocols vs. ours

	Ref.[6]	Ref.[23]	Ours.
Invoke(Exp)	2	2	0
The number of cloud server	2	2	1
Invoke(rand1)	1	1	0

To be emphasized, we do not use the subroutine ***Rand1*** and do not invoke the scheme **Exp** in our scheme **SSS** (for each invocation ***Rand1,*** user requires $O(\log^2 n)$ **MM**), while our scheme requires 3 invocation of **FBVE**(the total computation complexity of the user is $O(1)$ in our scheme **FBVE**) for signature verification. Thus, our scheme reduces the computation cost of the user substantially, which improves verification efficiency to a certain extent.

5. Conclusion

In this paper, we firstly presented three secure outsourcing schemes which enable users to securely outsource the computations of modular exponentiations to one untrusted cloud server. The first one is a secure outsourcing scheme for fixed base-variable exponent exponentiation (**FBVE**), the second is for variable base- variable exponent exponentiations (**Exp**), and the third is for simultaneous exponentiations (**Sexp**). User needs only $O(1)$ **MM** in our scheme **FBVE**. In our scheme **Exp** and **Sexp,** user needs $O(1)$ **MM** *and* two **MInv**.Compared with other proposed schemes, our schemes are superior in both efficiency and checkability. Moreover, our schemes are secure without any cryptographic assumptions, and we also analyzed and proved the efficience and security of our schemes . In a single untrusted cloud server model, the key problem of outsourcing modular exponentiations is that how to further reduce the computation cost of user on the future work.

References

[1] S. Microsystems, "Building customer trust in cloud computing with transparent security", https://www.sun.com/offers/details/sun transparency.xml, 2009.

[2] T. Matsumoto, K. Kato and H. Iami, "Speeding up secret computations with insecure auxiliary devices", CRYPTO 1990. LNCS 403, pp. 497-506, 1990.

[3] Shin-ichi Kawaumura and A. Shimbo, "Fast server-aided secret computation protocols for modular exponentiation", IEEE Journal on selected areas in communications, vol.11, nO. 5, 1993.

[4] P. Beguin and J. J. Quisquater, "Secure acceleration of DSS signatures using insecure server", ASIACRYPT1994. LNCS963, pp. 57-69, 1994.

[5] P. Q. Nguyen, I. E. Shparlinski and J. Stern. Distribution of modular sums and the security of server aided exponentiation. Workshop on Cryptography and Computational Number Theory, 1999.

[6] S. Hohenbergera and A. Lysyanskaya, "How to securely outsource cryptographic computations", TCC 2005. LNCS 3378, pp. 264-282, 2005.

[7] M. Jakobsson and S. Wetzel, "Secure server-aided signature generation", PKC 2001. LNCS 1992, pp. 383-401, 2001.

[8] G. B. Agnew, R. C. Mullin and S. A. Vanstone, "Fast exponentiation in GF (2^n)", CRYPTO 1988. LNCS 330, pp. 251-255, 1988.

[9] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson, "Fast exponentiation with precomputation", EUROCRYPTO 1992. LNCS 658, pp. 200-207, 1992.

[10] V. Boyko and M. Peinado. Speeding up discrete log and factoring based schemes via precomputation. EUROCRRYPTO 1998. LNCS 1403, pp. 221-232, 1998.

[11] J. Bos and M. Coster, "Addition chain heuristics", CRYPTO 1989. LNCS 435, pp. 400-407. 1990.

[12] P. de Rooij, "Efficient exponentiation using precomputation and vector addition chains", EUROCRYPTO 1994. LNCS 950, pp.389-399, 1995.

[13] C. H. Lim and P. J. Lee, "More flexible exponentiation with precomputation", CRYPTO 1994. LNCS 839, pp. 95-107, 1994.
[14] M.V. Dijk, D.Clarke, B.Gassend, G. E. Suh, S.D .Srinivas., "Speeding up Exponentiation using an Untrusted Computational Resource", Springer Science-Business Media, Inc.pp. 253-273, 2006.
[15] X. Ma, J.Li, F.G. Zhang, "Efficient and Secure Batch Exponentiations Outsourcing in Cloud Computing", 2012 Fourth International Conference on Intelligent Networking and Collaborative Systems.pp.600-605, 2012.
[16] M. Abadi, J. Feigenbaum, and J. Kilian, "On hiding information from an oracle", Journal of Comput. Syst. 39(1), pp. 21-50, 1989.
[17] D. Beaver and J. Feigenbaum, "Hiding instances in multioracle queries", In Proceedings of STAC '90, pp. 37-48, 1990.
[18] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, "Locally random reductions: Improvements and applications", Journal of Cryptology, 10(1), pp.17 – 36, 1997.
[19] D. Clarke, S. Devadas, M. van Dijk, B. Gassend, and G. E. Suh, "Speeding up Exponentiation using an Untrusted Computational Resource", Technical Report Memo 469, MIT CSAIL Computation Structures Group, August 2003.
[20] P. de Rooij, "On Schnorr's preprocessing for digital signature schemes", Journal of Cryptology, 10(1), pp.1-16, 1997.
[21] C.P. Schnorr, "Efficient identification and signatures for smart cards" In Proceedings of Crypto '89, volume 435 of LNCS, 1989.
[22] C.P. Schnorr, "Efficient signature generation by smart cards", Journal of Cryptography, pp.161-174, 1991.
[23] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New Algorithms for Secure Outsourcing of Modular Exponentiations", ESORICS 2012, LNCS 7459, pp.541–556, 2012. (At the same time, this paper was accepted in "IEEE Transactions on Parallel and Distributed Systems".)
[24] G. Ateniese, B. Demedeiros, "Identity-Based Chameleon Hash and Applications", FC 2004. LNCS, vol. 3110, pp.164-180, 2004.
[25] G. Ateniese, B. Demedeiros, "On the Key Exposure Problem in Chameleon Hashes", SCN 2004. LNCS, vol. 3352, pp.165-179, 2005.
[26] X. Chen, F. Zhang, K. Kim, "Chameleon Hashing Without Key Exposure", ISC 2004. LNCS, vol. 3225, pp. 87-98, 2004.
[27] X. Chen, F. Zhang, W. Susilo, Y. Mu, "Efficient Generic On-Line/Off-Line Signatures without Key Exposure", ACNS 2007. LNCS,vol. 4521, pp. 18-30, 2007.
[28] H. Krawczyk, T. Rabin, "Chameleon hashing and signatures. Proceeding of the 7th Annual Network and Distributed System Security Symposium (NDSS)", pp. 143-154, 2000.
[29] A. Shamir, Y. Tauman, "Improved Online/Offline Signature Schemes", CRYPTO 2001. LNCS, vol. 2139, pp. 355-367, 2001.
[30] G. Brassard, D. Chaum, "Crepeau, C. Minimum disclosure proofs of knowledge.Journal of Computer and System Sciences", vol. 37(2), pp.156-189, 1988.
[31] G. Dicrescenzo, R. Ostrovsky, "On Concurrent Zero-Knowledge with Preprocessing", CRYPTO 1999. LNCS,vol. 1666, pp. 485-502, 1999.
[32] M. Fischlin, R. Fischlin, "Efficient Non-malleable Commitment Schemes", CRYPTO 2000. LNCS, vol. 1880, pp. 413-431, 2000.
[33] R. Gennaro, "Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks", CRYPTO 2004. LNCS, vol. 3152, pp. 220-236, 2004.
[34] J. Garay, P. MacKenzie, K. Yang, "Strengthening Zero-knowledge Protocols Using Signatures", EUROCRYPT 2003. LNCS, vol. 2656, pp. 177-194, 2003.
[35] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press , 1996.