doi:10.14132/j.cnki.1673-5439.2019.05.012

双服务器模型下模幂运算外包新方案

张彦轩[1,2],李承峻[1],杨欣也[1],王少辉[1,2,3]

1. 南京邮电大学 计算机学院,江苏 南京 210023
2. 南京邮电大学 江苏省无线传感网高技术研究重点实验室,江苏 南京 210023
3. 南京邮电大学 江苏省大数据安全与智能处理重点实验室,江苏 南京 210023

摘要:对形如 $R = u^a \pmod N$ 的模幂运算,现有外包方案设计时通常忽略特定场景对具体参数 R,u,a,N 的隐私性要求,对 4 个参数均进行隐私性保护,导致方案复杂度过高。鉴于此,基于双服务器模型,文中针对不同场景设计了 3 个不同的模幂运算外包方案,方案通过合理的逻辑划分将原始的模幂运算转化为新运算,在保证有效保护特定应用场景中参数的隐私性的同时,用户的可验证概率能达到 1。理论分析和实验仿真均显示相较于现有方案,新方案的用户端效率大幅提高。

关键词:模幂运算;外包方案;隐私性保护;被动攻击

中图分类号:TP309 **文献标志码**:A **文章编号**:1673-5439(2019)05-0089-06

New scheme for outsourcing modular exponentiation with two servers

ZHANG Yanxuan[1,2], LI Chengjun[1], YANG Xinye[1], WANG Shaohui[1,2,3]

1. School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
2. Key Laboratory of Jiangsu High Technology Research for Wireless Sensor Network, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
3. Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Abstract: For the modular exponentiation like, the existing outsourcing schemes often ignore the privacy requirements of specific scenarios and parameters. The designs usually protect the privacy of all four parameters, resulting in too high computational complexity. Based on two-server model, three different modular exponentiation outsourcing schemes are designed for different scenarios, and the original operation is converted into a new scheme through reasonable logical division. The scheme ensures that the privacy parameters can be correctly protected during the whole computing process, and the user verifiable probability can reach 1. The theoretical analysis and simulation results show that the user end cost of new schemes is substantially decreased compared with that of the existing schemes.

Keywords: modular exponentiation; outsourcing scheme; privacy protection; passive attacks

随着大数据的发展,复杂数据的运算需求不断涌现,运算的外包应运而生。作为一种新的计算模式,运算的外包弥补了传统计算模式中客户端用户储存资源少、计算能力弱等各类缺陷[1-2],极大地降低了运算成本。然而,运算的外包也带来了用户安全隐私和数据正确性验证的安全挑战[3]。一方面,用户的隐私数据可能会在外包计算的交互过程中泄露;另一方面,攻击者的恶意干扰与破坏会导致返回的运算结果无效。用户需同时具备加密隐私数据和验证结果正确性的能力才能确保自身安全。目前,

收稿日期:2019-03-30;**修回日期**:2019-05-23 **本刊网址**:http://nyzr.njupt.edu.cn
基金项目:国家自然科学基金(61373006,61672016,61872192)资助项目
作者简介:张彦轩,男,硕士研究生;王少辉(通讯作者),男,博士,副教授,wangshaohui@njupt.edu.cn
引用本文:张彦轩,李承峻,杨欣也,等. 双服务器模型下模幂运算外包新方案[J]. 南京邮电大学学报(自然科学版),2019,39(5):89-94.

外包计算方案的设计根据服务器数目分为单服务器和双服务器两种模型。通常假设单服务器不可信,而双服务器模型中,至多只有一个服务器不可信。不可信服务器能对获取的用户数据进行篡改,计算返回结果也通常失真。现阶段外包的复杂计算主要包括双线性对运算[4]、大矩阵运算[5]、模幂运算等。

Hohenberger 等[6]首次提出了基于非同谋双服务器模幂运算的外包协议实现,给出了可验证外包方案的形式化定义,方案对模幂运算底数和指数的隐私性均进行了保护,但计算结果的可验证概率仅为 1/2。Chen 等[7]采用文献[6]的安全模型并改进了模幂运算算法,方案在双服务器模型下的可验证概率提高至 2/3。2013 年,Ma 等[8]提出了一个高可验证性的模幂运算外包计算方案,但方案在验证阶段需要用户进行大量模指数运算,验证效率较低。Ye 等[9]给出双服务器模型下模幂运算问题一种新的逻辑划分方式,但方案被攻击成功的概率有1/20,用户信息泄露风险较高。为消除双服务器非同谋的假设,文献[10]提出单服务器下模幂运算外包方案,但方案中需要用户进行本地指数运算,效率提升有限。2017 年,Ding 等[11]提出了单服务器下模幂运算外包方案,方案的可验证概率达到 119/120,但由于采用大量标量乘法,实际效率并不高。Zhou 等[12]提出了单服务器模型下有限群的模幂运算外包计算方案,其可验证概率接近于 1,但方案可验证性受限于安全参数,参数选择过小会导致可验证性下降。

实际通信中,由于存在来自恶意第三方的窃听攻击,用户与云服务器之间的交互往往是不安全的。通过监听特定信道或节点,恶意第三方可提取用户隐私信息。现有的双服务器模幂运算外包方案[6-12]均无法抵御被动窃听攻击。同时,现有外包方案[7,11,13]的设计通常对所有的相关参数均进行机密性保护,这往往超出实际应用的安全要求,如利用 RSA 算法执行加密时,并不需要对加密指数和密文提供保护。针对这两个问题,本文基于双服务器模型提出了新的模幂运算外包方案。对形如 $R = u^a(\bmod N)$ 的模幂运算,针对不同的应用场景,对数据的隐私度进行区分,用户可以对底数 u、指数 a 和结果 R 选择不同的机密性保护。在所提算法中,若存在服务器不诚实,用户均能以概率 1 检测出错误。方案同时能抵御外部攻击者的被动窃听攻击,攻击者无法通过侦听用户与服务器间的信息获知用户隐私。

1 预备知识

本节首先介绍方案涉及的定理,接着描述外包方案的系统模型,最后给出外包方案设计需要达到的安全目标。

1.1 安全性定义

本文方案假设模数 N 为分解困难的大合数,下述定理在文献[12]中有严格证明,这里不再赘述。

定理 1[12] 若 $N = p_1 \cdot p_2 \cdot \cdots \cdot p_m$,且 $p_1, p_2, \cdots, p_m$ 为互不相同的质数,那么有以下结论成立:

$$u^{a+k\phi(N)} = u^a(\bmod N)$$

其中,k 为一个随机选取的正整数,$\phi(N)$ 为欧拉函数,u 和 N 互素。

1.2 系统模型与安全目标

图 1 给出基于双服务器模型的模幂运算外包方案的系统模型,方案主要包含三方:用户 T、可信服务器 U_1 和恶意服务器 U_2。当用户需要进行复杂模幂运算操作时,由于其计算能力受限,往往会将此部分运算交与服务器。针对不同的应用场景,用户对运算 $R = u^a(\bmod N)$ 的外包会有不同的安全需求,如只隐藏模数 a,或需要对计算结果 R 和模数 a 都进行机密性保护。模幂运算外包方案的具体流程如下:

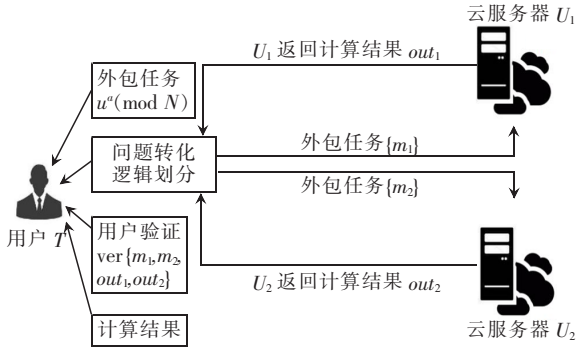

图 1 模幂运算外包方案系统模型

(1) 逻辑划分:用户 T 根据不同的安全需求,重新对计算任务进行逻辑划分,并将所需隐私信息隐藏于本地;

(2) 与服务器交互:用户整理可公开的信息,将隐藏之后的计算任务发送给 U_1 和 U_2,U_1 和 U_2 在分别收到计算任务 $\{m_1\}$ 和 $\{m_2\}$ 后,计算并返回结果 $\{out_1\}$ 和 $\{out_2\}$ 给用户 T;

(3) 验证:根据 U_1 和 U_2 的返回结果,用户对结果的正确性进行验证。若正确继续下一步,否则输

出错误信息;

(4) 计算结果:若验证通过,用户 T 根据服务器返回结果,得到模幂运算外包的结果。

我们假定模型中至多有一个恶意服务器,并且两个服务器非同谋。诚实服务器会忠实地执行用户方案;而恶意服务器不仅会尝试非法获取用户传输数据,也会篡改并故意返回错误的外包计算结果。因此,为降低恶意服务器带来的安全风险,一个有效的模幂运算外包方案需要满足以下 3 个安全目标:

(1) 隐私性:协议执行过程中,特定场景中的服务器与用户的交互过程不会泄露除公开信息外的任何需要保护的隐私信息;

(2) 可验证性:用户可对双服务器返回的计算结果 $\{out_1\}$ 和 $\{out_2\}$ 进行验证,若外包计算结果错误,方案均保证用户可以概率 1 进行辨别;

(3) 正确性:在用户和双服务器均诚实和正确地执行方案的前提下,用户可以根据返回的计算结果 $\{out_1\}$ 和 $\{out_2\}$ 正确还原模幂运算的结果。

2　外包方案设计

对于模幂运算 $R = u^a \pmod N$,我们重点考虑以下 3 种情况外包方案的设计:

(1) 对底数 u 提供机密性保护,指数 a 和结果 R 均可以公开;

(2) 对指数 a 提供机密性保护,而底数 u 和结果 R 的隐私性不作要求;

(3) 保护指数 a 和结果 R 的机密性,底数 u 的值可以公开。

我们针对不同场景中保护参数的不同,基于双服务器模型提出了 3 个不同的选择性参数隐私保护方案,并讨论了各方案的正确性、可验证性和隐私性。我们均假定服务器 U_1 始终为可信任服务器。

2.1　保护底数 u

以 RSA 加密为例,此时对 $R = u^a \pmod N$ 的运算,用户可以完全公开最终计算结果和指数,而必须对底数 u 提供机密性保护。针对这种情况,我们给出如下的外包方案。

首先用户需进行预计算操作,相关数据可预先存储在用户端。预存数据为随机对 (t, y),其中 $y = g^t \bmod N$,每次协议执行需要 3 个随机对,不妨记做 (t_1, y_1),(t_2, y_2),(t_3, y_3),其中 $g, t_1, t_2, t_3 \in Z_n^+$,而 $y_1 = g^{t_1} \bmod N$,$y_2 = g^{t_2} \bmod N$,$y_3 = g^{t_3} \bmod N$。

(1) 逻辑划分

为隐藏底数 u,不妨记 $u' = u \cdot y_1$,此时有:

$$R' = (u')^a = u^a \cdot y_1^a = u^a \cdot y_2 \cdot g^{t_1 a - t_2} \pmod N = u^a \cdot y_3 \cdot g^{t_1 a - t_3} \pmod N$$

此时信息集合整合如下:

$$SK = \{t_1, t_2, t_3, y_1, y_2, y_3, a, g\}$$

(2) 与服务器交互

用户 T 以随机顺序询问云服务器 U_1 以下内容:

$$U_1(u', a) \rightarrow (u')^a \pmod N$$

$$U_1(g, t_1 a - t_2) \rightarrow g^{t_1 a - t_2} \pmod N$$

其中,$U_1(u', a) \rightarrow (u')^a \pmod N$ 表示向服务器 U_1 发送 (u', a),服务器计算并返回 $(u')^a \pmod N$。同样的,用户 T 以随机顺序询问云服务器 U_2 以下内容:

$$U_2(u', a) \rightarrow (u')^a \pmod N$$

$$U_2(g, t_1 a - t_3) \rightarrow g^{t_1 a - t_3} \pmod N$$

用户 T 计算信息集合分别为:

$$PK_1 = PK_2 = (u')^a \pmod N$$

验证信息分别为:

$$VK_1 = \{g^{t_1 a - t_2} \pmod N\},\ VK_2 = \{g^{t_1 a - t_3} \pmod N\}$$

当云服务器完成所有运算任务时,计算结果将被返回至用户 T。

(3) 验证阶段

用户在接收到双方服务器的返回结果后,计算并验证 $y_3 \cdot g^{t_1 a - t_3}$ 与 $y_2 \cdot g^{t_1 a - t_2}$ 以及 PK_1 与 PK_2 的值是否一致,两者均相同继续下一步,否则输出错误信息。

(4) 计算结果

如果外包计算能通过用户端的验证,那么用户利用上述计算结果得到模幂运算结果 R:

$$R = (u')^a / (y_2 \cdot g^{t_1 a - t_2}) \pmod N$$

正确性:假设服务器均按照协议要求执行操作,则一定可以通过用户的验证,并且:

$$R = (u')^a / (y_2 \cdot g^{t_1 a - t_2}) = u^a \pmod N$$

运算结果正确。

可验证性:假设服务器 U_2 为恶意服务器,其在交互过程中返回的消息为 (U_2^1, U_2^2),为了通过用户的验证,其中之一(不妨为 U_2^1)的值必等于 $(u')^a \pmod N$,而 $y_3 \cdot U_2^2 = y_2 \cdot g^{t_1 a - t_2} = g^{t_1 a}$,即 $U_2^2 = g^{t_1 a - t_3}$ 必须成立,故用户能以概率 1 甄别恶意服务器 U_2 是否返回伪造的虚假计算。

隐私性:此方案中,攻击者可以通过侦听服务器与用户的交互过程,直接获取指数 a 和结果 R 的值。在此前提下,方案需保证底数 u 的隐私性。

攻击者通过信道侦听,可以获得 u' 的值,但由于 y_1 的值存储在用户本地,因其随机性,攻击者并不能从 u' 获得 $u = u'/y_1$ 的值;而参数 $t_1a - t_3$ 和 $t_1a - t_2$ 可以构造 2 个线性方程,但是却有 3 个未知数 t_1, t_2, t_3,故攻击者也不能从中提取 t_1 的信息,进而计算 y_1。因此,本方案可以很好地保护底数 u 的隐私性。

2.2 保护结果 R 和指数 a

本节中,用户外包模幂运算 $R = u^a \pmod N$ 时,不要求对运算底数 u 的隐私性进行保护,而需要对指数 a 和结果 R 提供保护。该场景不适用于要求指数 a 是固定不变的值,如 RSA 解密运算的解密密钥。针对这种情况,我们给出如下的外包方案。

同 2.1 节,用户需在本地提取预存储随机对 $(t_1, y_1), (t_2, y_2), (t_3, y_3)$,其中 $g, t_1, t_2, t_3 \in Z_n^+$,而 $y_1 = g^{t_1} \bmod N, y_2 = g^{t_2} \bmod N, y_3 = g^{t_3} \bmod N$。

(1) 逻辑划分

用户首先随机选择 $k, l \in Z_n$,并计算 $a_1 = a + k \cdot \phi(N), a_2 = a + l \cdot \phi(N)$;然后利用预存随机数 y_1,令 $\mu = u/y_1$,我们可得知 $R = u^a = (y_1 \cdot (u/y_1))^a \pmod N$,则有:

$$R = y_1^a \cdot \mu^a \pmod N = y_2 \cdot g^{t_1a - t_2} \cdot \mu^{a_1} \pmod N = y_3 \cdot g^{t_1a - t_3} \cdot \mu^{a_2} \pmod N$$

此时有信息集合整合如下:

$$SK = \{g, \mu, t_1, t_2, t_3, a, a_1, a_2\}$$

(2) 与服务器交互

用户 T 以随机顺序询问云服务器 U_1 以下内容:

$$U_1(g, t_1a - t_2) \rightarrow g^{t_1a - t_2} \pmod N$$

$$U_1(\mu, a_1) \rightarrow (\mu)^{a_1} \pmod N$$

同样,用户 T 以随机顺序询问云服务器 U_2 以下内容:

$$U_2(g, t_1a - t_3) \rightarrow g^{t_1a - t_3} \pmod N$$

$$U_2(\mu, a_2) \rightarrow (\mu)^{a_2} \pmod N$$

用户得到信息集合为:

$$PK_1 = \{g^{t_1a - t_2} \pmod N\}, PK_2 = \{g^{t_1a - t_3} \pmod N\}$$

验证信息集合为:

$$VK_1 = \{\mu^{a_1} \pmod N\}, VK_2 = \{\mu^{a_2} \pmod N\}$$

(3) 验证阶段

用户对返回值进行验证,若两个服务器返回值满足 $VK_1 = VK_2$ 且 $PK_1 \cdot y_2 = PK_2 \cdot y_3$ 则通过验证,否则输出错误信息。

(4) 计算结果

如果用户验证,则可以计算出模幂结果:

$$R = u^a = y_2 \cdot g^{t_1a - t_2} \cdot \mu^{a_1} \pmod N$$

正确性:若服务器正确执行方案协议,则返回结果必能通过用户的验证,并且由定理 1,我们知道:

$$y_2 \cdot g^{t_1a - t_2} \cdot \mu^{a_1} = g^{t_2} \cdot (u/y_1)^{a + k \cdot \phi(N)} \cdot g^{t_1a - t_2} = (u/y_1)^a \cdot g^{t_1a} = (y_1 \cdot u/y_1)^a = u^a \bmod N$$

可见所提方案满足正确性。

可验证性:假设服务器 U_2 为恶意服务器,其在交互过程中返回的消息为 (U_2^1, U_2^2),为了通过用户的验证,其中之一(不妨为 U_2^1)的值必等于 VK_1,而 $y_3 \cdot U_2^2 = y_2 \cdot g^{t_1a - t_2} = g^{t_1a}$,即 $U_2^2 = g^{t_1a - t_3}$ 必须成立,故用户能以概率 1 甄别恶意服务器 U_2 是否返回伪造的虚假计算。

隐私性:在本方案中,攻击者可以通过侦听服务器与用户的交互过程,获取 μ 的值,由 y_1 的随机性,攻击者并不能获知底数 u 的信息。

对于结果 R,攻击者获得消息 $\{g^{t_1a - t_2}, g^{t_1a - t_3}\}$ 的数据信息,攻击者需要猜测随机数 g^{t_2} 或 g^{t_3} 的值还原结果 R。虽然由定理 1,攻击者可以获得 $g^a = g^{a_1} \pmod N$,但 t_1、t_2、t_3 为用户存储在本地的随机数,因此无法得到 g^{t_2} 或 g^{t_3} 的值。

而对于指数 a,攻击者可以窃听到消息 $\{t_1a - t_2, t_1a - t_3, a + k \cdot \phi(N), a + l \cdot \phi(N)\}$,由于大数分解问题,直接找出 N 的欧拉函数理论上不可能,故未知变量的数目超过了方程的数目,从而攻击者无法获知随机数 t_1、t_2、t_3 或 k, l 的值。

综上,本方案可有效保护模幂运算结果和指数的信息。

2.3 保护指数 a

此时对于 $R = u^a \pmod N$,底数 u 和最终计算结果 R 均可以公开,唯一需要保密的是指数 a。外包方案执行如下:

(1) 逻辑划分

为保护指数 a,用户随机选择 $k, l \in Z_n$,$\phi(N)$ 为模数 N 的欧拉函数,令:

$$a_1 = a + k \cdot \phi(N), a_2 = a + l \cdot \phi(N)$$

此时信息整合如下:$SK = \{u, a, a_1, a_2\}$。

(2) 与服务器交互

用户询问云服务器 U_1 以下内容:

$$U_1(u, a_1) \rightarrow u^{a_1} \pmod N$$

同样,用户 T 向云服务器 U_2 询问以下计算:

$$U_2(u, a_2) \rightarrow u^{a_2} \pmod N$$

得到此时的信息和验证集合为 $VK_1 = PK_1 =$

$\{u^{a_1}\}$,$VK_2 = PK_2 = \{u^{a_2}\}$。

(3) 验证阶段

用户对返回值进行验证,若返回的 $u^{a_1} \bmod N$ 和 $u^{a_2} \bmod N$ 两者值相等则通过验证,否则输出错误信息。

(4) 计算结果

如果结果通过了用户端的验证,那么外包运算的最终结果如下:

$$R = u^a = u^{a_1} (\bmod N)$$

正确性:根据定理 1 易得 $u^a = u^{a_1} (\bmod N)$,即方案的正确性必然成立。

可验证性:在保护指数 a 的外包方案中,验证信息为 $VK_1 = \{u^{a_1} \bmod N\}$,$VK_2 = \{u^{a_2} \bmod N\}$,由定理 1 易得两者相等,因此用户能够以概率 1 判断是否存在恶意服务器伪造运算结果。

隐私性:由于因式分解的困难性,攻击者无法得到 $\phi(N)$ 的值,故其无法通过侦听到的 $a + k \cdot \phi(N)$ 和 $a + l \cdot \phi(N)$ 的值得到指数 a 的信息。

3　性能分析与仿真实验

3.1　性能比较

如表 1 所示,本节中给出本文方案与文献[10,12]的模幂外包方案的性能比较。性能比较主要针对用户端实际运算中最消耗资源的模乘运算 MM、模逆运算 MInv 以及随机数对产生个数 Init(R),通信延迟以及服务器运算效率因实际差异不计入性能统计。

表 1　新方案与外包方案性能比较

方案	MM	MInv	Init(R)	可验证概率
本文 2.1 节	4	1	3	1
本文 2.2 节	2	0	4	1
本文 2.3 节	2	0	2	1
文献[10]	$16 + 1.5\log(rt_1t_2)$	6	6	119/120
文献[12]	$5 + 3\log b$	0	0	$1 - 1/2b^2$

从表 1 可以看出,文献[10]的方案需要用户执行 6 次模逆运算,6 个随机数对,且可验证概率低于本文方案。同文献[12]方案相比,本文设计的复杂度最高的 2.1 方案需要预存随机数,但相应地少做了 $1 + 3\log b$ 次模乘运算,且可验证概率为 1,高于文献[12]中的 $1 - 1/2b^2$,即能保证用户以确定性概率获取外包计算的结果。同理,2.2 节方案和 2.3 方案均优于文献[12]方案。综上所述,本文所提外包计算新方案在安全性和性能方面均有较大提高。

3.2　仿真实验

本节基于 Java 语言对本文方案同直接模幂运算、文献[12]方案进行仿真比较。实验采用 i7 CPU,8 G 内存,Windows10 64 位操作系统的主机环境配置。

此次实验主要针对 128 bit、256 bit、512 bit、1 024 bit 4 类常用指数长度进行测试。考虑到底数 u 的长度一般不会超过指数长度,实验涉及的底数长度同样为 128 bit、256 bit、512 bit 和 1 024 bit,逐次递增。为了避免实验次数过少产生的误差,实验同样采用了多次采样计算平均值的做法,比较复杂度最高的 2.1 节方案和文献[12]的耗时对比,据此给出图 2 和图 3。

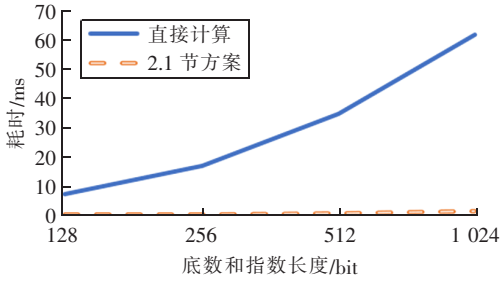

图 2　与直接计算耗时对比

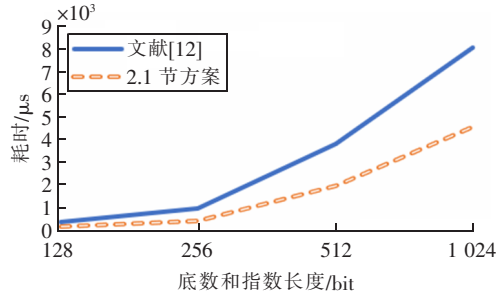

图 3　与文献[12]用户耗时对比

从图 3 中可以明显看出,随着底数长度取值的增大,两种方案用户端的执行时间均有增加,且本文方案与文献[12]间的用户端负载差值也在增大。与直接计算相比,外包运算能最大幅度降低指数对运算效率的影响,使得运行时间不呈线性增长,因此从实验模拟结果可以看出:本文的安全外包方案均优于直接模幂计算和文献[12]所提方案。

4　结束语

本文对基于双服务器模型的模幂运算外包方案的设计进行详细的阐述。方案对模幂运算中的底数和指数进行处理,逻辑转化原有的运算外包问题。根据应用场景的不同,方案中模幂运算的底数、指数和运算结果被执行不同的机密性保护。为保证用户

能正确验证外包计算结果,方案引入随机数转化原有计算方式,使得最终结果的可验证概率为 1。在安全性上,方案能保证被动攻击者无法在获知通信消息的前提下对用户隐私信息进行还原。实验表明,本文方案较现有方案有较大的效率提升。

目前提出的方案基于单恶意服务器的双服务器模型,而实际应用中双服务器可能均为恶意服务器,如何构造高效的可共谋双服务器模型下模幂运算外包方案是下一步考虑的重点研究问题。

参考文献:

[1] 李帅,付安民,苏铠,等. 基于单服务器的群上幂指数安全外包计算方案[J]. 计算机研究与发展,2018,55(11):2482 - 2489.
LI Shuai,FU Anmin,SU Mang,et al. Secure and verifiable protocol for outsourcing group power exponent to a single server[J]. Journal of Computer Research and Development,2018,55(11):2482 - 2489. (in Chinese)

[2] FIORE D,GENNARO R,PASTRO V. Efficiently verifiable computation on encrypted data[C]//ACM Conference on Computer and Communications Security. 2014:844 - 855.

[3] REN K,WANG C,WANG Q. Security challenges for the public cloud[J]. IEEE Internet Computing,2012,16(1):69 - 73.

[4] REN Y,DING N,WANG T,et al. New algorithms for verifiable outsourcing of bilinear pairings[J]. Science China Information Sciences,2016,59:99103.

[5] SHENG G,TANG C,GAO W. An efficient scheme for publicly verifiable computation of outsourced matrix multiplication[C]//International Conference on Network and System Security. 2016:349 - 362.

[6] HOHENBERGER S,LYSYANSKAYA A. How to securely outsource cryptographic computations[C]//Proceedings of the 2nd Theory of Cryptography Conference. 2005:264 - 282.

[7] CHEN X,LI J,MA J,et al. New algorithms for secure outsourcing of modular exponentiations[C]//European Symposium on Research in Computer Security. 2012:541 - 556.

[8] MA X,LI J,ZHANG F. Outsourcing computation of modular exponentiations in cloud computing[J]. Cluster Computing,2013,16(4):787 - 796.

[9] YE J,CHEN X,MA J. An improved algorithm for secure outsourcing of modular exponentiations[C]//IEEE AINA Workshops. 2015:73 - 76.

[10] WANG Y,WU Q,WONG D S,et al. Securely outsourcing exponentiations with single untrusted program for cloud storage[C]//European Symposium on Research in Computer Security. 2014:326 - 343.

[11] DING Y. Secure outsourcing of modular exponentiations under single untrusted program model[J]. Journal of Computer and System Sciences,2017,90:1 - 13.

[12] ZHOU K,AFIFI H M,REN J. ExpSOS:secure and verifiable outsourcing of exponentiation operations for mobile cloud computing[J]. IEEE Transactions on Information Forensics & Security,2017,12(11):2518 - 2531.

[13] SU Q,YU J,TIAN C,et al. How to securely outsource the inversion modulo a large composite number[J]. Journal of Systems and Software,2017,129:26 - 34.