单云服务器下的安全外包模幂运算

王健一 王 箭
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 模幂运算是加密和签名系统中最基础的运算。由于模幂运算需要耗费很大的计算成本,因此很多方案提出将模幂运算安全外包给云服务器。但是,现存的大多方案都需要两个不共谋的服务器来实现安全的模幂运算,一旦服务器共谋,就会导致外包隐私数据泄露。此外,很多现有方案都假设底数和指数都是保密的,但这并不适合于大多数现实应用场景。通常来说,为了减轻计算负担,只有敏感消息才需要被保密。为了解决上述问题,分别提出了固定底数(底数公开、指数保密)和固定指数(指数公开、底数保密)的安全外包方案。在该方案中客户端只需要使用一个云服务器,从而避免了两个服务器的共谋攻击。理论分析及实验结果证明了该方案的安全性和高效性。

关键词 云计算,安全外包算法,模幂运算,单服务器
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.11.023

Secure Outsourcing Modular Exponentiations with Single Untrusted Cloud Server

WANG Jian-yi WANG Jian
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Modular exponentiation is one of the most fundamental operations in many encryption and signature systems. Due to the heavy computation cost of modular exponentiation, many schemes have been put forward to securely outsource modular exponentiation to cloud. However, most of the existing approaches need two non-colluded cloud servers to implement the secure modular exponentiation, resulting in private data leakage once the cloud servers collude. Besides, most existing schemes assume both base and exponent in modular exponentiation are private, which does not conform to many real-world applications. Usually, in order to reduce the overhead of computation, only the sensitive messages should be privately protected. To solve the above problems, this paper proposed two secure outsourcing schemes based on fixedbase (public base and private exponent) or fixed exponent(private base and public exponent), respectively. In the proposed schemes, the client only needs one cloud server, thus avoiding collusion attack of two servers. Theoretical analysis and experimental results reveal the security and efficiency of the proposed schemes.

Keywords Cloud computing, Outsourcing-secure algorithm, Modular exponentiations, Single server

1 引言

云计算因能经济且高效地为用户提供动态存储和计算服务等而受到广泛的关注。利用云服务器的计算资源,一个资源受限的设备可以完成超出其计算能力的计算任务。

用户外包计算任务给云服务器时,不可避免地会出现很多新型的安全问题[9-10]。云服务器不能被完全信任,用户需要保护外包计算任务中的敏感信息;云服务器可能不返回正确的结果,用户需要具备验证云服务器返回结果真实性的能力[4]。

在大多数的加密和签名系统中,如 RSA 和 Paillier 加密[8],模幂运算是最基础、最昂贵的运算之一,很多计算受限的用户都通过云服务器来高效地完成计算任务。由于云服务器不可信,用户需要借助安全外包算法将计算任务外包给云服务器。

然而,现存的大多数安全外包模幂运算方案[1,3,11]都需要两个云服务器,它们都不能抵抗云服务器的共谋攻击,在效率和可验证率上都存在缺陷[12-13];并且很多运算方案的前提是底数和指数都是保密的[3,5-6],这并不符合现实中的很多具体情况。例如,在使用模幂运算的公钥加密中,底数和指数中有一个是公钥,另一个是明文,但只有明文需要保密。

针对上述问题,本文分别提出了单云服务器下固定底数(底数公开、指数保密)和固定指数(指数公开、底数保密)的安全外包模幂方案,并通过理论分析及实验结果证明了所提方案的安全性和高效性。

本文第 2 节阐述了系统模型和安全性定义;第 3 节阐述了固定底数和固定指数算法的具体执行过程;第 4 节阐述了本文算法的理论分析;第 5 节通过模拟实验证明了算法的高效性。

到稿日期:2017-10-17 返修日期:2018-01-02

王健一(1991-),男,硕士生,主要研究领域为云环境下的隐私保护,E-mail:jianyiwang@nuaa.edu.cn;**王 箭**(1968-),男,教授,主要研究领域为信息安全,E-mail:wangjian@nuaa.edu.cn(通信作者)。

2 系统模型和安全性定义

2.1 系统模型

方案的系统模型包括一个客户端和一个云服务器。客户端输入数据 x,计算任务为 $f(x)$。但是由于缺乏计算资源,客户端需要借助云服务器的帮助来计算 $f(x)$。考虑到输入数据 x 有可能是敏感的,客户端需要对输入数据进行转换来保护其安全。通过函数 f 和转换后的输入数据,云服务器可以完成计算任务。得到云服务器的返回结果后,客户端对返回结果进行转换,得到最后的 $f(x)$;同时,客户端有能力对数据的真实性进行验证。本文参照文献[15-16]对算法的执行过程进行描述,但是与文献[15-16]不同,本文方案不需要公钥,因此不包含密钥生成算法 KenGen(·)。本文方案可划分为以下 3 步。

1)ProbGen$(x,\tau,f)\rightarrow\sigma_x$:根据秘密输入 τ 将输入 x 转化成可公开的值 σ_x,然后将 σ_x 发送给云服务器。

2)Compute$(f,\sigma_x)\rightarrow\sigma_y$:根据 σ_x 和计算函数 f,云服务器按照客户端的要求进行计算,计算结束后将结果 σ_y 发送给客户端。

3)Verify$(\sigma_y,\tau)\rightarrow y\cup\perp$:客户端根据秘密输入 τ 对云服务器的返回结果 σ_y 进行验证,如果验证通过,即 $y=f(x)$,则云服务器计算正确,方案输出 y;否则,方案输出 $\perp$,云服务器返回错误结果。

本文函数 f 代表模幂运算,系统模型如图 1 所示。

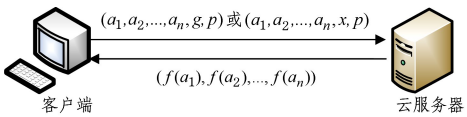

图 1 系统模型

Fig. 1 System model

固定底数(底数公开、指数保密)算法:底数 g 是公开的,指数 x_i 是保密的,客户端计算 $(g^{x_1},g^{x_2},\cdots,g^{x_t})\bmod p$ 的值。客户端通过固定底数算法对指数集合进行分解,然后组成一个数据公开的集合 A。云服务器返回 $g^{a_i}\bmod p$ 的结果给客户端。最后,客户端根据云服务器的返回结果得到 g^{x_i}。

固定指数(底数保密、指数公开)算法:指数 x 是公开的,底数 g_i 保密,客户端计算 $(g_1^x,g_2^x,\cdots,g_t^x)\bmod p$ 的值。客户端通过固定指数算法对指数集合进行分解,然后组成一个数据公开的集合 A。云服务器返回 $a_i^x\bmod p$ 的结果给客户端。最后,客户端通过云服务器的返回结果得到 g_i^x。

2.2 安全性定义

2.2.1 输入隐私和输出隐私

算法的安全性主要包括输入隐私和输出隐私。本文通过一个经典的不可区分性实验对输入隐私和输出隐私进行定义,以保证输入和输出信息不会被泄露。

当一个外包计算方案满足输入隐私时,它对两个不同输入所产生的输出是不可区分的。下面通过不可区分性实验来形式化定义输入隐私[15]。

Experiment $Exp_A^{Ipiv}[f,k]$

询问和响应过程:对于 $i=1,\cdots,l=poly(k)$,敌手根据已有信息不断询问 $A(x_1,\sigma_{x_1},\cdots,x_{i-1},\sigma_{x_{i-1}})\rightarrow x_i$。客户端响应敌手询问 ProbGen$(x_i,\tau_i)\rightarrow\sigma_{x_1}$。

挑战过程:敌手根据已有信息为客户端选择两个输入 $A(x_1,\sigma_{x_1},\cdots,x_l,\sigma_{x_l})\rightarrow(x_{c0},x_{c1})$。客户端通过掷硬币的方法从中随机选择一个值作为算法输入,假设选择 x_{cb} 作为输入。客户端计算 ProbGen$(x_{cb},\tau)\rightarrow\sigma_{x_{cb}}$,将 $\sigma_{x_{cb}}$ 发送给敌手。敌手根据输出对输入进行猜测,$A(x_1,\sigma_{x_1},\cdots,x_l,\sigma_{x_l},x_c,\sigma_{x_c})\rightarrow x'_{cb}$。如果 $x'_{cb}=\sigma_{x_{cb}}$,则输出 1,否则输出 0。

在上述实验过程中,敌手可以对算法进行多次询问,如果敌手根据算法的输出成功区分出输入,则表示该算法是不安全的。

在一个外包方案中,本文通过上面实验的方式定义敌手的优势:$Adv_A^{Ipriv}[f,k]=\text{Prob}[Exp_A^{Ipriv}[f,k]=1]-1/2$。一个外包计算方案满足输入隐私要求,当且仅当任何敌手在多项式时间内的优势可忽略,即 $Adv_A^{Ipriv}[f,k]<neg(\cdot)$,其中 $neg(\cdot)$ 代表可忽略的概率。

输出隐私与输入隐私类似[15],敌手在不断查询和挑战的过程中不能成功区分出不同的输出,在多项式时间内敌手的优势可忽略,即 $Adv_A^{Opriv}[f,k]<neg(\cdot)$。

2.2.2 β-可验证性

外包算法(T,U)满足 β-可验证性[12],算法需要满足以下条件:

1)T^U 是对算法的正确实现;

2)对于任意的输入 x,客户端 T 拥有不低于 β 的概率检测出云服务器 U 返回的错误结果。

2.2.3 α-高效性

外包算法(T,U)满足 α-高效性[14],算法需要满足以下条件:

1)T^U 是对算法的正确实现;

2)对于任意的输入 x,T 的运行时间都不会高于算法运行时间的 α 因子倍。

2.3 Rand 程序

Rand 程序[2-3]是用来加快算法速度和提高可验证率的。Rand 的主要输入是素数 p 和底数 g,输出是一个随机独立的指数对。下面根据算法要求的指数对的不同形式分别介绍 Rand_1 和 Rand_2。Rand_1 被用于生成随机指数对$(a,g^a\bmod p)$和固定底数;Rand_2 被用于生成随机指数对$(g,g^x\bmod p)$和固定指数。目前,EBPV 生成器是生成随机指数最安全高效的方法,它对于 n 比特的指数运行时间为 $O(\log^2 n)$。

3 安全外包模幂运算算法

本节主要介绍两种安全外包模幂运算算法的具体实现,两种算法都只需要一个云服务器。p 是一个大素数,g 是循环群 G_P 的生成元。

3.1 固定底数的安全外包模幂算法

固定底数的外包算法旨在通过将模幂运算安全外包给云服务器来得到 $(g^{x_1},g^{x_2},\cdots,g^{x_t})\bmod p$ 的结果。由于底数 g 是公开的,因此需要保护的隐私数据为$\{x_1,x_2,\cdots,x_t\}$。为了使云服务器在计算过程中得不到任何有用信息,并且客户端通过少量计算就能验证云服务器返回结果的正确性,把指数集合$\{x_1,x_2,\cdots,x_t\}$转换成没有关联的其他集合,并将转换后的集合 $A=\{a_1,a_2,\cdots,a_n\}$发送给云服务器。

云服务器基于信息(A,g,p)进行模幂运算后把计算结果返回给客户端。客户端根据集合 A 与$\{x_1,x_2,\cdots,x_t\}$的内在联系对云服务器返回的结果进行验证。若验证通过,则客户端通过简单计算就可以得到 g^{x_i} 的值。固定底数的安全外包模幂算法的正式描述如算法 1 所示。

算法 1　固定底数的安全外包模幂算法

输入:$\{x_1,x_2,\cdots,x_t\},g,p$

输出:$(g^{x_1},g^{x_2},\cdots,g^{x_t})\bmod p$

1. 客户端将指数集合$\{x_1,x_2,\cdots,x_t\}$转换成集合 A,最后使得 x_i 是集合 A 中 m 个元素的叠加和,集合 A 由集合 B,C,R 组成。
2. 令 $B=\{b_1,b_2,\cdots,b_{m-1}\}$,集合 B 是从$\mathbb{Z}_p$中随机选择的 m−1 个随机数。集合 C 可由公式 $c_i=x_i-\sum_{j=1}^{m-1}b_j,i\in[1,t]$得到。客户端调用 $Rand_1$ 函数生成 e 个指数对$(y_i,g^{y_i}),i\in[1,e]$。集合 R 通过公式 $r_{b_i}=y_i-\sum_{j=1}^{m-1}b_j$ 和 $r_{c_i}=y_i-\sum_{j=1}^{t}c_j,i\in[1,e]$得到。
3. 客户端通过伪随机置换函数将集合$\widetilde{A}=\{b_1,b_2,\cdots,b_{m-1},c_1,c_2,\cdots,c_t,r_{b_1},r_{b_2},\cdots,r_{b_e},r_{c_1},r_{c_2},\cdots,r_{c_e}\}$中的元素顺序打乱并记录 B,C,R 中元素的索引,得到集合 $A=\{a_1,a_2,\cdots,a_n\}$,$n=m+t+2e-1$。为了保证其安全性,n 的值需要大于 200。最后,客户端将 $\sigma_x=(A,g,p)$发送给云服务器。
4. 云服务器进行计算后,把计算结果集 $V=\{g^{a_1},g^{a_2},\cdots,g^{a_n}\}$发送给客户端。
5. 首先,客户端需要对返回结果进行验证。令 $S_{b_i}=\prod g^{b_i}\cdot g^{r_{b_i}}$,$S_{c_i}=\prod g^{c_i}\cdot g^{r_{c_i}},i\in[1,e]$。如果 $S_{b_i}=S_{c_i}=g^{y_i},i\in[1,e]$,则客户端可以认为服务器诚实地进行了运算,客户端可以计算得到 $g^{x_i}=\prod g^{b_i}\cdot g^{c_i},i\in[1,t]$;反之,客户端确定服务器的返回结果有误,服务器没有诚实地运算。

3.2　固定指数的安全外包模幂算法

固定指数算法的具体过程与固定底数算法的过程类似,其描述如算法 2 所示。

算法 2　固定指数的安全外包模幂算法

输入:$(g_1,g_2,\cdots,g_t),x,p$

输出:$(g_1^x,g_2^x,\cdots,g_t^x)\bmod p$

1. 客户端将底数集合$(g_1,g_2,\cdots,g_t)$转换成集合 A,最后使得 x_i 是集合 A 中 m 个元素的叠乘,集合 A 由集合 B,C,R 组成。
2. 令 $B=\{b_1,b_2,\cdots,b_{m-1}\}$,集合 B 是从$\mathbb{G}_p$中随机选择的 m−1 个随机数。集合 C 可由公式 $c_i=g_i/\prod_{j=1}^{m-1}b_j,i\in[1,t]$得到。客户端调用 $Rand_2$ 函数生成 e 个指数对$(g_{y_i},g_{y_i}^x),i\in[1,e]$。集合 R 通过公式 $r_{b_i}=g_{y_i}/\prod_{j=1}^{m-1}b_j$ 和 $r_{c_i}=g_{y_i}/\prod_{j=1}^{t}c_j,i\in[1,e]$得到。
3. 客户端通过伪随机置换函数将集合$\widetilde{A}=\{b_1,b_2,\cdots,b_{m-1},c_1,c_2,\cdots,c_t,r_{b_1},r_{b_2},\cdots,r_{b_e},r_{c_1},r_{c_2},\cdots,r_{c_e}\}$中元素的顺序打乱并记录 B,C,R 中元素的索引,得到集合 $A=\{a_1,a_2,\cdots,a_n\}$,$n=m+t+2e-1$。为了保证其安全性,n 的值需要大于 200。最后,客户端把(A,x,P)发送给云服务器。
4. 云服务器进行计算后,把计算结果集 $V=\{a_1^x,a_2^x,\cdots,a_n^x\}$ 发送给客户端。
5. 首先,客户端需要对返回结果进行验证。令 $S_{b_i}=\prod b_i^x\cdot r_{b_i}^x$,$S_{c_i}=\prod c_i^x\cdot r_{c_i}^x,i\in[1,e]$。如果 $S_{b_i}=S_{c_i}=g_{y_i}^x,i\in[1,e]$,则客户端可以认为服务器诚实地进行了运算,然后客户端可以计算得到 $g_i^x=\prod b_i^x\cdot c_i^x,i\in[1,t]$;反之,客户端确定服务器的返回结果有误,服务器没有诚实地运算。

4　算法分析

1)固定底数和固定指数的安全外包算法都满足正确性。

在固定底数的算法中,如果服务器返回的结果正确,客户端就可以根据服务器的返回值得到$(g^{x_1},g^{x_2},\cdots,g^{x_t})$,具体公式的计算过程如下:$g^{x_i}=g^{c_j}\cdot\prod g^{b_j}=g^{\sum b_j+c_i},i\in[1,t]$。

同理,固定指数算法的具体公式的计算过程如下:$g_i^x=c_i^x\cdot\prod b_j^x=(\prod b_j\cdot c_i)^x,i\in[1,t]$。

2)固定底数和固定指数的安全外包算法都满足安全性。

由于两个算法具有相似性,本节只需要对固定底数算法的外包安全性进行证明。由于固定底数算法已满足正确性,因此本节专注于证明固定底数算法的安全性。

SSP(Subset Sum Problem):给定一个包含 n 个正整数的集合$(a_1,a_2,\cdots,a_n)$和一个正整数 s,在集合内找到一个叠加和为 s 的子集,形式化描述为 $x_1,x_2,\cdots,x_n\in\{0,1\}$,求得一组 x_i 使得 $s=\sum_1^n x_ia_i$ 成立。

SSP 问题是一种特殊的背包问题。并不是所有的 SSP 问题都是难解问题,其难解程度依赖于集合$\{a_1,a_2,\cdots,a_n\}$中元素的个数 n。基于格的方法是当前解决 SSP 问题最有效的解决方法,当 n 不超过 200 时,SSP 问题是可以被解决的;但是当 n 超过 200 时,SSP 问题就是一个难解问题[17-18]。

对于固定指数算法而言,当不可信服务器在算法执行完成后仍然得不到任何指数信息时,则说明固定指数的算法满足外包算法的安全性。下文将对固定底数的算法的输入隐私和输出隐私进行安全性分析。

在输入隐私方面,不可信云服务器拥有集合 $A=[a_1,a_2,\cdots,a_n\}$,理论上算法的输出$(g^{x_1},g^{x_2},\cdots,g^{x_t})$是保密的,但是客户端可能因为加密需要公开某个 g^{x_i} 作为公开参数。假设有一个敌手成功打破本文算法的输入隐私,即敌手发现 $g^{x_i}=g^{\sum a}$。对于敌手而言,这就变成了 SSP 问题,即当 n 超过 200 时,在集合 A 中找到和为 x_i 的子集是一个难解问题,因此算法满足输入隐私。

同样地,在输出隐私方面,不可信云服务器根据客户端要求计算得到的计算结果为 $V=\{g^{a_1},g^{a_2},\cdots,g^{a_n}\}$。假设有一个敌手成功打破本文算法的输出隐私,即敌手发现 $g^{x_i}=\prod g^a=g^{\sum a}$,这同样可以转化为 SSP 问题,敌手仍然不能得到任何有用信息,因此算法满足输出隐私。

3)如果服务器欺骗客户端,客户端的可验证率为 $1-C_t^2/C_n^2-t/(C_n^{e+1}\cdot(e+1))$。

一个安全外包算法的 β-可验证率指客户端可以以不低于 β 的概率检测出服务器返回一个错误值的行为。本文算法的可验证率为 $1-C_t^2/C_n^2-t/(C_n^{e+1}\cdot(e+1))$。由于固定底数和固定指数算法具有相似性,本节只给出固定底数算法的可验证率证明。

固定底数算法的集合 A 由 B,C 和 R 组成,但客户端需要计算$\prod g^{b_i}$,$\prod g^{c_i}$,$g^{r_{b_i}}$,$g^{r_{c_i}}$,进而对以下等式进行验证。

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

服务器成功地欺骗了客户端意味着错误的返回结果通过了客户端的验证。如果服务器在计算结束后只随机修改一个

计算结果，则客户端的可验证率为 1，这种情况下服务器是不可能成功的。服务器成功欺骗客户端至少需要同时修改两个计算结果：此时只存在两种情况：①服务器对集合 C 内的两个结果分别增大 w 倍和减小到 $1/w$，这样服务器的返回结果就可以成功地通过客户端的验证，这种情况发生的概率为 C_t^2/C_n^2；②服务器端成功地找到了集合$\{c_i, r_{c_1}, \cdots, r_{c_e}\}$并从中找到 c_i，将 c_i 和$\{r_{c_1}, \cdots, r_{c_e}\}$中的元素分别增大 w 倍和减小到 $1/w$，概率为$(t/C_n^{e+1}) \cdot (1/(e+1)) = t/(C_n^{e+1} \cdot (e+1))$。

因此，客户端的可验证率为 $1-C_t^2/C_n^2 - t/(C_n^{e+1} \cdot (e+1))$。为了更好地提高效率，$e$ 的值应该远小于 t 和 m，确定了 t 和 e 的值后，可验证率与 n 的大小成正比，但是效率与 n 的大小成反比。考虑到用户对可验证率和效率有不同的需求，用户可以根据自己的实际情况调整 n 的值来平衡算法的可验证率和效率。在本文的模拟实验中，t 与 n 的比值不超过1/2，可验证率不低于 77.54%。

4)对算法的计算、存储和通信复杂度进行分析。

①计算复杂度：将本文算法与文献[1,5,7,12]中的算法进行对比。此处忽略了其他模加之类的操作，具体的效率和可验证性对比结果如表 1 所列。其中，MM(Modular Multiplication)代表模乘；MInv(Modular Inverse)代表模逆；IvkS(Invoke Service)代表与服务器的交互轮数；IvkR(Invoke Rand)代表调用 Rand 的次数；Check 代表可验证率；Server 代表服务器的个数。

表 1 效率和可验证率的对比

Table 1 Comparison of efficiency and checkability

算法性能评估指标	文献[1]^{1}算法	文献[7]算法	文献[1]^{3}算法	文献[5]算法	文献[12]算法	本文算法
MM	$m+t-2$	$7t$	$2m+2t-4$	$13t$	$8t$	$m+t+2e-1$
MInv	0	$2t$	0	6	$2t$	0
IvkR	0	$3t$	0	6	$2t$	e
IvkS	2	$6t$	1	$2t+2$	1	1
Check	3/4	2/3	1	p_1	≈ 1	p_2
Server	Two	Two	One	One	One	One

注：[1]和[1]^3 分别代表 Ma 等[1]提出的第 1 个和第 3 个模幂运算算法

表 1 中，$p_1 = 1 - t^2/10(4t^2+6t+2)$，$p_2 = 1 - C_t^2/C_n^2 - t/(C_n^{e+1} \cdot (e+1))$。与其双服务器算法相比，本文算法在 MM 上略差，由于 e 的取值不会很大，两种算法的效率差距不大，但本文算法基于一个服务器实现，不存在服务器共谋攻击的威胁。另外，本文算法的 MM 几乎只有单服务器算法的一半，因此本文算法更高效。由表 1 的对比分析可知，本文算法的效率和可验证性均优于文献[7]中的算法，并且单服务器也会为用户节省一定成本。虽然文献[5]将算法的可验证率几乎提高到 1，但是其将部分模幂运算任务交给客户端来完成，这样极大地降低了算法效率。与文献[12]相比，本文的 MM 和 Minv 个数均较低，具有更高的效率。虽然文献[12]将可验证率几乎提高到 1，但是给云服务器增加了原计算任务指数倍的计算量，虽然云服务器拥有强大的计算能力，但是如果接收到过于庞大的计算任务，也会增加计算时间，从而降低算法效率。

②存储复杂度：在固定底数算法中，客户端需要计算 S_{setB}，S_{setC} 和 g^{y_i}，因此客户端需要储存数据个数为 n。同样地，固定指数算法需要储存的数据个数也为 n。

③通信复杂度：在固定底数算法中，客户端将(A, g, p)发送给云服务器，云服务器把相应的计算结果返回给客户端，因此客户端和云服务器的通信轮数仅为 1。假设每个数的大小为 k 比特，则固定底数算法的通信复杂度为 $2nk$，同样地，固定指数算法的通信复杂度也为 $2nk$。

5 实验结果与分析

实验是在 Ubuntu 12.04 系统下，基于酷睿 i3-3320 处理器和 4GB 内存的主机，使用 C++和 GNU 高精度算术运算库完成的。

本次实验分别记录文献[1]^1、文献[1]^3、文献[5]和本文算法对不同数目的模幂运算所消耗的时间。通过理论分析可知，文献[12]和文献[7]的效率相对不足，因此没有针对文献[12]和文献[7]的算法进行模拟实验。由于本文两种算法的效率几乎相同，下文实验中进行将采用固定底数算法的效率来代表本文算法的效率。

模拟实验随机生成了 200 个 0～512 bit 的随机数，调用 2 次 Rand 函数，随机选择了一个 100bit 的底数 g 并随机生成了 100 个随机数。

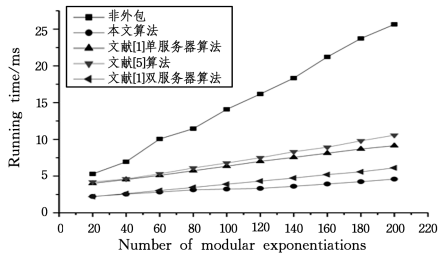

图 2 时间消耗对比

Fig. 2 Comparison of time cost

图 2 中横坐标为模幂运算个数，纵坐标为消耗时间。我们在不同模幂运算个数下对不同方案消耗的时间进行对比。根据图 2 中的实验结果可知，本文算法与其他几种算法相比拥有更高的效率。由于文献[1]^{1}双服务器算法的验证计算量随着模幂运算个数的增加而增长，且需要与云服务器多轮通讯，因此其效率相对较低。文献[1]^{3}单服务器算法的整体过程是执行两次双服务器算法，因此在效率上比文献[1]^{1}算法低，且耗时几乎为文献[1]^{1}算法的两倍。文献[5]算法的耗时相对较长，这是由于其算法需要对底数和指数都进行线性转换，且客户端需要承担部分模幂运算任务，因此效率相对不足。耗时最长的为不外包的方法，其耗时几乎是线性增加的，这也证明了安全外包算法的确能大幅提高模幂运算的效率。

结束语 本文针对固定底数(底数公开、指数保密)和固定指数(指数公开、底数保密)两种情况，分别提出了两种单云服务器下的安全外包模幂运算算法。最后，通过理论分析和实验结果证明了算法的安全性、高效性和高可验证率，且本文算法可以安全高效地实现模幂运算的云外包。

该方法还有一些后续问题值得研究：本文方法采用 Rand 函数对云服务器的返回结果进行验证，若 Rand 程序调用得过多，则会对算法的效率产生影响。因此，少用或不用 Rand 函数，仅通过建立数据内部的等式关系来对返回结果进行验证，同时保证高效率和高可验证率，是后续研究和改进的方向。

参 考 文 献

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

[2] HOHENBERGER S, LYSYANSKAYA A. How to Securely Outsource Cryptographic Computations[C]// International Conference on Theory of Cryptography. 2005: 264-282.

[3] CHEN X, LI J, MA J, et al. New Algorithms for Secure Outsourcing of Modular Exponentiations[C]// European Symposium on Research in Computer Security. Springer Berlin Heidelberg, 2012: 541-556.

[4] GOLLE P, MIRONOV I. Uncheatable Distributed Computations [C]// Topics in Cryptology-CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001. DBLP, 2001: 425-440.

[5] DING Y, XU Z, YE J, et al. Secure outsourcing of modular exponentiations under single untrusedprogrammemodel[J]. Journal of Computer & System Sciences, 2017, 90(1): 1-13.

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

[7] YE J, XU Z, DING Y. Secure outsourcing of modular exponentiations in cloud and cluster computing[J]. Cluster Computing, 2016, 19(2): 811-820.

[8] PAILLIER P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes[J]. Lecture Notes in Computer Science, 1999, 547(1): 223-238.

[9] ATALLAH M J, FRIKKEN K B. Securely outsourcing linear algebra computations[C]// ACM Symposium on Information, Computer and Communications Security. ACM, 2010: 48-59.

[10] BENJAMIN D, ATALLAH M J. Private and Cheating-Free Outsourcing of Algebraic Computations[C]// Sixth Conference on Privacy, Security and Trust. IEEE Computer Society, 2008: 240-245.

[11] REN Y, DING N, ZHANG X, et al. Verifiable Outsourcing Algorithms for Modular Exponentiations with Improved Checkability[C]// ACM on Asia Conference on Computer and Communications Security. 2016: 293-303.

[12] ZHAO L, ZHANG M, SHEN H, et al. Privacy-preserving Outsourcing Schemes of Modular Exponentiations Using Single Untrusted Cloud Server[J]. Ksii Transactions on Internet & Information Systems, 2017, 11(2): 826-845.

[13] REN K, WANG C, WANG Q. Security Challenges for the Public Cloud[J]. IEEE Internet Computing, 2012, 16(1): 69-73.

[14] WANG C, CAO N, REN K, et al. Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data[J]. IEEE Transactions on Parallel & Distributed Systems, 2012, 23(8): 1467-1479.

[15] CHUNG K M, KALAI Y, VADHAN S. Improved delegation of computation using fully homomorphic encryption[M]// Advances in Cryptology — CRYPTO 2010. Berlin: Springer-Verlag, 2010: 483-501.

[16] GENNARO R, GENTRY C, PARNO B. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers [M]// Advances in Cryptology - CRYPTO 2010. Berlin: Springer-Verlag, 2010: 465-482.

[17] BOYKO V, PEINADO M, VENKATESAN R. Speeding up Discrete Log and Factoring Based chemes via Precomputations[M]// Advances in Cryptology — EUROCRYPT'98. Berlin: Springer-Verlag, 1998: 221-235. S

[18] COSTER M J, JOUX A, LAMACCHIA B A, et al. Improved low-density subset sum algorithms[J]. Computational Complexity, 1992, 2(2): 111-128.

[19] HOROWITZ E, SAHNI S. Computing Partitions with Applications to the Knapsack Problem[M]. New York: Cornell University, 1972.

（上接第 142 页）

[18] GREENSMITH J, AICKELIN U, CAYZER S. Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection[M]// Artificial Immune Systems. Springer Berlin Heidelberg, 2005: 153-167.

[19] WANG Y Q, LIANG Y W, LIU S. Application-layer DDoS attack detection based on dendritic cell algorithm[J]. Computer Engineering and Design, 2015, 36(4): 841-845. (in Chinese)
王亚芹，梁意文，刘赛. 基于树突状细胞算法的应用层 DDoS 攻击检测[J]. 计算机工程与设计，2015，36(4)：841-845.

[20] GREENSMITH J, AICKELIN U. The Dendritic Cell Algorithm [J]. RevistaClínica Española, 2007, 202(10): 552-554.

[21] YI L L. Research on Statistical Characteristic Analysis and Modeling for User Behavior in Micro-blog Community Based on Human Dynamics[D]. Beijing: Beijing University of Posts and Telecommunications, 2012. (in Chinese)
易兰丽. 基于人类动力学的微博用户行为统计特征分析与建模研究[D]. 北京：北京邮电大学，2012.

[22] HE L, HE Y, HUO Y Q. Micro-blog user characteristics analysis and core user mining [J]. Intelligence Theory and Practice, 2011, 34(11): 121-125. (in Chinese)
何黎，何跃，霍叶青. 微博用户特征分析和核心用户挖掘[J]. 情报理论与实践，2011，34(11)：121-125.

[23] WANG X G. Empirical Analysison Behavior Characteristics and Relation Characteristics of Micro-blog Users — Take "SinaMicro-blog" for Example[J]. Library and Information Service, 2010, 54(14): 66-70. (in Chinese)
王晓光. 微博客用户行为特征与关系特征实证分析——以"新浪微博"为例[J]. 图书情报工作，2010，54(14)：66-70.

[24] Sina Weibo. Sunshine credit[EB/OL]. http://service.account.weibo.com/sunshine/guize.

[25] LIAN J, ZHOU X, CAO W, et al. SINA microblog data retrieval [J]. Journal of Tsinghua University, 2011, 51(10): 1300-1305. (in Chinese)
廉捷，周欣，曹伟，等. 新浪微博数据挖掘方案[J]. 清华大学学报(自然科学版)，2011，51(10)：1300-1305.

[26] YANG M, YIN J M, JI G L. Classification Methods on Imbalanced Data: a Survey[J]. Journal of Nanjing Normal University, 2008, 8(4): 7-12. (in Chinese)
杨明，尹军梅，吉根林. 不平衡数据分类方法综述[J]. 南京师范大学学报(工程技术版)，2008，8(4)：7-12.