

基于三重扰动和线性组合的批量矩阵乘法可验证计算方案

张天彭, 任志宇, 杜学绘, 王海超

(信息工程大学, 河南 郑州 450001)

摘要: 随着云计算、物联网技术的发展, 可验证计算作为一种新兴的计算技术得到了广泛的应用。可验证计算在为用户带来便利的同时也存在着安全挑战: 数据隐私性、结果可验证性和高效性。目前, 矩阵乘法可验证计算方案基本采用稀疏矩阵相乘的加密方法来保护数据的隐私性。分析稀疏矩阵加密算法后, 发现存在两个挑战: 一是行或列公因子泄露原始矩阵行或列数据, 二是零元素泄露原始矩阵零元素统计信息。同时, 现有方案对云服务器计算结果的验证效率也不理想。针对数据隐私保护挑战, 设计的批量矩阵乘法可验证计算方案利用三重扰动加密算法在不增加解密计算复杂度的同时达到更强的隐私保护效果, 其中, 通过构造特殊的上或下三角稀疏矩阵加入双重扰动(乘法扰动和加法扰动)来保护行或列数据, 通过构造特殊的加法稀疏矩阵加入单重扰动(加法扰动)来保护零元素信息。针对云服务器计算结果验证效率挑战, 所提方案利用矩阵线性组合技术实现计算结果的批量验证, 验证效率提高了约50倍, 并随着矩阵数量的增多而提升。性能分析表明, 所提方案基本没有增加客户端解密开销, 并提高了结果验证效率。

关键词: 三重扰动; 线性组合; 隐私增强; 效率提升; 可验证计算

中图分类号: TP309.7

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2024035

Verifiable computation scheme of batch matrix multiplication based on triple perturbation and linear combination

ZHANG Tianpeng, REN Zhiyu, DU Xuehui, WANG Haichao

Information Engineering University, Zhengzhou 450001, China

Abstract: With the development of cloud computing and internet of things technology, verifiable computing has been widely used as a new computing technology. While verifiable computing brings convenience to users, there are also security challenges: data privacy, verifiability of results, and efficiency. At present, the sparse matrix multiplication encryption method is used to protect the privacy of data in the matrix multiplication verifiable calculation scheme. After analyzing the sparse matrix encryption algorithm, it is found that there are two challenges. The one is that the

收稿日期: 2023-02-08; 修回日期: 2023-07-24

通信作者: 任志宇, ren_ktzy@163.com

基金项目: 中原科技创新领军人才项目 (No.224200510003)

Foundation Items: Central Plains Science and Technology Innovation Leading Talents Project (No.224200510003)

引用格式: 张天彭, 任志宇, 杜学绘等. 基于三重扰动和线性组合的批量矩阵乘法可验证计算方案[J]. 网络与信息安全学报, 2024, 10(02): 121-132.

Citation Format: ZHANG T P, REN Z Y, DU X H, et al. Verifiable computation scheme of batch matrix multiplication based on triple perturbation and linear combination[J]. Chinese Journal of Network and Information Security, 2024, 10(02): 121-132.

row or column common factor leaks the row or column data of the original matrix. The other is that the zero element leaks the statistic information of the zero element of the original matrix. Meanwhile, the existing scheme is also not ideal for the verification efficiency of cloud server computing results. Aiming at the challenge of data privacy protection, the designed batch matrix multiplication verifiable computation scheme uses triple perturbation encryption algorithm to achieve stronger privacy protection without increasing the computational complexity of encryption and decryption. Specifically, a special upper or lower triangular sparse matrix is constructed to add double perturbation (multiplicative perturbation and additive perturbation) to protect row or column data, and a special additive sparse matrix is constructed to add a single perturbation (additive perturbation) to protect zero element information. Aiming at the challenge of verification efficiency of cloud server computing results, the scheme uses matrix linear combination technology to realize batch verification of calculation results. The verification efficiency is increased by about 50 times and increases with the increase of the number of matrices. Performance analysis shows that this scheme does not increase the client encryption and decryption overhead and improves the efficiency of result verification.

Keywords: triple perturbation, linear combination, enhanced privacy, improved efficiency, verifiable computation

0 引言

信息社会的快速发展使得云计算成为当前最热门的技术之一,可验证计算作为云计算的一部分,已经成为一个重要的研究课题^[1-4]。可验证计算使得计算能力有限的客户端能够以安全和可验证的方式将繁重的工作任务外包给计算能力强大的云。矩阵计算在科学研究中的应用非常广泛^[5-6],大量应用于人工智能、数字图像处理、电路分析、社会科学、物理现象研究等领域。矩阵乘法计算作为矩阵运算的基本形式之一,扮演着非常重要的角色。矩阵乘法运算通常伴随着很高的计算复杂度,资源受限的客户端往往无法及时处理大量数据,需要将计算任务外包给强大的云服务器。但外包过程中存在一些挑战^[7-8],如数据隐私性、结果可验证性、方案高效性等。因此,设计正确、安全、可验证、高效的矩阵乘法可验证计算方案是必需的。

近年来,有很多关于隐私保护的矩阵乘法可验证计算的研究。常用的解决方案有同态加密和矩阵变换。同态加密对密文数据执行任意计算等同于对明文数据执行相同的计算。云服务器可以直接对用户加密的密文进行相应的计算,计算过程中无法获得用户的隐私数据,返回的结果等同于直接对明文进行同样的计算。使用同态加密保护矩阵乘法数据隐私,更多的是研究矩阵编码方法。学者先后提出基于单指令多数据技术^[9]的行序、列序、对角线序^[10]矩阵编码方法,二进制和

非二进制^[11]矩阵编码方法,超立方结构^[12]矩阵编码方法和单密文中加密多矩阵^[13]矩阵编码方法。可验证计算的效率得到提升,但由于同态加密方案大多只支持部分同态运算(加法同态或乘法同态),而且存在噪声随运算次数增加而增长、阶段计算量较大、密文大小膨胀等问题,会需要额外的特殊硬件,使得方案复杂、效率低,难以实际应用。矩阵变换利用特殊矩阵与原始矩阵进行相应运算达到盲化数据保护隐私的效果。Salinas^[14]、Fu^[15]等利用随机矩阵产生加法扰动来保护数据隐私,但解密时客户端的计算量与原始计算相当,效率不高。Lei^[16]、Kumar^[17]、张兴兰^[18]等基于稀疏矩阵或特殊矩阵(正交矩阵和对角矩阵)产生乘法扰动进行矩阵加密,提高了矩阵加解密效率,但会泄露原始矩阵中零元素的统计信息(具体分析见文献[19-22])。Zhang等^[19]通过有限域内的幺模矩阵和稀疏矩阵设计了可以隐藏零元素信息的方案,但是需要使用大量的私密矩阵,增加了客户端的计算开销和存储消耗。Kumar^[20-21]、Wu^[22]等利用稀疏矩阵相乘和随机矩阵相加的方法进行加密,在乘法扰动的同时加入了加法扰动,既保护了零元素信息又保持了加解密效率,但发现存在行或列公因子泄露原始矩阵行或列数据的问题。综合比较,矩阵变换既能保护数据隐私,又能保持较高的效率,整体效益较好。

本文在综合文献[20-22]的优点和不足的基础上,通过构造三重扰动加密算法对矩阵进行加密,

增强了对数据隐私的保护。同时,利用矩阵线性组合技术实现了矩阵乘法计算结果的批量验证,提高了计算结果的验证效率,具体贡献如下。

1) 通过构造特殊的上或下三角稀疏矩阵在原始矩阵中加入双重扰动(乘法扰动和加法扰动),在不影响矩阵加解密效率的基础上,达到了对原始矩阵更强的隐私保护,解决了行或列公因子泄露原始矩阵行或列数据的问题。

2) 利用矩阵分块技术,通过构造特殊的加法稀疏矩阵在原始矩阵中加入单重扰动(加法扰动)来保护零元素信息。通过提高加法矩阵的稀疏度,降低了解密阶段的计算复杂度,解密的效率更高。

3) 利用矩阵线性组合技术设计了矩阵乘法计算结果批量验证方案,验证效率提高了约50倍,并随着矩阵数量的增多而提升。

4) 对所提方案进行理论分析和实验验证。实验结果验证了算法的有效性,表明所提算法达到了全部设计目标。

1 可验证计算描述

1.1 安全可验证计算

安全可验证计算方案由5个算法组成^[23]: $SVC = (\text{keyGen}, \text{ProbGen}, \text{Compute}, \text{Verify}, \text{Decryption})$ 。安全可验证计算方案模型如图1所示。

1) $\text{KeyGen}(\lambda) \rightarrow (\text{EK}, \text{DK})$: 密钥生成算法。输入安全参数 λ ,输出密钥对 (EK, DK) ,加密密钥 EK 用来加密数据,解密密钥 DK 用来解密数据,算法 KeyGen 由客户端执行。

2) $\text{ProbGen}(f, x, \text{EK}) \rightarrow (F, \sigma_x, \tau_x)$: 输入处理

算法。输入为计算函数 f ,函数输入 x 和加密密钥 EK ,输出为编码后的计算函数 F ,函数输入 σ_x 和验证证据 τ_x ,算法 ProbGen 由客户端执行。

3) $\text{Compute}(F, \sigma_x) \rightarrow \sigma_y$: 计算执行算法。输入编码后的计算函数 F 和输入 σ_x ,输出对应的函数计算结果 $y = f(x)$ 的编码值 σ_y ,算法 Compute 由服务器执行。

4) $\text{Verify}(\tau_x, \sigma_y) \rightarrow T \cup F$: 结果验证算法。根据云服务器返回的验证证据 τ_x 和编码值 σ_y ,判断计算结果是正确还是错误的,算法 Verify 由客户端执行。

5) $\text{Decryption}(\sigma_y, \text{DK}) \rightarrow y$: 结果解密算法。输入验证通过的结果的编码值 σ_y 和解密密钥 DK ,输出函数真实计算结果 y ,算法 Decryption 由客户端执行。

1.2 威胁模型

可验证计算方案模型的安全威胁源于云服务器的3种威胁模型,分别是可信模型、半可信模型和不可信模型^[23-24],本文主要针对不可信模型展开研究。

可信模型:云服务器正确遵循方案算法,未经授权不会访问客户端信息。

半可信模型:云服务器表现为“诚实但好奇”。服务器遵循算法指令并生成正确的结果,但试图通过数据分析获得客户端的隐私数据。

不可信模型:云服务器会表现出恶意行为,是最强的对抗模型。服务器可能是“懒惰的”“好奇的”和“不诚实的”,可能会偏离算法指令,可能伪造错误结果并尝试通过客户端验证,

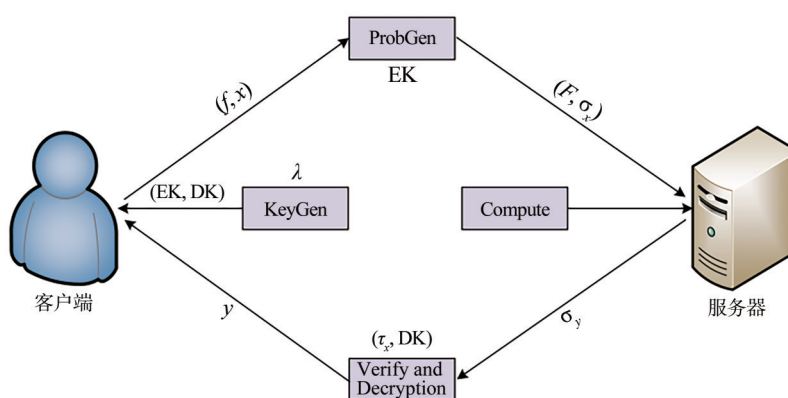


图1 安全可验证计算方案模型
Figure 1 Secure verifiable computation scheme model

以节省其资源。

1.3 设计目标

不可信云服务器模型下设计的可验证计算方案应满足以下 4 个要求^[25]。

定义 1 正确性：如果客户端和云服务器正确执行方案，那么客户端最后得到的结果一定是正确的，即不可信工作者不能让客户端接受错误的结果。

定义 2 隐私性：包括客户端的输入隐私（云服务器不能从接收到的数据中获取客户端的隐私信息）和输出隐私（计算结果对云服务器也是私有的）。对于任意不可信的云服务器，从客户端发送的加密数据或者计算结果中获得客户端隐私信息的概率是可以忽略的。

定义 3 可验证性：对于任意的云服务器，其返回的错误结果通过验证算法的概率是可忽略的。这意味着客户端能够以接近于 1 的概率检测出错误的结果。

定义 4 高效性：客户端执行方案所需的时间必须远小于完成计算任务本身的时间。

2 三重扰动矩阵加密算法

2.1 矩阵基本知识

定义 5 置换函数：对于任意给定的有限集合 $S = \{1, 2, \dots, n\}$ ，定义在 S 上的置换函数 $\pi: S \rightarrow S$ 表示为

$$\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \pi_1 & \pi_2 & \pi_3 & \cdots & \pi_n \end{pmatrix} \quad (1)$$

其中， $\pi_1, \pi_2, \dots, \pi_n$ 是 $1, 2, \dots, n$ 的一个全排列。

$$\begin{pmatrix} 0 & \alpha_1 & 0 & 0 \\ 0 & 0 & \alpha_2 & 0 \\ \alpha_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_4 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} 0 & 0 & \beta_1 & 0 \\ 0 & \beta_2 & 0 & 0 \\ 0 & 0 & 0 & \beta_3 \\ \beta_4 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} \alpha_1 \beta_4 a_{24} & \alpha_1 \beta_2 a_{22} & \alpha_1 \beta_1 a_{21} & \alpha_1 \beta_3 a_{23} \\ \alpha_2 \beta_4 a_{34} & \alpha_2 \beta_2 a_{32} & \alpha_2 \beta_1 a_{31} & \alpha_2 \beta_3 a_{33} \\ \alpha_3 \beta_4 a_{44} & \alpha_3 \beta_2 a_{42} & \alpha_3 \beta_1 a_{41} & \alpha_3 \beta_3 a_{43} \\ \alpha_4 \beta_4 a_{14} & \alpha_4 \beta_2 a_{12} & \alpha_4 \beta_1 a_{11} & \alpha_4 \beta_3 a_{13} \end{pmatrix} \quad (5)$$

分析式(5)中加密矩阵的行或列可知，加密矩阵的每行元素都有一个公因子 α_i 、每列元素都有一个公因子 β_j ，因此不可信云服务器可以首先分析加密矩阵的行元素找到其公因子 α_i ，然后分析加密矩阵的列元素找到其公因子 β_j ，最后恢复出原始矩阵的真实数据 a_{ij} 。

针对这个问题，本文在添加乘法扰动的同时加入加法扰动，即使用特殊构造的上或下三角稀

定义 6 克罗内克函数：一般使用符号 $\delta_{x,y}$ 表示，具体定义为在该函数中，对于任意的 x 和 y ，如果 x 和 y 的值相等，则函数值为 1；如果 x 和 y 的值不相等，则函数值为 0，如式(2)所示：

$$\delta_{x,y} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \quad (2)$$

定义 7 置换矩阵：对于一个可逆的方阵，如果该方阵的每行和每列都只有一个元素为 1，其余的元素都是 0，那么这个矩阵叫作置换矩阵。可以用置换函数 π 和克罗内克函数 $\delta_{x,y}$ 生成随机置换矩阵 P ，矩阵 P 的第 i 行第 j 列元素的表达式为

$$P(i, j) = \delta_{\pi(i), j} \quad (3)$$

显然，只有当 $\pi(i) = j$ 时，矩阵在该位置上的元素值才是 1，其余位置都为 0。

定义 8 稀疏矩阵：当矩阵中数值为 0 的元素个数远大于非 0 的元素个数，并且零元素没有分布规律时，称该矩阵为稀疏矩阵。利用随机置换矩阵 P 和随机正整数集合 α 可以生成可逆的随机稀疏矩阵，生成公式为

$$P_{\pi}(i, j) = \alpha_i \delta_{\pi(i), j} \quad (4)$$

上述矩阵基本知识主要用于可验证计算方案中的矩阵加解密阶段，起到保护矩阵数据隐私的目的。

2.2 加密改进

矩阵加密基本上是使用矩阵乘法引入乘法扰动，即在原始矩阵左右两侧各乘上一个如上所述的随机稀疏矩阵^[21-22]，以 4×4 矩阵为例，具体过程如下。

稀疏矩阵在原始矩阵中加入双重扰动。该上或下三角稀疏矩阵除对角元素非零外，在每一行的对角元素后方或每一列对角元素下方随机选择一个位置赋予非零值，并且要与该行或该列的对角元素不同（上或下三角稀疏矩阵生成算法如算法 1 所示）。如果两个元素相等，依然存在如式(5)所示的公因子问题。改进后，将原始矩阵与特殊构造的上或下三角稀疏矩阵相乘进行加密时，在原有

乘法扰动的基础上加入加法扰动,即双重扰动,从而增强对原始矩阵隐私数据的保护。

算法1 上或下三角稀疏矩阵生成算法

输入: 正整数集合 $\alpha: \{\alpha_1, \alpha_2, \dots, \alpha_m, \dots, \alpha_{2m-1}\}$

输出: 上三角矩阵 U 或下三角矩阵 L

```

1) for  $i = 1$  up to  $m$  do
2)    $U(i, i) = \alpha_i$ 
3)   当  $i < m$  时, 从  $(i+1, m)$  中随机选择一个值  $j$ ,  $U(i, j) = \alpha_{m+i}$ 
4) end for
5) for  $j = 1$  up to  $m$  do
6)    $L(j, j) = \alpha_j$ 
7)   当  $j < m$  时, 从  $(j+1, m)$  中随机选择一个值  $i$ , 令  $L(i, j) = \alpha_{m+j}$ 
8) end for

```

在解决原始矩阵零元素泄露问题上,当前多数算法采用的是通过矩阵加法引入单重加法扰动。加法矩阵由均匀密度分布的随机矩阵^[21]或同一个随机元素构成的随机矩阵^[22]构成。但是引入稠密矩阵需要多项式的计算时间和内存需求。经过分析发现,如果加入一些限制条件,则可以使得原始矩阵在加法扰动过程中依然保持其原有的计算复杂度,达到既保护零元素又不增加矩阵计算难度的效果。通过引入矩阵分块技术,客户端在更小的部分执行矩阵加法,针对性保护原始矩阵零元素。对于矩阵 M , 如果 $M_{ij} = 0$, 那么随机选择一个非零元与其相加,即 $M'_{ij} = M_{ij} + z$ 。矩阵 Z 除位置 (i, j) 处元素不为0,其余元素都是

零。这样执行 $M' = M + Z$ 后,既保护了矩阵 M 的零元素,又不影响其计算复杂度。加法矩阵生成算法如算法2所示。

算法2 加法矩阵生成算法

输入: 随机数 z

输出: 加法矩阵 Z

```

1) for  $i = 1$  up to  $m$  do
2)   for  $j = 1$  up to  $n$  do
3)     当  $M(i, j) = 0$  时, 令  $Z(i, j) = z$ 
4)   end for
5) end for

```

3 批量矩阵乘法可验证计算方案

3.1 系统模型

假设有一个服务提供商,为管理的所有同类型终端设备提供服务。终端设备由于资源受限,无法执行一些大规模运算,服务提供商需要对每个终端提供的隐私数据执行矩阵乘法计算并将结果返回给终端。由于服务提供商资源无法满足批量终端的计算量和时效性,服务提供商希望将计算任务外包给云服务器。批量矩阵乘法可验证计算方案模型如图2所示。

本文方案模型中的实体有3个:客户端、服务提供商和云服务器。

客户端: 主要指资源受限的终端,如智能穿戴设备、智能手机等,向服务提供商提交终端隐私数据,希望得到服务提供商的服务。为了防止自身的私密信息被窃取,客户端必须对原始矩阵的敏感

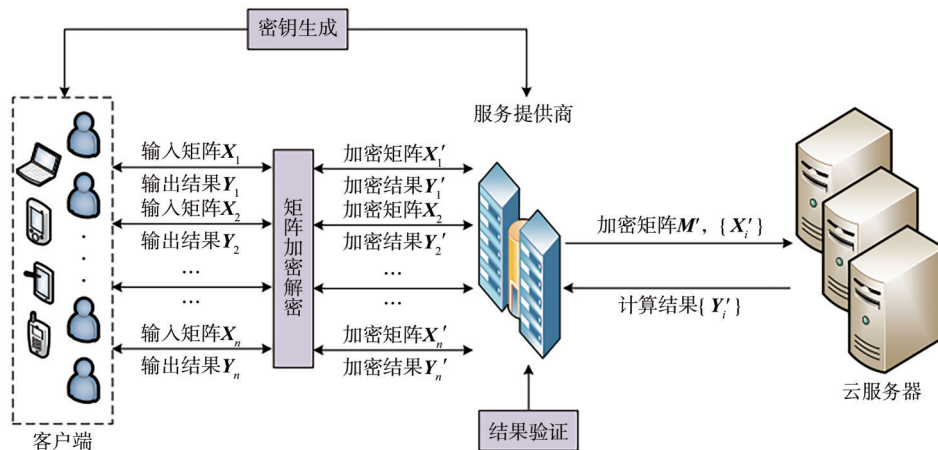


图2 批量矩阵乘法可验证计算方案模型

Figure 2 Batch matrix multiplication verifiable computation scheme model

信息进行隐藏。在从服务提供商处得到结果后，客户端需进行恢复计算，从而得到真正的结果。

服务提供商：由于自身的计算资源有限，服务提供商不能及时处理服务请求，向终端用户提供服务，所以将计算任务外包给云服务器。服务提供商需要对云服务器返回的结果进行验证，以保证计算结果的正确性。为防止自己的服务被窃取，服务提供商需将服务提供矩阵进行加密。本文方案模型中的服务提供商属于半可信模型，试图通过数据分析获得客户端的隐私数据。

云服务器：云服务器的计算资源非常丰富，在收到服务提供商请求的计算任务后，首先利用自身资源完成计算任务，然后将计算结果返回给服务提供商。本文方案模型中的云服务器属于不可信模型，试图获得客户端和服务提供商的隐私数据，并为节省资源试图返回错误计算结果。

3.2 计算步骤

为了不失矩阵乘法的一般性，令服务提供商要解决的任务是 k 个客户端请求的矩阵乘法 $\{Y_i\} = M \{X_i\}$ ，其中 $M \in \mathbb{R}^{m \times l}$ ， $X_i \in \mathbb{R}^{l \times n}$ ， $i \in [1, k]$ （矩阵 M 为服务提供商向客户提供某种服务的矩阵函数，客户端向服务矩阵 M 输入数据 X_i ，从而获取对应的服务功能）。服务提供商将这个计算任务外包给云服务器来完成。由于客户端和服务提供商不希望云服务器知道其原始数据，因此每个客户端将输入矩阵 X_i 转换成加密矩阵 X'_i ，服务提供商将矩阵函数 M 转换成 M' 。然后，服务提供商将转换后的计算任务 $\{Y'_i\} = M' \{X'_i\}$ 发送给云服务器。云服务器计算结果 $\{Y'_i\} = M' \{X'_i\}$ 并返回给服务提供商。服务提供商验证结果的正确性时，可同时批量验证 k 个矩阵乘法的结果，验证通过后各客户端可分别解密得到真实结果。为体现公平性，避免因通信等造成优先发送请求计算任务的客户端长时等待，约定服务提供商采用时间轮询的方式请求服务器，即每隔时间 t 将已收集的请求计算任务外包给服务器。接下来对所提可验证计算方案的步骤进行详细描述。

Step1 密钥生成：根据三重扰动加密算法，需要构造特殊的上或下三角稀疏矩阵和加法稀疏矩阵。因此服务提供商和客户端 i 分别选择安全参数 λ 和 λ_i ， λ 和 λ_i 是正整数。根据原始矩阵 M 和

X_i 的大小，分别生成随机集合 $\alpha: \{\alpha_1, \alpha_2, \dots, \alpha_{2m-1}\}$ ($\alpha_i \in [1, 2\lambda m]$) 和 $\beta_i: \{\beta_1, \beta_2, \dots, \beta_{2n-1}\}$ ($\beta_{ij} \in [1, 2\lambda_i n]$)，作为后续上或下三角稀疏矩阵中的非零元素；分别生成随机数 z_i ($z_i \in [1, \lambda]$) 和 $z_{i,2}$ ($z_{i,2} \in [1, \lambda_i l]$)，作为后续加法矩阵中的非零元素；分别生成随机置换 $\pi \leftarrow \text{RP}(1, \dots, m)$ 和 $\pi_i \leftarrow \text{RP}(1, \dots, n)$ ，起到打乱原始矩阵行列排列的作用。用 $\text{SPEK} = \{\alpha, z_i, \pi\}$ 代表服务提供商生成的加密密钥。用 $\text{CEK}_i = \{\beta_i, z_{i,2}, \pi_i\}$ 代表客户端 i 生成的加密密钥。

Step2 矩阵加密：根据算法 1 和算法 2，服务提供商和客户端根据加密密钥首先生成三重扰动加密算法中所需的上或下三角稀疏矩阵和加法稀疏矩阵。服务提供商和客户端 i 首先引入单重加法扰动，将原始矩阵 M 和 X_i 与加法稀疏矩阵 Z_1 和 $Z_{i,2}$ 分别相加得到 M' 和 X'_i ，来保护零元素的隐私信息；然后引入双重扰动，将 M' 和 X'_i 与上或下三角稀疏矩阵分别相乘得到 M'' 和 X''_i ，来保护行或列隐私信息。由于经过三重扰动加密处理，所以云服务器不能从接收到的数据中获取到服务提供商和客户端的敏感信息。矩阵加密算法如算法 3 所示。

算法 3 矩阵加密算法

输入： $M, \{X_i\}, \text{SPEK}, \{\text{CEK}_i\}$

输出： $M'', \{X''_i\}$

1) 服务提供商使用密钥 SPEK ，采用算法 1 生成上三角矩阵 U ，并计算 U^{-1} ，其中 $U \in \mathbb{R}^{m \times m}$ ；采用算法 2 生成加法矩阵 Z_1 ，其中 $Z_1 \in \mathbb{R}^{m \times l}$ ；利用随机置换 π 生成随机置换矩阵 P ，并计算 P^{-1} 。

2) 客户端 i 使用密钥 CEK_i 采用算法 1 生成下三角矩阵 L_i ，并计算 L_i^{-1} ，其中 $L_i \in \mathbb{R}^{n \times n}$ ；采用算法 2 生成加法矩阵 $Z_{i,2}$ ，其中 $Z_{i,2} \in \mathbb{R}^{l \times n}$ ；利用随机置换 π_i 生成随机转换矩阵 P_i ，并计算 P_i^{-1} 。

3) 加入单重加法扰动： $M' = M + Z_1$ ， $X'_i = X_i + Z_{i,2}$ 。

4) 加入双重扰动： $M_L = PU$ ， $M'' = M_L M'$ ， $M_{Ri} = L_i P_i$ ， $X''_i = X'_i M_{Ri}$ 。

5) 服务提供商计算 $M_L^{-1} = U^{-1} P^{-1}$ ，客户端计算 $M_{Ri}^{-1} = P_i^{-1} L_i^{-1}$ 。

6) 服务提供商将 (M_L^{-1}, Z_1) 发给每个客户端，客户端 i 将解密密钥 $\text{CDK}_i = (M_L^{-1}, Z_1, M_{Ri}^{-1}, Z_{i,2})$

保存到本地。

Step3 矩阵计算：云服务器计算 $\{Y_i''\} = M'' \{X_i''\}$ 并将结果 $\{Y_i''\}$ 发回给服务提供商。

Step4 结果验证：为提高结果验证效率，服务提供商首先利用矩阵线性组合技术对客户端的加密矩阵集合 $\{X_i''\}$ 和结果矩阵集合 $\{Y_i''\}$ 进行预处理，然后进行验证。服务提供商每运行一次方案，都批量计算和验证多个矩阵乘法，通过平摊实现了效率提升。服务提供商采用蒙特卡罗算法^[6]对结果进行验证。根据蒙特卡罗算法，客户端可以判断出结果是正确的或错误的。结果验证算法如算法4所示。

算法4 结果验证算法

输入： $\{Y_i''\}, M'', \{X_i''\}$

输出： 结果正确或者结果错误

- 1) $Y = Y_1'' + Y_2'' + \dots + Y_k''$
- 2) $X = X_1'' + X_2'' + \dots + X_k''$
- 3) for $i = 1$ up to s do
- 4) 生成一个随机的 $n \times 1$ 的二进制向量 r ，然后计算 $Z = M''(Xr) - Yr$
- 5) if $Z \neq (0, \dots, 0)^T$
- 6) 返回“结果错误”，并立即终止
- 7) end if
- 8) end for
- 9) 返回“结果正确”，结束验证。

Step5 结果解密：客户端利用存储在本地的解密密钥 CDK_i 对结果进行恢复计算。结果解密算法如算法5所示。其中，由于服务提供商和客户端在构造矩阵 Z_1 、 $Z_{i,2}$ 时只有个别位置是非零元，所以解密时仅需要进行矩阵非零元对应位置处的标量乘法，降低了计算复杂度。

算法5 结果解密算法

输入： $\{Y_i''\}, M'', \{X_i''\}, M_L^{-1}, \{M_{Ri}^{-1}\}, Z_1, \{Z_{i,2}\}$

输出： $\{Y_i\}$

- 1) 计算矩阵 $H_i = Z_1 X_i$ ；初始化矩阵 H_i 为 0 矩阵
- 2) for $r = 1$ up to m do
- 3) if 矩阵 Z_1 当前行有非 0 元，记下列标 $(c_1, c_2, \dots, c_a) (a \leq l)$
- 4) for $c = 1$ up to n do
- 5) $H_i(r, c) = z_1 * (X_i(c_1, c) + \dots$

$+ X_i(c_a, c))$

6) end for

7) end if

8) end for

9) 同理计算矩阵 $Q_i = MZ_{i,2}$

10) 计算结果 $Y_i = M_L^{-1} Y_i'' M_{Ri}^{-1} - H_i - Q_i - Z_1 Z_{i,2}$

4 安全性分析

本节将对所提方案的正确性、隐私性、可验证性进行分析。

4.1 正确性

定理1 如果各方都严格按照方案执行，并且不出现软件故障和硬件问题，则客户端最后得到的结果一定是正确的。

证明 根据方案的矩阵加密阶段可以得到加密后的矩阵分别为

$$M'' = PU(M + Z_1) = M_L(M + Z_1) \quad (6)$$

$$X_i'' = (X_i + Z_{i,2})L_i P_i = (X_i + Z_{i,2})M_{Ri} \quad (7)$$

服务提供商接收到云服务器返回的结果是

$$\begin{aligned} Y'' &= M'' X'' = \\ M_L(M + Z_1)(X_i + Z_{i,2})M_{Ri} &= \\ M_L(MX_i + Z_1 X_i + MZ_{i,2} + Z_1 Z_{i,2})M_{Ri} \end{aligned} \quad (8)$$

客户端进行结果恢复计算

$$H_i = Z_1 X_i \quad (9)$$

$$Q_i = MZ_{i,2} \quad (10)$$

$$\begin{aligned} Y_i &= M_L^{-1} Y_i'' M_{Ri}^{-1} - H_i - Q_i - Z_1 Z_{i,2} = \\ M_L^{-1} M_L(M + Z_1)(X_i + Z_{i,2})M_{Ri} M_{Ri}^{-1} - \\ H_i - Q_i - Z_1 Z_{i,2} &= \\ (M + Z_1)(X_i + Z_{i,2}) - H_i - Q_i - Z_1 Z_{i,2} &= \\ MX_i \end{aligned} \quad (11)$$

显然结果是正确的。

证毕。

4.2 隐私性

云服务器可能试图通过数据分析获得客户端的隐私数据，包括输入数据和输出结果。本文方案使用三重扰动加密方法达到保护数据隐私的目标，即通过构造特殊的上或下三角稀疏矩阵在原始矩阵中加入双重扰动（乘法扰动和加法扰动），通过构造特殊的加法稀疏矩阵在原始矩阵中加入单重扰动（加法扰动）。分析可知，云服务器获取输入矩阵和输出矩阵的可能性是一个可以忽略的

概率,以此来保护客户端的输入隐私和输出隐私。

定理 2 方案能够保护客户端数据的输入隐私。

证明 客户端、服务提供商在将矩阵发送给云服务器之前分别对矩阵 $M^{m \times l}$ 和 $X_i^{l \times n}$ 进行加密计算,得到 $M'' = PU(M + Z_1)$ 和 $X_i'' = (X_i + Z_{i,2})L_iP_i$ 。由算法 1 得知,矩阵 U 和 L 加入双重扰动,可以避免通过公因子分析得知原始矩阵行或列数据。由算法 2 得知,加法矩阵 Z_1 和 $Z_{i,2}$ 中的非零元素刚好隐藏了原始矩阵的零元素。

如果云服务器想要从矩阵 M'' 中恢复出矩阵 M ,则云服务器需要知道随机置换矩阵 P 、上三角矩阵 U 和矩阵 Z_1 。由矩阵基本知识得随机置换矩阵 P 有 $m!$ 种可能。由算法 1 得知,上三角矩阵 U 中的元素从 $[1, 2\lambda m]$ 中随机选取。另外,矩阵除对角元素外,每行在随机位置上还存在一个随机数,所以矩阵 U 共有 $(m-1)!(2\lambda m)^{2m-1}$ 种可能。由算法 2 得知,矩阵 Z_1 由元素 z_1 组成, z_1 从 $[1, \lambda l]$ 中随机选取。假设矩阵 M 中有 a 个零元,并随机分布,则共有 C_{ml}^a (排列组合) 种可能。所有矩阵 Z_1 共有 $\lambda l C_{ml}^a$ 可能。因此,云服务器从

$$\text{Prob}_Y = \frac{1}{m!(m-1)!n!(n-1)!(2\lambda m)^{2m-1}(2\lambda_i n)^{2n-1}\lambda_i \lambda l^2 C_{ml}^a C_{ln}^a} \quad (14)$$

在本文方案中,矩阵的维数总是很高,所以 m 、 l 和 n 都很大。如果安全参数 λ 、 λ_i 足够大,云服务器获取矩阵 Y_i 的可能性就是一个可以忽略的概率,所以本文方案可以保护客户端的输出隐私。

证毕。

4.3 可验证性

云服务器可能会偏离算法指令,可能会伪造错误结果并尝试通过客户端验证,以节省其资源。本文方案采用蒙特卡罗算法^[16]对结果进行验证,当云服务器返回的错误结果通过客户端验证程序的概率可以忽略时,客户端得到的结果是正确的。

定理 4 云服务器返回的错误结果通过客户端验证程序的概率是可以忽略的。

证明 在方案的结果验证阶段,服务提供商采用蒙特卡罗算法对结果进行验证。服务提供商根据验证参数 s 的值进行重复验证。在重复

矩阵 M'' 中恢复出矩阵 M 的概率为

$$\text{Prob}_M = \frac{1}{m!(m-1)!(2\lambda m)^{2m-1}\lambda l C_{ml}^a} \quad (12)$$

同理,云服务器从矩阵 X_i'' 中恢复出矩阵 X_i 的概率为

$$\text{Prob}_{X_i} = \frac{1}{n!(n-1)!(2\lambda_i n)^{2n-1}\lambda_i l C_{ln}^a} \quad (13)$$

在本文方案中,矩阵的维数总是很高,所以 m 、 l 和 n 都很大。如果安全参数 λ 、 λ_i 足够大,云服务器获取矩阵 M 和 X_i 的可能性就是一个可以忽略的概率,所以本文方案可以保护客户端的输入隐私。

证毕。

定理 3 方案能够保护客户端数据的输出隐私。

证明 云服务器在客户端的请求下进行矩阵乘法运算,最终得到的结果是 $Y_i'' = PU(M + Z_1)(X_i + Z_{i,2})L_iP_i$ 。服务器想要从矩阵 Y_i'' 中恢复出 Y_i ,则需要知道随机置换矩阵 P 和 P_i 、上三角矩阵 U 、下三角矩阵 L_i 、加法矩阵 Z_1 和 $Z_{i,2}$ 。由定理 2 的证明步骤得知,云服务器获得矩阵 Y_i 的概率为

$$\frac{1}{m!(m-1)!n!(n-1)!(2\lambda m)^{2m-1}(2\lambda_i n)^{2n-1}\lambda_i \lambda l^2 C_{ml}^a C_{ln}^a} \quad (14)$$

验证的过程中,如果服务提供商能够识别不正确的结果,则说明验证成功;如果服务提供商没有识别出错误结果,则验证失败。定义错误结果通过服务提供商验证算法的概率为 Prob_v ,定义一轮验证算法中错误结果通过的概率为 Prob_{v1} 。

在结果验证阶段,服务提供商计算 $Z = M''(Xr) - Yr$,其中 r 是一个随机的 $n \times 1$ 的二进制向量。令

$$Z = (M''X - Y)r = Qr = (z_1, z_2, \dots, z_m)^T \quad (15)$$

假设现在云服务器返回的结果不是外包任务的计算结果,则 Q 中至少有一个元素不为 0,即 $Q(i,j) \neq 0$ 。根据矩阵与向量的乘法,可以得到

$$z_i = \sum_{k=1}^n Q_{i,k} r_k = \sum_{k=1}^n Q_{i,k} r_k - Q_{i,j} r_j + Q_{i,j} r_j = y + Q_{i,j} r_j \quad (16)$$

根据全概率公式可得

$$P(z_i = 0) = P(z_i = 0|y = 0)P(y = 0) + P(z_i = 0|y \neq 0)P(y \neq 0) \quad (17)$$

由矩阵与向量乘法公式得

$$\begin{cases} P(z_i = 0|y = 0) = P(r_j = 0) = 1/2 \\ P(z_i = 0|y \neq 0) \leq P(r_j = 1) = 1/2 \end{cases} \quad (18)$$

所以

$$P(z_i = 0) \leq (P(y = 0) + P(y \neq 0)) \cdot 1/2 \quad (19)$$

$$P(z_i = 0) \leq (P(y = 0) + (1 - P(y = 0))) \cdot 1/2 \quad (20)$$

$$P(z_i = 0) \leq 1/2 \quad (21)$$

所以，在一轮验证中，错误结果通过验证的概率为

$$\text{Prob}_{v1} = (0, \dots, 0)^T \leq P(z_i = 0) \leq 1/2 \quad (22)$$

由于服务提供商重复 s 次验证，所以

$$\text{Prob}_v \leq 1/2^s \quad (23)$$

当验证次数 s 足够大时， Prob_v 就是一个可以忽略的概率。当云服务器返回的错误结果通过客户端验证程序的概率可忽略时，客户端得到的结果是正确的。

证毕。

5 性能分析

5.1 计算复杂度分析

下面对所提方案的计算复杂度进行理论分析。

1) 矩阵加密：生成密钥后，服务提供商和客户端分别计算 $M'' = PU(M + Z_1)$ 和 $X_i'' = (X_i + Z_{i,2})L_iP_i$ ，计算复杂度为 $O(2ml + 2ln)$ 。

2) 结果验证：在结果验证阶段，服务提供商通过执行算法4进行验证。服务提供商批量验证了 k 个计算结果，计算复杂度为 $O(smn)$ 。

3) 结果解密：客户端 i 通过算法5得到真实结果 Y_i 。计算 $H_i = Z_1X_i$ 和 $Q_i = MZ_{i,2}$ 时，假设矩阵 Z_1 中有 a 行存在非零元 ($a \leq m$)，矩阵 $Z_{i,2}$ 中有

b 列存在非零元 ($b \leq n$)，由算法5得知， k 个客户端的总计算复杂度为 $O(4kmn + kan + kbm)$ 。当 $a = m$ ， $b = n$ 时，计算复杂度最大，为 $O(6kmn)$ 。

客户端和服务提供商计算 k 个矩阵乘法的总计算复杂度为 $O((4k + s)mn + 2ml + 2ln + kan + kbm)$ ，最大值为 $O((6k + s)mn + 2ml + 2ln)$ 。但是，客户端直接计算 k 个矩阵乘法的计算复杂度是 $O(kmln)$ 。本文方案可以降低客户端的计算成本。

表1给出了现有方案的综合比较，其中， $\mathbf{R}$ 表示实数域。 t_e 表示客户端加密的成本， t_{dv} 表示客户端在解密和验证阶段的成本，un 表示论文没有给出具体的指令。

5.2 实验分析

本文使用 Matlab R2022a 来完成实验，代码按照前文的算法设计完成，矩阵由程序随机生成。实验环境是 Intel Core i7-11800H，CPU 的频率是 2.30 GHz，内存是 32.0 GB 的计算机。

由文献[22]得知蒙特卡罗算法的参数值越大，方案的性能增益越小。本文实验设置蒙特卡罗算法的参数 $s = 20$ 。云服务器计算资源无限，在实验过程中不考虑云服务器计算所需时间，只对方案中客户端需要执行的计算进行时间上的测试。本文方案是在文献[22]的基础上改进而来的，所以对比两个方案中客户端执行方案所需的时间。为了保证实验的正确性，防止异常数据干扰实验结果，以下实验结果都是重复 20 次后取的平均值。

在相同的实验环境和参数下，测试本文方案与文献[22]方案的客户端对不同维度的原始矩阵进行加解密操作的总时间，结果如表2所示，其中，矩阵维度 $m : l : n = 3 : 4 : 5$ ，两个方案与矩阵加解密总时间比值如图3所示。

实验结果表明，当矩阵维度较低时（矩阵维

表1 现有方案的综合比较
Table 1 Comprehensive comparison of existing options

方案	矩阵类型	t_e	t_{dv}	保护零元素	抵抗公因子
Kumar 等 ^[20] 方案	$X \in \mathbf{R}^{m \times l}$ $Y \in \mathbf{R}^{l \times n}$	$O(3(ml + ln))$	un	否	否
Wu 等 ^[22] 方案	$X \in \mathbf{R}^{m \times l}$ $Y \in \mathbf{R}^{l \times n}$	$O(2(ml + ln))$	$O((2k + ks)mn + kln + kml)$	是	否
本文方案	$X \in \mathbf{R}^{m \times l}$ $Y \in \mathbf{R}^{l \times n}$	$O(2(ml + ln))$	$O((4k + s)mn + kan + kbm)$ Max: $O((6k + s)mn)$	是	是

度 $l \leq 4\,000$), 文献[22]方案与本文方案矩阵加解密总时间的比值约等于 1; 矩阵维度 $l > 4\,000$ 时, 比值在 0.9 ~ 0.95 内, 本文方案比文献[22]方案的矩阵加解密的时间成本大。但本文方案相对文献[22]方案解决了行或列公因子泄露原始矩阵行或列数据的问题, 达到了对原始矩阵更强的隐私保护效果。

表2 矩阵加解密总时间

Table 2 Total time of matrix encryption and decryption

矩阵维度 l	本文方案加解密总时间 T_{ed}/s	文献[22]方案加解密总时间 T_{ed-22}/s	T_{ed-22}/T_{ed}
800	0.016 8	0.016 9	1.005 9
1 600	0.065 6	0.062 8	0.957 3
2 400	0.132 4	0.131 1	0.990 1
3 200	0.236 1	0.232 9	0.986 4
4 000	0.353 3	0.340 3	0.963 2
6 400	1.029 6	0.965 9	0.938 1
8 000	1.639 1	1.492 7	0.910 7

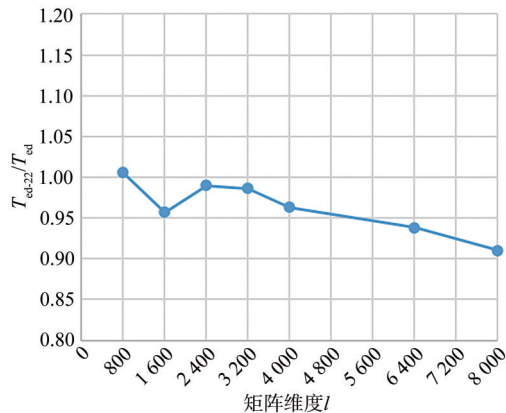


图3 方案矩阵加解密总时间比值

Figure 3 Scheme matrix encryption and decryption total time ratio

设置矩阵维度 $m:l:n=3:4:5$, 分别在 $l=4\,000/6\,400/8\,000$ 的条件下进行 3 组实验, 测试本文方案与文献[22]方案随着客户端数量的增多进行结果验证操作的时间, 结果验证时间如表 3 ($l=4\,000$)、表 4 ($l=6\,400$)、表 5 ($l=8\,000$) 所示, 两方案结果验证时间比值如图 4~图 6 所示。

实验结果表明, 在相同的矩阵维度下, 本文方案的结果验证效率有了显著提高。当客户端数量为 100 时, 结果验证效率提高了约 50 倍, 并随着客户端数量的增多, 结果验证的效率持续提升。实验结果表明, 利用矩阵线性组合技术设计

的矩阵乘法计算结果批量验证方案确实达到了提高结果验证效率的目标。

表3 结果验证时间 ($l=4\,000$)Table 3 Result verification time ($l=4\,000$)

客户端数量/个	本文方案结果验证时间 T_v/s	文献[22]方案结果验证时间 T_{v-22}/s	T_{v-22}/T_v
100	0.448 5	24.092 2	53.717 2
200	0.665 4	48.184 4	72.414 1
300	0.894 4	72.276 6	80.810 1
400	1.177 2	96.368 8	81.862 7
500	1.409 5	120.461 0	85.463 6
800	2.139 0	192.737 6	90.106 4
1 000	2.461 3	240.922 0	97.884 0

表4 结果验证时间 ($l=6\,400$)Table 4 Result verification time ($l=6\,400$)

客户端数量/个	本文方案结果验证时间 T_v/s	文献[22]方案结果验证时间 T_{v-22}/s	T_{v-22}/T_v
100	1.194 7	60.838 1	50.921 1
200	1.906 7	121.676 2	63.815 0
300	2.509 4	182.514 3	72.731 5
400	3.138 4	243.352 4	77.539 4
500	3.743 8	304.190 5	81.251 2
800	5.663 5	486.704 8	85.936 0
1 000	6.859 9	608.381 0	88.686 1

表5 结果验证时间 ($l=8\,000$)Table 5 Result verification time ($l=8\,000$)

客户端数量	本文方案结果验证时间 T_v/s	文献[22]方案结果验证时间 T_{v-22}/s	T_{v-22}/T_v
100	1.911 7	99.093 9	51.833 1
200	3.125 1	198.187 8	63.416 3
300	4.240 9	297.281 7	70.098 4
400	5.305 0	396.375 6	74.716 5
500	6.442 6	495.469 5	76.904 1
800	9.671 0	792.751 2	81.971 9
1 000	11.576 4	990.939 0	85.599 3

6 结束语

本文构造了一个可以实现批量矩阵乘法安全可验证计算的方案, 通过三重扰动加密算法, 使用特殊构造的上或下三角稀疏矩阵和加法稀疏矩阵进行加密, 保护了原始矩阵中零元素信息, 解决了通过公因子获取原始矩阵信息的问题; 利用

矩阵线性组合技术实现了矩阵乘法的批量计算和验证。通过理论分析和实验结果,本文方案满足了安全可验证计算的正确性、隐私性、可验证性。虽然本文方案实现了矩阵乘法的批量计算和验证,减少了客户端成本,但方案只适用于同一个矩阵函数的矩阵乘法,不适用于多对矩阵批量乘法。未来,将更加关注方案能够同时进行多对矩阵批量乘法可验证计算。

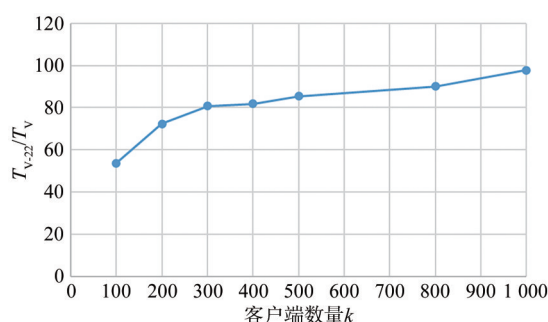


图4 方案结果验证时间比值($l = 4000$)

Figure 4 Scheme results verify the time ratio ($l = 4000$)

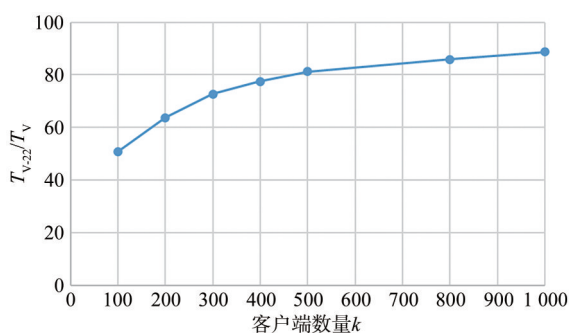


图5 方案结果验证时间比值($l = 6400$)

Figure 5 Scheme results verify the time ratio ($l = 6400$)

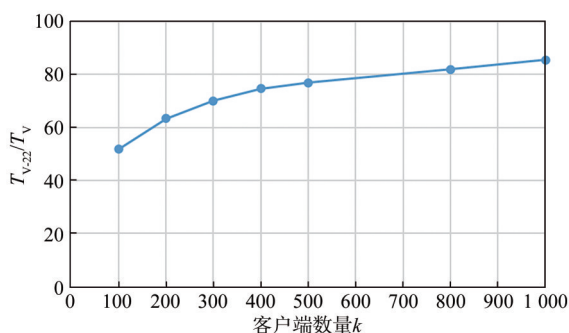


图6 方案结果验证时间比值($l = 8000$)

Figure 6 Scheme results verify the time ratio ($l = 8000$)

参考文献:

[1] CHEN X. Introduction to secure outsourcing computation[J]. Synthesis Lectures on Information Security, Privacy, & Trust,

2016, 8(2): 1-93.

- [2] LI P, LI J, HUANG Z, et al. Privacy-preserving outsourced classification in cloud computing[J]. Cluster Computing, 2018, 21(1): 277-286.
- [3] SHAO J, WEI G. Secure outsourced computation in connected vehicular cloud computing[J]. IEEE Network, 2018, 32(3): 36-41.
- [4] CHEN H, LIAO Y. Improvement of an outsourced attribute-based encryption scheme[J]. Soft Computing, 2019, 23(22): 11409-11417.
- [5] ALI M, KHAN M, TUNG N T. Segmentation of dental X-ray images in medical imaging using neutrosophic orthogonal matrices[J]. Expert Systems with Applications, 2018, 91: 434-441.
- [6] TUCKER L H, CONDE-GONZÁLEZ A, COBICE D, et al. MALDI matrix application utilizing a modified 3D printer for accessible high resolution mass spectrometry imaging[J]. Analytical Chemistry, 2018, 90(15): 8742-8749.
- [7] SUN P J. Security and privacy protection in cloud computing: Discussions and challenges[J]. Journal of Network and Computer Applications, 2020, 160: 102642.
- [8] WEI L, ZHU H, CAO Z, et al. Security and privacy for storage and computation in cloud computing[J]. Information Sciences, 2014, 258: 371-386.
- [9] SMART N P, VERCAUTEREN F. Fully homomorphic SIMD operations[J]. Designs, Codes and Cryptography, 2014, 71(1): 57-81.
- [10] HALEVI S, SHOU P V. Algorithms in helib[C]//Annual Cryptology Conference. 2014: 554-571.
- [11] DUONG D H, MISHRA P K, YASUDA M. Efficient secure matrix multiplication over LWE-based homomorphic encryption[J]. Tatra Mountains Mathematical Publications, 2016, 67(1): 69-83.
- [12] RATHEE D, MISHRA P K, YASUDA M. Faster PCA and linear regression through hypercubes in HELIB[C]//Proceedings of the 2018 Workshop on Privacy in the Electronic Society. 2018: 42-53.
- [13] JIANG X, KIM M, LAUTER K, et al. Secure outsourced matrix computation and application to neural networks[C]//Proceedings of the 2018 ACM SIGSAC Conference On Computer and Communications Security. 2018: 1209-1222.
- [14] SALINAS S, LUO C, CHEN X, et al. Efficient secure outsourcing of large-scale linear systems of equations[C]//2015 IEEE Conference on Computer Communications (INFOCOM). 2015: 1035-1043.
- [15] FU S, YU Y, XU M. A secure algorithm for outsourcing matrix multiplication computation in the cloud[C]//Proceedings of the Fifth ACM International Workshop On Security in Cloud Computing. 2017: 27-33.
- [16] LEI X, LIAO X, HUANG T, et al. Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud[J]. Information Sciences, 2014, 280: 205-217.

- [17] KUMAR M, MEENA J, VARDHAN M. Privacy preserving, verifiable and efficient outsourcing algorithm for matrix multiplication to a malicious cloud server[J]. Cogent Engineering, 2017, 4(1).
- [18] 张兴兰,刘祥. 安全高效的可验证大型线性方程组求解外包计算方案[J]. 网络与信息安全学报, 2017, 3(6): 1-7.
ZHANG X L, LIU X. Secure efficient and verifiable large linear equations solve outsourcing computing scheme[J]. Chinese Journal of Network and Information Security, 2017, 3(6): 1-7.
- [19] ZHANG S, TIAN C, ZHANG H, et al. Practical and secure outsourcing algorithms of matrix operations based on a novel matrix encryption method[J]. IEEE Access, 2019, 7: 53823-53838.
- [20] KUMAR M, VARDHAN M. Secure and verifiable outsourcing algorithm for large-scale matrix multiplication on public cloud server[M]//Engineering Vibration, Communication and Information Processing. Berlin: Springer, 2019: 575-586.
- [21] KUMAR M, MISHRA V, SHUKLA A, et al. A novel publicly delegable secure outsourcing algorithm for large-scale matrix multiplication[J]. Journal of Intelligent & Fuzzy Systems, 2020, 38(5): 6445-6455.
- [22] WU Y, LIAO Y, LIANG Y, et al. Secure and efficient protocol for outsourcing large-scale matrix multiplication to the cloud [J]. IEEE Access, 2020, 8: 227556-227565.
- [23] CHEN X, LI J, MA J, et al. New algorithms for secure outsourcing of modular exponentiations[J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 25(9): 2386-2396.
- [24] LEI X, LIAO X, HUANG T, et al. Outsourcing large matrix inversion computation to a public cloud[J]. IEEE Transactions on Cloud Computing, 2013, 1(1): 1-1.
- [25] GENNARO R, GENTRY C, PARNO B. Non-interactive verifiable computing: outsourcing computation to untrusted workers [C]//Annual Cryptology Conference. 2010: 465-482.

[作者简介]



张天彭 (1994-), 男, 河南许昌人, 信息工程大学硕士生, 主要研究方向为可验证计算、网络与信息安全。



任志宇 (1974-), 女, 河南汤阴人, 博士, 信息工程大学副教授, 主要研究方向为网络与信息安全。



杜学绘 (1968-), 女, 河南新乡人, 博士, 信息工程大学教授、博士生导师, 主要研究方向为网络与信息安全、空间信息网络、云计算安全。



王海超 (1997-), 男, 江苏无锡人, 信息工程大学硕士生, 主要研究方向为区块链安全、网络与信息安全。