Computer Science and Application 计算机科学与应用, 2019, 9(8), 1500-1506
Published Online August 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.98168

Hans汉斯

Secure and Efficient Computing Matrix Inversion in Cloud

Weiwei Jing[1], Youwen Zhu[2]

[1]Nanjing Research Institute of Electronics Technology, Nanjing Jiangsu
[2]Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu
Email: wwjingx@163.com

Received: Jul. 25th, 2019; accepted: Aug. 5th, 2019; published: Aug. 12th, 2019

Abstract

This paper studies secure matrix inversion outsourcing in cloud. We point out the security weakness of the existing scheme, and then propose a new solution which perturbs the private matrix by multiplying two random dense matrices and thus can improve the security. Besides, we present a new method to generate random dense matrices and achieve fast matrix multiplication, by which we can dramatically reduce users' computation cost. We theoretically prove the security of our proposed scheme, and confirm the high efficiency of our proposed scheme by simulation experiments.

Keywords

Cloud Computing, Secure Outsourcing, Privacy Preservation, Matrix Inversion

基于云计算的安全高效矩阵求逆方案

荆巍巍[1]，朱友文[2]

[1]南京电子技术研究所，江苏 南京
[2]南京航空航天大学，江苏 南京
Email: wwjingx@163.com

收稿日期：2019年7月25日；录用日期：2019年8月5日；发布日期：2019年8月12日

摘 要

本文研究了基于云计算的矩阵求逆安全外包问题。针对现有方案安全性较弱等问题，我们提出了一种新的矩阵安全变换机制，可以使用随机的稠密矩阵对用户输入矩阵进行安全乘法变换，能够有效提升用户输入输出的安全性。同时，我们设计了一种新的稠密随机矩阵生成方法，可以支持快速的矩阵乘法，从

文章引用: 荆巍巍, 朱友文. 基于云计算的安全高效矩阵求逆方案[J]. 计算机科学与应用, 2019, 9(8): 1500-1506.
DOI: 10.12677/csa.2019.98168

而能够大幅降低用户的计算复杂度。我们通过理论分析证明了所提方案的安全性。最后，我们通过模拟实验验证了所提方案的运行效率。

关键词

云计算，安全外包，隐私保护，矩阵求逆

1. 引言

云计算[1] [2]可以为人们或中小企业以即用即付的方式提供便捷的计算能力和存储资源，具有减少设备维护开销、快速部署、支持动态资源调整等优势。越来越多的个人和企业将数据存储和计算任务外包到云服务器上完成，然而外包过程会导致用户私有数据泄漏给云服务器，其中可能包含个人健康状况、收入、企业用户数据等隐私信息和私密数据。外部入侵或者内部不诚实员工偷盗数据等因素都可能导致云服务器上的数据发生严重的泄漏。此外，云服务器可能因为软硬件错误或者主动偷懒，导致返回的计算结果不正确。针对这些问题，云计算中的安全外包计算成为了近年来的重要研究话题之一，其目的在于利用云服务器的计算能力完成复杂的运算，同时保护运算所涉及的敏感输入和输出的隐私性，并有效验证云服务器返回结果的正确性。

矩阵求逆是一种常见的计算任务，在许多场景中有着广泛应用[3] [4] [5]，比如方程组求解、回归计算等。如果矩阵的维度为 $n \times n$，矩阵求逆的时间复杂度高达 $O(n^{2.373})$ [6] [7]。当 n 很大时，矩阵求逆耗时很多，配置较弱的客户端往往难以快速完成这一技术，甚至无法承受其计算过程。通过按需租用云服务器，配置较弱的客户端能够以较低的代价完成矩阵求逆。然而，待求逆的矩阵往往是用户的隐私数据，并不能直接向云服务器暴露。为了应对这一矛盾，论文[8]提出了矩阵求逆安全外包问题，并设计了一种高效的实现机制，可以将用户端的计算量降低到 $O(n^2)$。然而，论文[8]的方案仅使用乘以随机数的方式对矩阵中的每个元素进行变换，这种方法会导致两方面的问题；1) 如果矩阵中某个元素等于 0，这种变换方法会直接泄漏该元素的值，不能提供任何安全保护；2) 这种变换方法会泄漏多个元素乘积的比值，在攻击者获得部分元素值的情况下，会导致其他元素的泄漏。

针对上述问题，本文设计了一种新的矩阵求逆安全外包方案。新方案利用稠密矩阵对原始矩阵进行变换，使得变换过程既包括随机乘法又包括随机的加法，消除上述安全问题。重要的是，虽然使用稠密矩阵对原始矩阵进行变换，我们所提方案的依然可以将用户端的时间复杂度降到 $O(n^2)$，能够有效提升计算效率。此外，我们还改进了结果验证机制，使用双验证策略，降低了错误结果被接受的概率。

论文其余部分安排如下。第 2 部分介绍了系统模型和设计目标；第 3 部分展示了我们的新方案；第 4 部分对新方案进行了分析和评估；第 5 部分总结了本文。

2. 系统型和设计目标

2.1. 系统模型

本文的系统模型结构如图 1 所示，其中包含两个参与者，分别是云服务器和用户。云服务器具有较强的计算能力，可以根据用户的需求完成计算过程。用户拥有私有的 $n \times n$ 维矩阵 $\boldsymbol{X}$，他希望获得矩阵 $\boldsymbol{X}$

的逆矩阵 $\boldsymbol{Y}$。然而，用户只能承受 $O(n^2)$的计算复杂度，无法承受普通矩阵求逆所需要的时间复杂度。同时，用户的矩阵 $\boldsymbol{X}$ 包含了隐私数据，用户也无法将该矩阵直接和云服务器共享。为此，用户生成密钥 K，利用密钥 K 将矩阵 $\boldsymbol{X}$ 变换为矩阵 $\boldsymbol{X}'$，然后将矩阵 $\boldsymbol{X}'$发送云服务器，使得云服务器无法从 $\boldsymbol{X}'$推导出 $\boldsymbol{X}$ 的有用信息。接着，云服务器计算矩阵 $\boldsymbol{X}'$的逆矩阵 $\boldsymbol{Y}'$ (即图 1 中 $f(\cdot)$表示矩阵求逆)，并将返回给用户。最后，用户验证 $\boldsymbol{Y}'$的正确性，并根据 $\boldsymbol{Y}'$计算出矩阵 $\boldsymbol{X}$ 的逆 $\boldsymbol{Y}$。

相应地，基于云计算的矩阵求逆安全外包机制共包含五个子算法，其功能简介如下。

- **Gen**(λ)→K：用户根据安全参数 λ 生成密钥 K，用于后续的加密和解密过程。
- **Enc**(K, $\boldsymbol{X}$)→$\boldsymbol{X}'$：用户利用密钥 K 将私有输入 $\boldsymbol{X}$ 加密成 $\boldsymbol{X}'$。
- **Compute**(X')→$\boldsymbol{Y}'$：云服务器计算矩阵 $\boldsymbol{X}'$的逆矩阵 $\boldsymbol{Y}'$。
- **Verify**($\boldsymbol{Y}'$)→0/1：用户验证云服务器返回值 $\boldsymbol{Y}'$的正确与否，1 表示正确，0 表示不正确。
- **Dec**(K, $\boldsymbol{Y}'$)→$\boldsymbol{Y}$：用户在验证 $\boldsymbol{Y}'$正确的情况下，利用 K 解密 $\boldsymbol{Y}'$得到矩阵 $\boldsymbol{X}$ 的逆矩阵 **Y**。

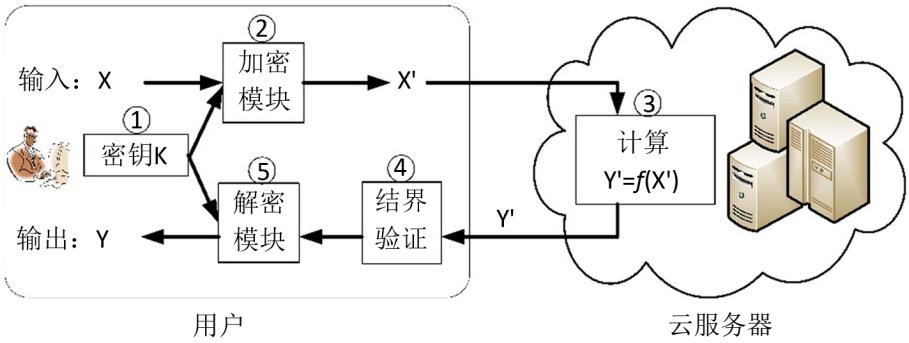

Figure 1. The system model of secure outsourcing computation based on cloud
图 1. 基于云计算的安全外包计算系统模型结构

2.2. 设计目标

下面，我们从安全性和效率等方面分别描述我们的设计目标。

1) 输入隐私性：安全外包方案需要保护用户的输入数据的隐私性，即矩阵 $\boldsymbol{X}$ 是用户的隐私数据，在方案执行前后都不能泄漏给云服务器或任何第三方。

2) 输出隐私性：用户的输出结果 $\boldsymbol{Y}$ 也是隐私的，其隐私性需要有效保护，云服务器不能获得 $\boldsymbol{Y}$ 的有用信息。

3) 效率：用户端总的计算开销应该远小于直接矩阵求逆所需要的开销，即安全外包可以降低用户的的计算开销。

4) 正确性：用户能够以较高的概率验证云服务器返回的结果正确与否。如果云服务器返回的 $\boldsymbol{Y}'$是错误的，用户能够以很高的概率检测出不正确的输出；如果云服务器返回的的 $\boldsymbol{Y}'$是正确的，$\boldsymbol{Y}'$能够通过用户的验证。

3. 方案设计

3.1. 主要思路

本文中，我们设计了一种利用稠密的随机矩阵对用户的输入矩阵 $\boldsymbol{X}$ 进行变换的方法。下面，我们先介绍方案的整体思路。

为了保护矩阵 $\boldsymbol{X}$，我们将使用随机的稠密矩阵 $\boldsymbol{D}_L$ 和 $\boldsymbol{D}_R$ 对矩阵 $\boldsymbol{X}$ 进行变换，它们分别对 $\boldsymbol{X}$ 进行左乘和右乘，即变换后的矩阵 $\boldsymbol{X}' = \boldsymbol{D}_L\boldsymbol{X}\boldsymbol{D}_R$。当矩阵的维度都是 $n \times n$ 时，计算两个矩阵相乘的时间复杂度是 $O(n^3)$，因此简单使用这种方法虽然可以保护 $\boldsymbol{X}$ 的隐私性，但是无法降低用户端的计算开销。这里，我们

使用如下方式生成随机的稠密矩阵 $\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$。令 $\boldsymbol{D_L}=\boldsymbol{L}+\boldsymbol{PQ}$，$\boldsymbol{D_R}=\boldsymbol{R}+\boldsymbol{UV}$，这里 $\boldsymbol{L}$ 和 $\boldsymbol{R}$ 是随机的对角矩阵，$\boldsymbol{P}$ 和 $\boldsymbol{U}$ 是 n*1 的随机矩阵，$\boldsymbol{Q}$ 和 $\boldsymbol{V}$ 是 $1\times n$ 的随机矩阵。假设 $\boldsymbol{L}$ 和 $\boldsymbol{R}$ 的对角线元素依次分别是 $(L_1,L_2,\cdots,L_n)$ 和 $(R_1,R_2,\cdots,R_n)$，$\boldsymbol{P}=(P_1,P_2,\cdots,P_n)^{\mathrm{T}}$，$\boldsymbol{U}=(U_1,U_2,\cdots,U_n)^{\mathrm{T}}$，$\boldsymbol{Q}=(Q_1,Q_2,\cdots,Q_n)$，$\boldsymbol{V}=(V_1,V_2,\cdots,V_n)$。那么，$\boldsymbol{D_L X}=(\boldsymbol{L}+\boldsymbol{PQ})\boldsymbol{X}=\boldsymbol{LX}+\boldsymbol{P}(\boldsymbol{QX})$。不难看出，$\boldsymbol{LX}$ 和 $\boldsymbol{P}(\boldsymbol{QX})$可以在 $O(n^2)$时间内完成，因此 $\boldsymbol{D_L X}$ 的计算复杂度是 $O(n^2)$。类似地，我们可以在 $O(n^2)$时间内求出 $\boldsymbol{X}'=\boldsymbol{D_L X D_R}$。从安全性上看，$\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$ 都是稠密矩阵，对矩阵 $\boldsymbol{X}$ 的变换过程同时包括随机的乘法和加法操作，因此可以避免文献[8]方案的安全问题。用户得到 $\boldsymbol{X}'$后，将其发送给云服务器。在云服务器返回矩阵 $\boldsymbol{X}'$的逆矩阵 $\boldsymbol{Y}'$给用户后，用户可以用如下方法解密 $\boldsymbol{Y}'$，得到 $\boldsymbol{X}$ 的逆矩阵 $\boldsymbol{Y}$。

$$\boldsymbol{Y}=\boldsymbol{D_R Y}'\boldsymbol{D_L}.$$

易于说明，用户可以在 $O(n^2)$时间内完成 $\boldsymbol{Y}=\boldsymbol{D_R Y}'\boldsymbol{D_L}$ 计算。此外，当且仅当矩阵 $\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$ 都是可逆矩阵，$\boldsymbol{D_R Y}'\boldsymbol{D_L}=\boldsymbol{D_R}(\boldsymbol{D_L X D_R})^{-1}\boldsymbol{D_L}=\boldsymbol{X}^{-1}$ 成立。下面，我们将介绍矩阵 $\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$ 中每个元素的生成方法，这个方法可以确保 $\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$ 都是可逆矩阵。

为此，我们先介绍以下引理。

引理 1 ([7])：假设矩阵 $\boldsymbol{M}$ 是一个方阵，那么将 $\boldsymbol{M}$ 的任何一列乘以一个倍数然后加到另外一列，不会改变 $\boldsymbol{M}$ 的行列式。

引理 2 ([9])：假设矩阵 $\boldsymbol{E}$，$\boldsymbol{F}$，$\boldsymbol{G}$，$\boldsymbol{H}$ 的维度分别是 $a\times a$，$a\times b$，$b\times a$，$b\times b$，且矩阵 $\boldsymbol{H}$ 是可逆的，那么

$$\det\begin{bmatrix}\boldsymbol{E} & \boldsymbol{F}\\ \boldsymbol{G} & \boldsymbol{H}\end{bmatrix}=\det(\boldsymbol{H})*\det(\boldsymbol{E}-\boldsymbol{FH}^{-1}\boldsymbol{G}).$$

基于引理 1 和 2，我们可以得到如下定理 1。

定理 1：假设 $\boldsymbol{D_L}=\boldsymbol{L}+\boldsymbol{PQ}$，$\boldsymbol{D_R}=\boldsymbol{R}+\boldsymbol{UV}$，其中 $\boldsymbol{L}$ 和 $\boldsymbol{R}$ 是随机的对角矩阵，$\boldsymbol{P}$ 和 $\boldsymbol{U}$ 是 $n\times 1$ 的随机矩阵，$\boldsymbol{Q}$ 和 $\boldsymbol{V}$ 是 $1\times n$ 的随机矩阵。$\boldsymbol{L}$ 和 $\boldsymbol{R}$ 的对角线元素依次分别是$(L_1,L_2,\cdots,L_n)$和$(R_1,R_2,\cdots,R_n)$，$\boldsymbol{P}=(P_1,P_2,\cdots,P_n)^{\mathrm{T}}$，$\boldsymbol{U}=(U_1,U_2,\cdots,U_n)^{\mathrm{T}}$，$\boldsymbol{Q}=(Q_1,Q_2,\cdots,Q_n)$，$\boldsymbol{V}=(V_1,V_2,\cdots,V_n)$。如果$Q_1=V_1=1$，那么

$$\det(\boldsymbol{D_L})=\left(1+\frac{P_1}{L_1}+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)*\prod_{j=1}^{n}L_j,$$

$$\det(\boldsymbol{D_R})=\left(1+\frac{U_1}{R_1}+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)*\prod_{j=1}^{n}R_j,$$

即 $\det(\boldsymbol{D_L})=(1+\boldsymbol{QL}^{-1}\boldsymbol{P})*\det(\boldsymbol{L})$，$\det(\boldsymbol{D_R})=(1+\boldsymbol{VR}^{-1}\boldsymbol{U})*\det(\boldsymbol{R})$。

证明：由于矩阵 $\boldsymbol{D_L}$ 和 $\boldsymbol{D_R}$ 的情况类似，下面我们先证明矩阵 $\boldsymbol{D_L}$ 的行列式满足定理 1。根据定理 1 的说明，我们可以看出

$$\boldsymbol{D_L}=\begin{bmatrix}P_1+L_1 & Q_2P_1 & \cdots & Q_nP_1\\ P_2 & Q_2P_2+L_2 & \cdots & Q_nP_2\\ \vdots & \vdots & \ddots & \vdots\\ P_n & Q_2P_n & \cdots & Q_nP_n+L_n\end{bmatrix}.$$

基于引理 1，如果我们将矩阵 $\boldsymbol{D_L}$ 的第一列乘以$-Q_i$，然后加到第 i 列$(i=1,2,\cdots,n)$，不会改变矩阵 $\boldsymbol{D_L}$ 的行列式。即在矩阵 $\boldsymbol{D_L}$ 的第 i 列加上 $-Q_i*(P_1+L_1,P_2,\cdots,P_n)$ 后，可以得到

$$\det(\boldsymbol{D_L})=\det\begin{bmatrix}P_1+L_1 & -Q_2L_1 & \cdots & -Q_nL_1\\ P_2 & L_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ P_n & 0 & \cdots & L_n\end{bmatrix}.$$

令矩阵 $\boldsymbol{E}=\left[P_1+L_1\right]$，$\boldsymbol{F}=\left[-Q_2L_1,\cdots,-Q_nL_1\right]$，$\boldsymbol{G}=\left[P_2,\cdots,P_n\right]^{\mathrm{T}}$，$\boldsymbol{H}$ 为一个 $(n-1)*(n-1)$ 的对角矩阵，且 $\boldsymbol{H}$ 的对角元素依次为 $L_2,L_3,\cdots,L_n$。基于引理 2，我们可以得到

$$\det\left(\boldsymbol{D}_L\right)=\det\begin{bmatrix}\boldsymbol{E} & \boldsymbol{F}\\ \boldsymbol{G} & \boldsymbol{H}\end{bmatrix}=\det\left(\boldsymbol{H}\right)*\det\left(\boldsymbol{E}-\boldsymbol{F}\boldsymbol{H}^{-1}\boldsymbol{G}\right).$$

此外，$\det\left(\boldsymbol{E}-\boldsymbol{F}\boldsymbol{H}^{-1}\boldsymbol{G}\right)=P_1+L_1-\sum_{i=2}^{n}\frac{-P_iQ_iD_1}{L_i}$，$\det\left(\boldsymbol{H}\right)=\prod_{j=2}^{n}L_j$。因此，

$$\det\left(\boldsymbol{D}_L\right)=\left(P_1+L_1+\sum_{i=2}^{n}\frac{P_iQ_iD_1}{L_i}\right)*\prod_{j=2}^{n}L_j=\left(1+\frac{P_1}{L_1}+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)*\prod_{j=1}^{n}L_j=\det\left(\boldsymbol{D}_L\right)=\left(1+\boldsymbol{Q}\boldsymbol{L}^{-1}\boldsymbol{P}\right)*\det\left(\boldsymbol{L}\right).$$

相似地，可以证明 $\det\left(\boldsymbol{D}_R\right)=\left(1+\frac{U_1}{R_1}+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)*\prod_{j=1}^{n}R_j=\left(1+\boldsymbol{V}\boldsymbol{R}^{-1}\boldsymbol{U}\right)*\det\left(\boldsymbol{R}\right)$。

综上所述，定理 1 是正确的。

根据定理 1，通过如下方式我们可以确保矩阵 $\boldsymbol{D}_L$ 和 $\boldsymbol{D}_R$ 都是可逆的。

1) 对于 $i=1$ 到 n，随机选择非零的 L_i 和 R_i。

2) 设定 $Q_1=V_1=1$，并对于 $i=2$ 到 n，随机选择非零的$\{P_i, Q_i\}$和$\{U_i, V_i\}$；然后随机选择 P_1 和 U_1，使得 $P_1\neq-L_1*\left(1+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)$且$U_1\neq-R_1*\left(1+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)$。

易于看出，上述方法可以确保 $\det\left(\boldsymbol{D}_L\right)\neq0$ 且 $\det\left(\boldsymbol{D}_R\right)\neq0$，从而可以确保矩阵 $\boldsymbol{D}_L$ 和 $\boldsymbol{D}_R$ 都是可逆的。

3.2. 方案步骤

基于上述思路，我们所提方案的具体步骤如下。

- **Gen:** 用户 1) 随机生成 $n\times n$ 对角矩阵 $\boldsymbol{L}$ 和 $\boldsymbol{R}$，其对角线元素依次分别是 $\left(L_1,L_2,\cdots,L_n\right)$ 和 $\left(R_1,R_2,\cdots,R_n\right)$，并且满足任意的 L_i 和 R_i 都不等于 0。2) 对于 $I=2$ 到 n，分别随机选择非零的$\{P_i, Q_i\}$和$\{U_i, V_i\}$；然后随机选择 P_1 和 U_1，使得 $P_1\neq-L_1*\left(1+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)$且$U_1\neq-R_1*\left(1+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)$。3) 设定矩阵 $\boldsymbol{P}=\left(P_1,P_2,\cdots,P_n\right)^{\mathrm{T}}$，$\boldsymbol{U}=\left(U_1,U_2,\cdots,U_n\right)^{\mathrm{T}}$，$\boldsymbol{Q}=\left(1,Q_2,\cdots,Q_n\right)$，$\boldsymbol{V}=\left(1,V_2,\cdots,V_n\right)$。4) 将密钥 K 设置为 $\mathrm{K}=\left\{\boldsymbol{L},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{U},\boldsymbol{V}\right\}$。
- **Enc：** 用户利用密钥 K 将私有输入 $\boldsymbol{X}$ 加密成 $\boldsymbol{X}'$，使得 $\boldsymbol{X}'=\left(\boldsymbol{L}+\boldsymbol{P}\boldsymbol{Q}\right)\boldsymbol{X}\left(\boldsymbol{R}+\boldsymbol{U}\boldsymbol{V}\right)$。
- **Compute：** 用户将加密后的矩阵 $\boldsymbol{X}'$发送给云服务器。云服务器计算矩阵 X'的逆矩阵 $\boldsymbol{Y}'$，并将矩阵 $\boldsymbol{Y}'$返回给用户。
- **Verify：** 用户随机生成 $n\times1$ 的矩阵 $\boldsymbol{S}=\left(S_1,S_2,\cdots,S_n\right)^{\mathrm{T}}$，然后计算 $\boldsymbol{S}'=\boldsymbol{X}'\left(\boldsymbol{Y}'\boldsymbol{S}\right)$。用户比较 $\boldsymbol{S}'$和 $\boldsymbol{S}$，如果 $\boldsymbol{S}'=\boldsymbol{S}$，则返回值 $\boldsymbol{Y}'$通过验证；否则 $\boldsymbol{Y}'$没通过验证。
- **Dec：** 在验证 $\boldsymbol{Y}'$正确的情况下，用户利用密钥 K 解密云服务器返回的矩阵 $\boldsymbol{Y}'$，得到矩阵 $\boldsymbol{Y}=\left(\boldsymbol{R}+\boldsymbol{U}\boldsymbol{V}\right)\boldsymbol{Y}'\left(\boldsymbol{L}+\boldsymbol{P}\boldsymbol{Q}\right)$。

4. 方案分析

下面，我们将分析本文所提方案的安全性、隐私性和运行效率，并利用模拟实验进行了评估。

4.1. 理论分析

正确性： 我们通过以下两个定理说明本文所提方案的正确性。

定理 2：本文所提方案中 $\boldsymbol{L}+\boldsymbol{PQ}$ 和 $\boldsymbol{R}+\boldsymbol{UV}$ 都是可逆的。

证明：基于定理 1，易于看出

$$\det\left(\boldsymbol{L}+\boldsymbol{PQ}\right)=\left(1+\frac{P_1}{L_1}+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)*\prod_{j=1}^{n}L_j,$$

$$\det\left(\boldsymbol{R}+\boldsymbol{UV}\right)=\left(1+\frac{U_1}{R_1}+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)*\prod_{j=1}^{n}R_j,$$

因为 $L_i\neq 0$，$R_i\neq 0$，$P_1\neq -L_1*\left(1+\sum_{i=2}^{n}\frac{P_iQ_i}{L_i}\right)$ 且 $U_1\neq -R_1*\left(1+\sum_{i=2}^{n}\frac{U_iV_i}{R_i}\right)$，所以确保 $\det\left(\boldsymbol{L}+\boldsymbol{PQ}\right)\neq 0$ 且 $\det\left(\boldsymbol{R}+\boldsymbol{UV}\right)\neq 0$。即 $\boldsymbol{L}+\boldsymbol{PQ}$ 和 $\boldsymbol{R}+\boldsymbol{UV}$ 都是可逆的。

定理 3：通过本文所提方案，用户可以获得正确的矩阵 $\boldsymbol{Y}$，即矩阵 $\boldsymbol{Y}$ 是矩阵 $\boldsymbol{X}$ 的逆矩阵。

证明：在该方案中，我们有 $\boldsymbol{Y}=\left(\boldsymbol{R}+\boldsymbol{UV}\right)\boldsymbol{Y}'\left(\boldsymbol{L}+\boldsymbol{PQ}\right)$ 且 $\boldsymbol{Y}'=\left(\left(\boldsymbol{L}+\boldsymbol{PQ}\right)\boldsymbol{X}\left(\boldsymbol{R}+\boldsymbol{UV}\right)\right)^{-1}$。此外，定理 2 已经证明 $\boldsymbol{L}+\boldsymbol{PQ}$ 和 $\boldsymbol{R}+\boldsymbol{UV}$ 都是可逆的。

因此，$\boldsymbol{Y}=\left(\boldsymbol{R}+\boldsymbol{UV}\right)\left(\boldsymbol{R}+\boldsymbol{UV}\right)^{-1}\boldsymbol{X}^{-1}\left(\boldsymbol{L}+\boldsymbol{PQ}\right)^{-1}\left(\boldsymbol{L}+\boldsymbol{PQ}\right)=\boldsymbol{X}^{-1}$，即矩阵 $\boldsymbol{Y}$ 是矩阵 $\boldsymbol{X}$ 的逆矩阵。

安全性：在我们的方案中，$X'=\left(L+PQ\right)X\left(R+UV\right)$。服务器只能得到 $\boldsymbol{X}'$，对 $\boldsymbol{L}+\boldsymbol{PQ}$ 和 $\boldsymbol{R}+\boldsymbol{UV}$ 中的数据都是未知的。此外，$\boldsymbol{L}+\boldsymbol{PQ}$ 和 $\boldsymbol{R}+\boldsymbol{UV}$ 都是可逆的稠密矩阵，而且它们是用户随机选择的，因此服务器无法从矩阵 $\boldsymbol{X}'$推导出矩阵 $\boldsymbol{X}$ 的有用信息。考虑 $\boldsymbol{X}$ 中某个元素为 0 的特殊情况，在 $\boldsymbol{X}'$中该元素变成了其他元素的随机线性组合，因此云服务器也无法判断矩阵 $\boldsymbol{X}$ 中该值是否为 0。所以，我们的方案可以实现用户输入的隐私性。

相应地，我们有 $\boldsymbol{Y}=\left(\boldsymbol{R}+\boldsymbol{UV}\right)\boldsymbol{Y}'\left(\boldsymbol{L}+\boldsymbol{PQ}\right)$，即 $\boldsymbol{Y}'=\left(\boldsymbol{R}+\boldsymbol{UV}\right)^{-1}\boldsymbol{Y}\left(\boldsymbol{L}+\boldsymbol{PQ}\right)^{-1}$。类似地，由于 $\boldsymbol{R}+\boldsymbol{UV}$ 和 $\boldsymbol{L}+\boldsymbol{PQ}$ 对云服务器是随机的和未知的，所以服务器无法从矩阵 $\boldsymbol{Y}'$推导出矩阵 $\boldsymbol{Y}$ 的有用信息，即本文方案可以保护用户输出的隐私性。

计算复杂度：在我们的方案中，用户需要生成随机的 $\mathrm{K}=\{\boldsymbol{L},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{U},\boldsymbol{V}\}$，并计算 $\boldsymbol{X}'=\left(\boldsymbol{L}+\boldsymbol{PQ}\right)\boldsymbol{X}\left(\boldsymbol{R}+\boldsymbol{UV}\right)$ 和 $\boldsymbol{Y}=\left(\boldsymbol{R}+\boldsymbol{UV}\right)\boldsymbol{Y}'\left(\boldsymbol{L}+\boldsymbol{PQ}\right)$。其中，生成随机 K 的时间复杂度是 $O(n)$，计算 $\boldsymbol{X}'$和 $\boldsymbol{Y}$ 的时间复杂度都是 $O(n^2)$。因此，用户的时间复杂度合计是 $O(n^2)$。云服务器端只需要完成矩阵求逆，计算矩阵 $\boldsymbol{X}'$的逆矩阵 $\boldsymbol{Y}'$，因此云服务器的时间复杂度是 $O(n^{2.373})$。

可以看出，该方案可以有效降低用户端的计算复杂度；同时云服务器只需要执行常规的的矩阵求逆运算，无需承担额外的计算。

4.2. 实验评估

我们进一步通过模拟实验验证了本文所提方案的运行效率。实验平台是配置有 Intel Core(TM) i7-4770 3.4 GHz CPU 和 8.0 GB 内存的计算机，模拟实验运行在 Matlab R2013b 上。

表 1 展示了模拟实验的运行结果。如表 1 所示，本方案可以有效降低用户端的计算时间。随着矩阵维度 n 的增长，用户运行时间降低的幅度逐渐增加。这是因为外包前用户执行矩阵求逆的时间复杂度是 $O(n^{2.373})$，通过本方案借助云服务器后用户的时间复杂度降低到 $O(n^2)$，因此用户时间降低的比例会随着 n 的增加而变大。我们计算了用户直接进行矩阵求逆时间 T_o 和本方案中用户执行时间 T_u 的比值，记为用户加速比。从表 1 可以看出，在 n 取值为 100 到 5000 时，本方案中的用户加速比的为 52.373 到 3858.564 之间。维度 n 越大，用户加速比越大。这也与上述理论分析结果基本相符。

总的来说，本方案的方案可以利用云服务器有效提升用户的计算效率，减少用户的计算开销。

Table 1. Running time of simulation experiments
表 1. 模拟实验运行时间

矩阵维度 n	原始矩阵求逆时间 T_o (毫秒)	本方案运行时间(毫秒)		用户加速比 T_o/T_u
		用户 T_u	云服务器 T_s	
100	1.002	0.019	0.999	52.737
500	197.809	0.663	200.121	298.354
1000	1103.051	1.839	1089.476	599.810
2000	9000.213	7.002	9060.092	1285.377
3000	29108.172	13.598	30001.044	2140.622
5000	136110.830	35.275	138029.425	3858.564

5. 结束语

本文提出了一种新的基于云计算的矩阵求逆安全外包方案。该方案实现了一种新的稠密随机矩阵生成方法及其快速的矩阵乘法，进而提出了利用稠密矩阵对用户矩阵进行安全的变换的方法，可以在支持矩阵求逆外包的同时有效提升用户输入输出的安全性。我们通过理论分析证明了所提方案的安全性和计算复杂度。最后，通过模拟实验，我们证实了所提方案可以有效减少用户的计算开销，提升其运算效率。

基金项目

本文工作受到国家重点研发计划(项目号：2017YFB0802300)的资助。

参考文献

[1] Li, X., Zhu, Y., Wang, J. and Zhang, J. (2019) Efficient and Secure Multi-Dimensional Geometric Range Query over Encrypted Data in Cloud. *Journal of Parallel and Distributed Computing*, **131**, 44-54. https://doi.org/10.1016/j.jpdc.2019.04.015

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2010) A View of Cloud Computing, *Communication of ACM*, **53**, 50-58. https://doi.org/10.1145/1721654.1721672

[3] Zhou, L., Zhu, Y. and Raymond Choo, K.-K. (2018) Efficiently and Securely Harness Cloud to Solve Linear Regression and Other Matrix Operations. *Future Generation Computer Systems*, **81**, 404-413. https://doi.org/10.1016/j.future.2017.09.031

[4] Tao, R., Meng, X.-Y. and Wang, Y. (2010) Image Encryption with Multiorders of Fractional Fourier Transforms. *IEEE Transaction on Information Forensics and Security*, **5**, 734-738. https://doi.org/10.1109/TIFS.2010.2068289

[5] Zhang, X., Qian, Z., Ren, Y. and Feng, G. (2011) Watermarking with Flexible Self-Recovery Quality Based on Compressive Sensing and Compositive Reconstruction. *IEEE Transaction on Information Forensics and Security*, **6**, 1223-1232. https://doi.org/10.1109/TIFS.2011.2159208

[6] 张学奇, 赵梅春. 线性代数[M]. 第 2 版. 北京: 中国人民大学出版社, 2015.

[7] Leon, S.J. (2009) Linear Algebra with Applications. 8th Edition, Pearson Prentice Hall, Upper Saddle River, NJ.

[8] Lei, X., Liao, X., Huang, T., Li, H. and Hu, C. (2013) Outsourcing Large Matrix Inversion Computation to a Public Cloud. *IEEE Transactions on Cloud Computing*, **1**, 78-87. https://doi.org/10.1109/TCC.2013.7

[9] Brookes, M. (2011) The Matrix Reference Manual. Imperial College London, London.

知网检索的两种方式：

1. 打开知网首页：http://cnki.net/，点击页面中“外文资源总库 CNKI SCHOLAR”，跳转至：http://scholar.cnki.net/new，搜索框内直接输入文章标题，即可查询；
   或点击“高级检索”，下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询。
2. 通过知网首页 http://cnki.net/顶部“旧版入口”进入知网旧版：http://www.cnki.net/old/，左侧选择“国际文献总库”进入，搜索框直接输入文章标题，即可查询。

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org