# A Note on Two Outsourcing Algorithms of Modular Exponentiations

Xinlei Qi[1], Yunhai Zheng[2], and Chengliang Tian[2,3]
(Corresponding author: Chengliang Tian)

School of Cyberspace Security, Xi'an University of Posts and Telecommunications[1]
Xi'an 710121, China
College of Computer Science & Technology, Qingdao University[2]
Qingdao 266071, China
Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University[3]
Hangzhou 311121, China
Email: tianchengliang@qdu.edu.cn

(Received Oct. 16, 2021; Revised and Accepted May 6, 2022; First Online May 9, 2022)

## Abstract

Recently, Ren *et al.* presented two algorithms for outsourcing modular exponentiations [*IEEE Transactions on Cloud Computing, 9(1):145–154, 2021*], which aim to make the remote cloud server help resource-constrained clients securely perform expensive modular exponentiations in cryptography. In this note, we show their algorithms are incorrect due to the misuse of the Euler theorem in the verification step. Moreover, we suggest a remedial measure for the two-server algorithm.

*Keywords: Cloud Computing; Computation Outsourcing; Euler's Theorem; Modular Exponentiation*

## 1 Introduction

With the prevalence of cloud computing, outsourcing locally data-intensive activities such as large-scale data storage and heavy computational tasks to a remote resource-abundant cloud server has become a popular computing paradigm [3, 9, 10, 12]. However, the sensitivity of the local client's data and the potential incredibility of the cloud server bring many security concerns to this promising computing paradigm [1,2,14,16,18]. Therefore, how to efficiently and securely outsource various large-scale computations has increasingly attracted researchers' attentions [17]. Currently, due to the extremely large integer operations in public-key cryptography, secure outsourcing of heavy cryptographic computations has become a hot topic [8,13,19].

Given two large primes $p, q$ satisfying $q \mid p-1$, an integer $a \in \mathbb{Z}_q$, and an integer $u \in \mathbb{Z}_p^\star$ such that $u^q \equiv 1 \bmod p$, computing $u^a \bmod p$ is a basic operation in the Digital Signature Standard (DSS) of NIST [11], which needs to perform about $1.5n$ modular multiplication for a $n$ bits exponent. In practice, the bit lengths of $p$ and $q$ could be as large as 3072 and 256, respectively, which results in the modular exponentiation operation being very expensive for local devices with limited computation resources. Consequently, the secure delegation of modular exponentiations has been extensively investigated [4–7, 20, 21]. Among these, Ren *et al.* [15] recently proposed two algorithms for outsourcing modular exponentiation to the cloud servers. In this note, we investigate their algorithms and point out a severe flaw in the verification step. Meanwhile, for the two-server case, we present a suggestion to amend this flaw.

## 2 Review of Ren *et al.*'s Outsourcing Algorithms

### 2.1 Ren *et al.*'s Outsourcing Algorithm with Two Non-colluding Servers

**Preprocessing:** Client $T$ generates three triples $(\alpha, \alpha^{-1}, g^\alpha)$, $(\beta, \beta^{-1}, g^\beta), (t_1, t_1^{-1}, g^{t_1})$, where $\alpha, \beta, t_1 \in \mathbb{Z}_q$, $g \in \mathbb{Z}_p^\star$ and $g^q = 1 \bmod p$. Let $v = g^\alpha$.

**Logical Division:** On inputting the base $u$ and the exponent $a$, the client $T$ first computes $w, \gamma$ such that $u = wv \bmod p$, $\alpha a = \beta + \gamma$. Then, $T$ picks two random numbers $t, h_1 \in \mathbb{Z}_q$, a prime $n$ which is co-prime with $w$, and computes $s, h_2$ such that $a \equiv \varphi(n)t + s \bmod q$, $\varphi(n)t = h_1 + h_2$, where $\varphi(n)$ is the *Euler function* of $n$. Hence, $u^a \bmod p = (wv)^a \bmod p = g^{\alpha a} w^a \bmod p = g^\beta g^\gamma w^a \bmod p = g^\beta g^\gamma w^s w^{h_1} w^{h_2} \bmod p$. Finally, $T$ sends $((\gamma/t_1, g^{t_1}), (h_1, w), (s, w))$ and $((\gamma/t_1, g^{t_1}), (h_2, w), (s, w))$ to the servers $U_1$ and $U_2$, respectively, in a random order.

**Cloud Computing:** Servers $U_1$ and $U_2$ compute $((g^{t_1})^{\gamma/t_1}, w^{h_1}, w^s)$ and $((g^{t_2})^{\gamma/t_2}, w^{h_2}, w^s)$ in $\mathbb{Z}_p^\star$, respectively and return their results to $T$. Just as the authors said in Section 3.1 of [15]: '... the servers output $y^x \bmod p$ when they receive the inputs $(x, y)$'.

**Client Verification and Recovery:** Client $T$ verifies the correctness of the results returned from servers by checking whether the following equations hold.

$$g^\gamma = U_1\left(\gamma/t_1, g^{t_1}\right) = U_2\left(\gamma/t_1, g^{t_1}\right) \quad (1)$$
$$U_1(s, w) = U_2(s, w) \quad (2)$$
$$w^{h_1}w^{h_2} = w^{\varphi(n)t} \equiv 1 \bmod n \quad (3)$$

If they hold, $T$ recovers $u^a = g^\beta g^\gamma w^s w^{h_1} w^{h_2} \bmod p$.

### 2.2 Ren *et al.*'s Outsourcing Algorithm with Single Server

**Preprocessing:** Client $T$ first generates a set of blinding pairs $\{(\alpha, g^\alpha, g^{-\alpha}), (\beta, g^\beta), (\theta_x, g^{\theta_x}), (t_x^{-1}, g^{t_x}), (\xi_j, g^{-\xi_j}), \mu_j, k, g^{\Sigma_{i=1}^k \xi_i\mu_i}, g^{\Sigma_{i=k+1}^b \xi_i\mu_i}\}$ for some chosen positive integer $b$ and $x = 1,2,3,4, j = 1,2,\cdots,b+3, k \in \{2,\cdots,b-2\}$. Let $v = g^\alpha$.

**Logical Division:** On inputting the base $u$ and the exponent $a$, the client $T$ first computes $w, \gamma$ such that $u = wv \bmod p$, $\alpha a = \gamma + \beta$. Then, $T$ randomly chooses $t_5 \in \mathbb{Z}_q$ and a prime $n_1$, which is co-prime with $w$, and computes $s$, $r_1$, $r_2$ and $w_j$ such that $a = \varphi(n_1)t_5 + s \bmod q$, $r_1 = \varphi(n_1)t_5 - \sum_{i=1}^k \mu_i$, $r_2 = \varphi(n_1)t_5 + s + \sum_{i=k+1}^b \mu_i$, and $w_j = wg^{-\xi_j}, j = 1,\cdots,b+3$, where $\varphi(n_1)$ denotes the $Euler function$ of $n_1$. Finally, $T$ randomly chooses a prime $n_2$, which is relatively prime with $g$ and computes $t_6, t_7, t_8, t_9$ such that $\gamma \equiv \varphi(n_2)t_6 + \theta_1 \bmod q, s\xi_{b+1} \equiv \varphi(n_2)t_7 + \theta_2 \bmod q$, $r_1\xi_{b+2} \equiv \varphi(n_2)t_8 + \theta_3 \bmod q, r_2\xi_{b+3} \equiv \varphi(n_2)t_9 + \theta_4 \bmod q$, where $\varphi(n_2) = n_2 - 1$ is the $Euler function$ of $n_2$. Now,

$$\begin{aligned} u^a &= (wv)^a \\ &= w^a g^{\alpha a} \\ &= w^{\varphi(n_1)t_5+s} g^\gamma g^\beta \\ &= w^{\varphi(n_1)t_5} w^s g^{\varphi(n_2)t_6+\theta_1} g^\beta \\ &= w^{\varphi(n_1)t_5} w_{b+1}^s g^{s\xi_{b+1}} g^{\varphi(n_2)t_6+\theta_1} g^\beta \\ &= w^{\varphi(n_1)t_5} w_{b+1}^s g^{\varphi(n_2)t_7+\theta_2} g^{\varphi(n_2)t_6+\theta_1} g^\beta. \end{aligned}$$

Hence, $T$ sends $((\varphi(n_2)t_6 + \theta_1)/t_1, g^{t_1})$, $((\varphi(n_2)t_7 + \theta_2)/t_2, g^{t_2})$, $((\varphi(n_2)t_8 + \theta_3)/t_3, g^{t_3})$, $((\varphi(n_2)t_9+\theta_4)/t_4, g^{t_4})$, $(\mu_j, w_j)$, $(s, w_{b+1})$, $(r_1, w_{b+2})$, and $(r_2, w_{b+3})$ to the server $U$ in a random order, where $j = 1,2,\cdots,b$.

**Cloud Computing:** The server $U$ computes $\eta_1 = (g^{t_1})^{(\varphi(n_2)t_6+\theta_1)/t_1}$, $\eta_2 = (g^{t_2})^{(\varphi(n_2)t_7+\theta_2)/t_2}$, $\eta_3 = (g^{t_3})^{(\varphi(n_2)t_8+\theta_3)/t_3}$, $\eta_4 = (g^{t_4})^{(\varphi(n_2)t_9+\theta_4)/t_4}$ and $w_j^{\mu_j}, w_{b+1}^s, w_{b+2}^{r_1}, w_{b+3}^{r_2}$ in $\mathbb{Z}_p^\star$ for $j = 1,\cdots,b$, and returns the results to $T$.

**Client Verification and Recovery:** $T$ verifies the correctness of the results from $U$ by checking whether the following equations hold.

$$\eta_x \equiv g^{\theta_x} \bmod n_2, x = 1,2,3,4 \quad (4)$$

$$w_{b+2}^{r_1}\left(\prod_{i=1}^k w_i^{\mu_i}\right) \cdot g^{\sum_{i=1}^k \xi_i\mu_i}\eta_3 = w^{\varphi(n_1)t_s} \equiv 1 \bmod n_1 \quad (5)$$

$$w_{b+3}^{r_2}\eta_4 \equiv \left(\prod_{i=k+1}^b w_j^{\mu_i}\right) \cdot g^{\sum_{i=k+1}^b \xi_i\mu_i} w_{b+1}^s \eta_2 \bmod n_1. \quad (6)$$

If they hold, $T$ recovers $u^a \equiv w^{\varphi(n_1)t_5} w_{b+1}^s \eta_1\eta_2 g^\beta \bmod p$.

## 3 Analysis and Revision

### 3.1 Analysis of the Algorithm with Two Servers

As mentioned in Section 3.1 of [15], the servers output $y^x \bmod p$ when they receive the inputs $(x, y)$. If the servers are honest, the returned results can pass the verification Equation (1) and Equation (2). The verification Equation (3) is from the speculation $w^{\varphi(n)t} = w^{h_1+h_2} = w^{h_1}w^{h_2} \equiv 1 \bmod n$. However, after sending pairs $(h_1, w), (h_2, w)$ to servers, the values $w^{h_1}, w^{h_2}$ are computed not in $\mathbb{Z}$, but in $\mathbb{Z}_p^\star$. Since, generally, $w^{h_1} \bmod p \cdot w^{h_2} \bmod p \neq 1 \bmod n$, the verification Equation (3) doesn't hold. That is, even if the servers $U_1$ and $U_2$ perform the specified computation task honestly, the client $T$ will reject their results. We illustrated this flaw with the toy Example 1 in the appendix. We illustrate the algorithm's incorrectness with the following toy example.

**Example 1.** *Let $q = 3$, $p = 7$, $u = 4$, $a = 2$.*

- *Client $T$ chooses the parameters $g = 4$, $\alpha = 2$, $\beta = 2$, $t_1 = 2$, and precomputes $g^\alpha = 2$, $g^\beta = 2$, $g^{t_1} = 2$, and $t_1^{-1} = 2$. Let $v = g^\alpha = 2$.*
- *On inputting $(u, a) = (4, 2)$, the client $T$ first computes $w = uv^{-1} = 2$ and $\gamma = \alpha a - \beta = 2 \cdot 2 - 2 = 2$. Then, $T$ chooses $t = 2$, $h_1 = 1$ and a prime $n = 3$, and computes $s = a - \varphi(n)t \bmod 3 = 1$, $h_2 = \varphi(n)t - h_1 = 3$. Finally, $T$ sends $(1,2)$, $(1,2)$ and $(1,2)$ to the server $U_1$, and sends $(1,2)$, $(3,2)$ and $(1,2)$ to the server $U_2$.*
- *Server $U_1$ returns $g^\gamma = w^{h_1} = w^s = 2 \bmod 7$, $U_2$ returns $g^\gamma = w^s = 2 \bmod 7, w^{h_2} = 1 \bmod 7$.*
- *The client $T$ verifies $w^{h_1}w^{h_2} = 2 \cdot 1 \neq 1 \bmod 3$, and thus, rejects the results.*

A natural idea to circumvent this flaw is that client $T$ inquiries the value of $y^x$ in $\mathbb{Z}$ or $\mathbb{Z}_n^\star$ instead of in $\mathbb{Z}_p^\star$.

1) If in $\mathbb{Z}$, for honest servers, the client can obtain the correct result. However, it is impractical. In practice, $w$ can be as large as 3072 bits [11] and, to be against exhaustive attack, $h_i$ should be as large as 64 bits. Hence, $w^{h_i}$ could be an integer with $3072 \cdot 2^{64}$bits$\approx 2^{32}$TB. Such huge a number is impossible to store and handle for a resource-constrained client, even for a resource-abundant server.

2) If in $\mathbb{Z}_n^\star$, although the result returned from an honest cloud can pass the verification, the client $T$ can not recover the correct result which should be calculated in $\mathbb{Z}_p^\star$. Also, this can be easily illuminated with the following toy example.

**Example 2.** *Let $q = 3$, $p = 7$, $u = 4$, $a = 2$.*

- *Client $T$ chooses the parameters $g = 4$, $\alpha = 2$, $\beta = 2$, $t_1 = 2$, and precomputes $g^\alpha = 2$, $g^\beta = 2$, $g^{t_1} = 2$, and $t_1^{-1} = 2$. Let $v = g^\alpha = 2$.*
- *On inputting $(u, a) = (4, 2)$, the client $T$ first computes $w = uv^{-1} = 2$ and $\gamma = \alpha a - \beta = 2 \cdot 2 - 2 = 2$. Then, $T$ chooses $t = 2$, $h_1 = 1$ and a prime $n = 3$, and computes $s = a - \varphi(n)t \bmod 3 = 1$, $h_2 = \varphi(n)t - h_1 = 3$. Finally, $T$ sends $(1, 2)$, $(1, 2)$ and $(1, 2)$ to the server $U_1$, and sends $(1, 2)$, $(3, 2)$ and $(1, 2)$ to the server $U_2$.*
- *Server $U_1$ returns $g^\gamma \bmod n = w^{h_1} \bmod n = w^s \bmod n = 2^1 \bmod 3 = 2$, $U_2$ returns $g^\gamma \bmod n = w^s \bmod n = 2^1 \bmod 3 = 2, w^{h_2} \bmod n = 2^3 \bmod 3 = 2$.*
- *The client $T$ verifies $w^{h_1} w^{h_2} = 2 \cdot 2 = 1 \bmod 3$, and thus, accepts the result. Then $T$ recovers $g^\beta g^\gamma w^s w^{h_1} w^{h_2} \bmod p = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \bmod 7 = 4$. However, the actual result $u^a \bmod p = 4^2 \bmod 7 = 2$. Therefore, $u^a \not\equiv w^{\varphi(n_1)t_5} w_{b+1}^s \eta_1 \eta_2 g^\beta \bmod p$.*

Overall, both of the above simple revisions are infeasible.

## 3.2 Analysis of the algorithm with single server

Similarly, for the single-server algorithm, the verification equations are incorrect due to the following simple observation: by Euler's theorem, $m^{\varphi(n)} \equiv 1 \bmod n$ for some prime $n$ and any integer $m$ with $\gcd(m, n) = 1$, but, in general $m^{\varphi(n)} \bmod p \neq 1 \bmod n$. According to their proposed algorithm, it is easy to verify that $\eta_x = g^{\varphi(n_2)t_{x+5}} g^{\theta_x} \bmod p$, $w_{b+2}^{r_1} \left(\prod_{i=1}^k w_i^{\mu_i}\right) \cdot g^{\sum_{i=1}^k \xi_i \mu_i} \eta_3 = w^{\varphi(n_1)t_s} \bmod p$, and

$$w_{b+3}^{r_2} \eta_4 \left( \left( \prod_{i=k+1}^b w_j^{\mu_i} \right) \cdot g^{\sum_{i=k+1}^b \xi_i \mu_i} w_{b+1}^s \eta_2 \right)^{-1} = w^{\varphi(n_1)t_s} \bmod p.$$

Therefore, based on the above-mentioned observation, the verification Equations (4)-(6) generally fail even for an honest server. See a toy example below.

**Example 3.** *Select $q = 3$, $p = 7$, $u = 4$, $a = 2$*

- *Client $T$ chooses $g = 4$, $b = 4$, $k = 2$, $\alpha = 2$, $\beta = 2$, $t_i = \theta_i = 2$, $\xi_j = \mu_j = 2$, and precomputes $v = g^\alpha = 2$, $g^\beta = 2$, $g^{t_i} = g^{\theta_i} = 2$, $g^{\xi_j} = 2, g^{\sum_{i=1}^k \mu_i \xi_i} = 2$, $g^{\sum_{i=k+1}^b \mu_i \xi_i} = 2$, where $i = 1, 2, 3, 4$, $j = 1, \cdots, 7$.*
- *On inputting $(u, a) = (4, 2)$, client $T$ first computes $w = uv^{-1} = 2$ and $\gamma = \alpha a - \beta = 2$. Then, $T$ chooses $t_5 = 2$, $n_1 = 3$, and computes $s = 1$, $r_1 = 0$, $r_2 = 9$, $w_j = 8$ for $j = 1, \cdots, 7$. Finally, $T$ chooses a prime $n_2 = 3$, computes $t_6 = 0$, $t_7 = 0$, $t_8 = 2$, $t_9 = 2$, and sends $(4, 2)$, $(4, 2)$, $(12, 2)$, $(12, 2)$, $(\mu_j, w_j) = (2, 8)$, $(s, w_{b+1}) = (1, 8)$, $(r_1, w_{b+2}) = (0, 8)$, $(r_2, w_{b+3}) = (9, 8)$ to the server $U$.*
- *Server $U$ computes and returns $\eta_1 = \eta_2 = 2$, $\eta_3 = \eta_4 = 1$, $w_j^{\mu_j} = w_{b+1}^s = w_{b+2}^{r_1} = w_{b+3}^{r_2} = 1$.*
- *The client $T$ verifies the server returned results. Obviously, $\eta_i = g^{\theta_i} \bmod n_2$ for $i = 1, 2$ and $\eta_i \neq g^{\theta_i} \bmod n_2$ for $i = 3, 4$. Meanwhile, $w_{b+2}^{r_1} \left(\prod_{i=1}^k w_i^{\mu_i}\right) \cdot g^{\sum_{i=1}^k \xi_i \mu_i} \eta_3 = 2 \neq 1 \bmod n_1$. Thus, $T$ rejects the corrected results.*

## 3.3 Revision

For the two-server algorithm, we can make a minor adaptation to amend the above-mentioned flaw. In the **Logical Division** step, we adapt the parameter $n$ to be a large prime with the same size as $p$. In the **Cloud Computing** step, the server $U_1$ is required to compute the values of $((g^{t_1})^{\gamma/t_1} \bmod N, w^{h_1} \bmod N, w^s \bmod N)$, and the server $U_2$ is required to compute the values of $((g^{t_2})^{\gamma/t_2} \bmod N, w^{h_2} \bmod N, w^s \bmod N)$, where $N = pn$ and is sent to the servers by the client. In the **Client Verification and Recovery** step, client $T$ verifies the correctness of the results returned from servers by checking:

$$\begin{aligned} U_1\left(\gamma/t_1, g^{t_1}\right) &= U_2\left(\gamma/t_1, g^{t_1}\right) \\ U_1(s, w) &= U_2(s, w), \\ g^\gamma &= U_1\left(\gamma/t_1, g^{t_1}\right) \bmod p, \\ U_1(h_1, w) U_2(h_2, w) \bmod n &= w^{\varphi(n)t} \bmod n \\ &= 1. \end{aligned}$$

If they hold, $T$ recovers

$$u^a = g^\beta g^\gamma U_1(s, w) U_1(h_1, w) U(h_2, w) \bmod p.$$

The correctness of our revised version is from the following basic fact: for any integer $M$, $M \bmod N \bmod p = M \bmod p$, $M \bmod N \bmod n = M \bmod n$. If the servers are honest, then $U_1(\gamma/t_1, g^{t_1}) = U_2(\gamma/t_1, g^{t_1}) = (g^{t_1})^{\gamma/t_1} \bmod N$, $U_1(s, w) = U_2(s, w) = w^s \bmod N$,

$U_1(h_1, w) = w^{h_1} \bmod N$, and $U_2(h_2, w) = w^{h_2} \bmod N$. Hence, $U_1(\gamma/t_1, g^{t_1}) \bmod p = (g^{t_1})^{\gamma/t_1} \bmod N \bmod p = (g^{t_1})^{\gamma/t_1} \bmod p = g^{\gamma}$, and $U_1(h_1, w)U_2(h_2, w) = (w^{h_1} \bmod N \cdot w^{h_2} \bmod N) \bmod n = (w^{h_1}w^{h_2}) \bmod n = w^{\varphi(n)t} \bmod n = 1$. All the verification equations hold. Meanwhile,

$$\begin{aligned} & g^{\beta}g^{\gamma}U_1(s, w)U_1(h_1, w)U(h_2, w) \bmod p \\ = \quad & (g^{\beta}g^{\gamma}(w^s \bmod N)(w^{h_1} \bmod N)(w^{h_2} \bmod N)) \bmod p \\ = \quad & (g^{\beta}g^{\gamma}w^s w^{h_1} w^{h_2}) \bmod p = u^a. \end{aligned}$$

The privacy and the efficiency analysis are essentially the same as that in [15]. It is worth mentioning that, as a byproduct, the revised algorithm can also protect the privacy of the modulo number $p$. The security is based on the hardness of factoring large integers.

# 4 Conclusion

We point out a severe misuse of Euler's theorem in Ren *et al.*'s algorithms, which results in their algorithms incorrect. Moreover, we modify the two-server algorithm to amend this flaw. However, for the single-server algorithm, it may need a fundamental rework.

# Acknowledgments

This study was supported by National Natural Science Foundation of China (61702294), National Development Foundation of Cryptography (MMJJ20170126) and the Open Research Fund of Key Laboratory of Cryptography of Zhejiang Province (ZCL21003).

# References

[1] D. S. AbdElminaam, "Improving the security of cloud computing by building new hybrid cryptography algorithms," *International Journal of Electronics and Information Engineering*, vol. 8, no. 1, pp. 40–48, 2018.

[2] Z. Cao and L. Liu, "A note on two schemes for secure outsourcing of linear programming," *Int. J. Netw. Secur.*, vol. 19, no. 2, pp. 323–326, 2017. [Online]. Available: http://ijns.jalaxy.com.tw/contents/ijns-v19-n2/ijns-2017-v19-n2-p323-326.pdf

[3] M. Chen, C. Liu, and M. Hwang, "Securedropbox: a file encryption system suitable for cloud storage services," in *ACM Cloud and Autonomic Computing Conference, CAC '13, Miami, FL, USA - August 05 - 09, 2013*, S. Hariri and A. Sill, Eds. ACM, 2013, pp. 21:1–21:2. [Online]. Available: https://doi.org/10.1145/2494621.2494642

[4] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *Computer Security – ESORICS 2012*, S. Foresti, M. Yung, and F. Martinelli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 541–556.

[5] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2386 – 2396, 2014.

[6] C. Chevalier, F. Laguillaumie, and D. Vergnaud, "Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions," in *Computer Security – ESORICS 2016*, I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds. Cham: Springer International Publishing, 2016, pp. 261–278.

[7] 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, no. C, pp. 1–13, Dec. 2017. [Online]. Available: https://doi.org/10.1016/j.jcss.2016.11.005

[8] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Theory of Cryptography*, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 264–282.

[9] W. Hsien, C. C. Yang, and M. Hwang, "A survey of public auditing for secure data storage in cloud computing," *Int. J. Netw. Secur.*, vol. 18, no. 1, pp. 133–142, 2016. [Online]. Available: http://ijns.jalaxy.com.tw/contents/ijns-v18-n1/ijns-2016-v18-n1-p133-142.pdf

[10] M. Hwang, T. Sun, and C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *J. Circuits Syst. Comput.*, vol. 26, no. 5, pp. 1 750 072:1–1 750 072:16, 2017. [Online]. Available: https://doi.org/10.1142/S0218126617500724

[11] C. F. Kerry and P. D. Gallagher, "Digital signature standard (dss)," *FIPS PUB*, pp. 186–4, 2013.

[12] C. Liu, W. Hsien, C. C. Yang, and M. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing," *Int. J. Netw. Secur.*, vol. 18, no. 4, pp. 650–666, 2016. [Online]. Available: http://ijns.jalaxy.com.tw/contents/ijns-v18-n4/ijns-2016-v18-n4-p650-666.pdf

[13] L. Liu and Z. Cao, "A note on "efficient algorithms for secure outsourcing of bilinear pairings"," *International Journal of Electronics and Information Engineering*, vol. 5, no. 1, pp. 30–36, 2016.

[14] P. S. Masoumeh Zareapoor and M. A. Alam, "Establishing safe cloud: Ensuring data security and performance evaluation," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 88–99, 2014.

[15] Y. Ren, M. Dong, Z. Qian, X. Zhang, and G. Feng, "Efficient algorithm for secure outsourcing of modular exponentiation with single server," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 145–154, 2021.

[16] M. A. D. Saeid Rezaei and M. Bayat, "A lightweight and efficient data sharing scheme for cloud computing," *International Journal of Electronics and Information Engineering*, vol. 9, no. 2, pp. 115–131, 2018.

[17] Z. Shan, K. Ren, M. Blanton, and C. Wang, "Practical secure computation outsourcing: A survey," *Acm Computing Surveys*, vol. 51, no. 2, pp. 1–40, 2018.

[18] J. Singh, "Cyber-attacks in cloud computing: A case study," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 78–87, 2014.

[19] C. Tian, J. Yu, H. Zhang, H. Xue, C. Wang, and K. Ren, "Novel secure outsourcing of modular inversion for arbitrary and variable modulus," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 241–253, 2022.

[20] 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 *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Cham: Springer International Publishing, 2014, pp. 326–343.

[21] K. Zhou, M. H. Afifi, and J. Ren, "Expsos: Secure and verifiable outsourcing of exponentiation operations for mobile cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2518–2531, Nov 2017.

# Biography

**Xinlei Qi** received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 2006 and 2009, respectively. He is currently with the School of Cyberspace Security, Xi'an University of Posts and Telecommunications, as a Lecturer. His research interests include privacy preservation and cloud computing security

**Yunhai Zheng** received the B.E. degree in automation from Qingdao University in 2017. He is currently pursuing the M.S. degree in the College of Computer Science and Technology, Qingdao University. Her research interests include cloud computing security, secure computation outsourcing. .

**Chengliang Tian** received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 2006 and 2009, respectively, and the Ph.D. degree in information security from Shandong University, Ji'nan, China, in 2013. He held a post-doctoral position with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing. He is currently with the College of Computer Science and Technology, Qingdao University, as an Associate Professor. His research interests include lattice-based cryptography and cloud computing security.