A Secure Algorithm for Outsourcing Matrix Multiplication Computation in the Cloud

Shaojing Fu
College of Computer
National University of Defense Technology
Changsha,China
Sate Key Laboratory of Cryptology
Beijing,China
Science and Technology on Information Assurance Laboratory
Beijing,China

Yunpeng Yu[*]
College of Computer
National University of Defense Technology
Changsha,China
yuyunpeng@nudt.edu.cn

Ming Xu
College of Computer
National University of Defense Technology
Changsha,China

ABSTRACT

Matrix multiplication computation (MMC) is a common scientific and engineering computational task. But such computation involves enormous computing resources for large matrices, which is burdensome for the resource-limited clients. Cloud computing enables computational resource-limited clients to economically outsource such problems to the cloud server. However, outsourcing matrix multiplication to the cloud brings great security concerns and challenges since the matrices and their products often usually contains sensitive information. In a previous work, Lei et al. [1] proposed an algorithm for secure outsourcing MMC by using permutation matrix and the authors argued that it can achieve data privacy. In this paper, we first review the design of Lei's scheme and find a security vulnerability in their algorithm that it reveals the number of zero element in the input matrix to cloud server. Then we present a new verifiable, efficient, and privacy preserving algorithm for outsourcing MMC, which can protect the number privacy of zero elements in original matrices. Our algorithm builds on a series of carefully-designed pseudorandom matrices and well-designed privacy-preserving matrix transformation. Security analysis shows that our algorithm is practically-secure, and offers a higher level of privacy protection than the state-of-the-art algorithm.

Categories and Subject Descriptors

D.4.6 [**Operating systems**]: Security and Protection—*Security kernels, Verification* ; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*Determinants, Sparse, structured, and very large systems (direct and iterative methods)*

General Terms

Design, Security, Algorithms

Keywords

Cloud computing; Privacy-preserving;Matrix multiplication; Outsourcing computation

[*]Corresponding author

SCC'17, April 02 2017, Abu Dhabi, United Arab Emirates
© 2017 ACM. ISBN 978-1-4503-4970-3/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3055259.3055263

1. INTRODUCTION

With cloud computing becoming more widely utilized, more and more users with computational resource-constraint devices tend to outsource their computing needs to the cloud server which has plenty of computing resources in a pay-per-use manner, relieving the clients from computation burden. However, directly outsourcing computation to the cloud inevitably brings in new security concerns and challenges[2, 3]. The first concern is the *privacy* of input and output data. Data privacy has become a critical issue for cloud users when they host their data on remote and untrusted cloud storage. To protect the privacy of sensitive data, the client should encrypt the sensitive data before outsourcing and decrypt the returned results from the cloud after outsourcing. The second concern is the *verification* of the outsourcing computation results. The cloud is not fully trusted. For example, for the financial incentives, the cloud may decrease the amount of the computations and then return invalid results. Consequently, the client needs to verify the correctness of the returned outputs. The third concern is *efficiency* of outsourcing computation. In the outsourcing process, the computation on the client side must be substantially smaller than performing the original computational problem on its own.

Over the past few years, many computational tasks have been designed for secure outsourcing. For example, secure outsourcing large-scale systems of Modular Exponentiations[4, 5], bilinear pairings[6, 7], Attribute-based encryprion[8, 9], linear equations [10, 11, 12, 13], large matrix inversion [14], large Matrix Determinant [3] and linear programming [15]. Matrix multiplication computation (MMC) of the form XY is one of the most basic algebraic problems in common scientific and engineering computing task. MMC is widely used in many applications, including sliding mode analysis, discriminant analysis, 3D graphics simulations and so on. In practice, MMC requires too much computation resources for the resource-constraint clients. Despite the tremendous benefits of cloud computing, directly outsourcing MMC to the cloud inevitably brings in new security challenges [2]. The original matrices X and Y usually contain sensitive information, but the cloud is not fully trusted. Therefore, it is very important for the clients to efficiently outsource MMC to the cloud while preserving the data privacy.

In recent years, many secure outsourcing algorithms have been proposed for solving MMC. Atallah et al. [16] first investigated the problem of secure outsourcing for solving MMC. However, directly utilizing their scheme for large MMC will cause large amount of memory overhead and their scheme reveals too much sensitive information to the untrusted parties. Benjamin et al. [17] proposed a secure algorithm for outsourcing MMC to two non-colluding servers which cannot resist the collusive attack. Atallah et al. [18] proposed an algorithm for securely outsourcing MMC to the cloud, using Shamirąŕs secret sharing technique. But it requires extremely heavy communication overhead and only works over finite field $\mathbb{Z}_p$. Recently, Lei et al. [1] designed an algorithm to securely outsource MMC by introducing permutation matrix to preserve the data privacy. We show that, however, the security arguments given for Lei et al's work do not hold, the cloud can obtain the number of zero elements in original matrix from the encrypted matrix. Therefore, we are motivated to design a new algorithm that enables clients to outsource privacy-preserving large-scale matrix multiplication computation to the cloud server while apparently relieving the client of its high computation burden.

Our algorithm is designed to meet the efficiency and security requirements under the computation outsourcing model. Our design delegates the most expensive computation for MMC to the cloud to get rid of computation costs. To protect the data privacy, the original matrix is hidden by well designed privacy-preserving matrix transformation. Security analysis proves that our algorithm can properly protect the privacy of input/output data of outsourced MMC. The contributions of this paper can be summarized as follows:

1) We formulate the problem of privacy-preserving MMC outsourcing. We examine the state-of-the-art algorithm and show its security weakness when meeting the practical needs under the MMC outsourcing model.

2) We propose a privacy-preserving algorithm for secure outsourcing of MMC in cloud systems, which reduces the computation burden on the client side. And the computation complexity for the client is no more than $O(n^2)$.

3) Through a series of disguise-based techniques, our algorithm can protect data privacy of MMC, especially the number of zero elements in the original matrix.

4) We provide extensive theoretical analysis and experimental evaluation to demonstrate its high efficiency and security compared to the previous works.

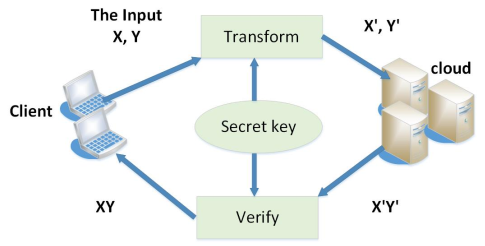

Figure 1: A secure system model for outsourcing MMC

The rest of this paper is organized as follows: In Section 2, we present the system and threat model together with the design goals of our scheme. Section 3 gives the preliminaries and notations that are used in our paper. In Section , we provide a close overview of the MMC proposed by Lei et al. [1], and identify its security weaknesses. Section provides the detailed construction of our scheme together with the theoretical analysis. Section analyzes the performance of our scheme in experiments. Finally, we conclude our work in Section 8.

2. PROBLEM FORMULATION

In this section, we present the system and threat model for secure outsourcing of MMC, and introduce our design goals.

2.1 System Model

We consider an asymmetric computing outsourcing architecture involving two different parties, as shown in Fig.1. The cloud server contains the unlimited computing resources in a pay-per-use manner. Meanwhile, the computational resource-limited client cannot carry out the heavy computation of MMC locally, where the complexity is usually $O(n^\rho)$ $(2 < \rho \leqslant 3)$. Therefore, the client can outsource the MMC to the cloud server. During the outsourcing process, the client outsources the most expensive computation to the cloud server and retrieves the solution from the returned results. And the client has to preserve the input and output data privacy.

In this work, we pay more attention on finding a scheme to securely and effectively outsource the MMC for $X \in \mathbb{R}^{m\times n}$ and $Y \in \mathbb{R}^{n\times s}$. Instead of directly sending the original X, Y to the cloud server, the client first uses k (k is a small integer) random vectors to transform the X, Y into the encrypted version X', Y'. The encrypted version X', Y' is sent to the cloud server. After receiving the returned result of encrypted matrix multiplication $X'Y'$ from the cloud server, the client should be able to first verify whether the result is correct. If the returned result is correct, the client then uses the secret k random vectors to get the desired solution for the original computation.

2.2 Threat Model

The security threats faced by the outsourcing computation system model primarily come from the behavior of the cloud server. There exist two levels of threat models in outsourcing: honest but curious (semi-honest) cloud model and malicious cloud model[19]. In the honest but curious cloud model, the cloud server is guaranteed to properly conducts the process specification. However, the cloud may try its best to derive sensitive information from the input data and the result of its own computations. In the malicious cloud model, the cloud server behaves unfaithfully or intentionally sabotage the computation for a rational economic agent, e.g. to return an erroneous result to save its computing resources, while hoping not to be detected by the client.

In this paper, we can view the threat model may behave two levels of threat models. That is, the cloud server which can be attacked/compromised may attempt to retrieve sensitive information from the original matrix, and the multiplication XY. The cloud may also behave unfaithfully to return an erroneous result to save the computing resources. We also assume that the cloud knows the outsourcing algorithm. Specifically, the privacy threats for MMC come from the sensitive information in the original matrix X, Y, and the multiplication XY. Our algorithm should be practically-secure and handle result verification against the malicious behave of the cloud.

2.3 Design Goals

In this paper, we identify the following goals that the outsourcing scheme should achieve:

1) Correctness: If the client and the cloud honestly follow the scheme, the cloud should return the correct solution and the client can obtain the correct result of the original MMC.

2) Security: The cloud cannot derive any sensitive information of the original matrix X, Y from the transformed X', Y' , including the number of zero element in the original matrix.

3) Efficiency: The local computation done by the client (including key generation and transformation computation) should be substantially less than the original MMC on his own. In particular, the overall computational complexity for the client is no more than $O(n^2)$.

4) Verification: The returned solution from the faithful cloud server must be decrypted and verified successfully by the client. No erroneous solution from the cheating cloud server can pass the verification with a non-negligible probability.

3. PRELIMINARIES AND NOTATIONS

In this section, we describe some preliminaries and notations that are used in this paper.

Kronecker delta function: In mathematics, the Kronecker delta function is a function of two variables, usually just positive integers. The function is 1 if the variables are equal, and 0 otherwise:

$$\delta_{x,y} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{1}$$

Random permutation: Random permutation function is well studied in Combinatorial mathematics and group theory. In Cauchy's two-line notation, there are the preimage elements in the first row, and the image elements of the preimage elements in the second row. The random permutation function can be expressed as:

$$\pi : \begin{pmatrix} 1 & \cdots & n \\ p_1 & \cdots & p_n \end{pmatrix} \tag{2}$$

where $\pi(i) = p_i (i = 1, \cdots, n)$. We define π^{-1} as the inverse function of π. When the condition satisfies $\pi(i) = i$, the permutation is called identical permutation denoted by I. The description of random permutation generation is as follows:

Algorithm 1 Random Permutation generation

```
1: set $\pi = I$
2: for $i = n : 2$ do
3:     select a random integer $j$ where $1 \leq j \leq i$;
4:     swap $\pi(i)$ and $\pi(j)$;
5: end for
```

Computational indistinguishability: To enable the client to securely delegate computing tasks to the cloud server, the outsourced data should appear random. The notion of privacy is defined as computational indistinguishability[20]. The notion of computational indistinguishability is central to the theory of cryptography. Informally speaking, two probability distributions are computationally indistinguishable if no efficient algorithm can tell them apart (or distinguish them). Two probability ensembles $X = \{X_{\rm n}\}_{n \in N}$ and $Y = \{Y_{\rm n}\}_{n \in N}$ are computationally indistinguishable if for every probabilistic polynomial time distinguisher D there exists a negligible function *negl* such that

$$|Pr[D(X_{\rm n}) = 1]| - |Pr[D(Y_{\rm n}) = 1]| < negl(n) \tag{3}$$

The probability ensemble is consisted of a series of random variables $X_1, X_2, \cdots$, and is denoted by $X = \{X_{\rm n}\}_{n \in N}$. The notation $D(X_{\rm n})$ denotes that x is distinguished from distribution $X_{\rm n}$. Distinguisher $D(X)$ outputs 1 if it can distinguish x form distribution $X_{\rm n}$ in a probabilistic polynomial-time.

Moreover, the notation can be extended to the case where a distinguisher D has access to multiple samples of the vectors X and Y. That is, to distinguish two matrices[12].

Definition 1. *Let $R \in \mathbb{R}^{n \times n}$ be a random matrix with elements in its jth column with interval $[-R_j, R_j]$ ($\forall j \in [1, n]$). Matrices R and $Q \in \mathbb{R}^{n \times n}$ are computationally indistinguishable if for every probabilistic polynomial time distinguisher $D(\cdot)$, there exists a negligible function negl such that*

$$|Pr[D(r_{i,j}) = 1]| - |Pr[D(q_{i,j}) = 1]| \leq negl \tag{4}$$

where $i \in [1, n]$,$j \in [1, n]$, $r_{i,j}$ is the element in the ith row and jth column of R, and $q_{i,j}$ is the element in the ith row and jth column of Q. Distinguisher $D(\cdot)$ returns 1 when it identifies the input as a uniform distribution in the range $[-R_j, R_j]$.

4. REVIEW OF THE ALGORITHM PROPOSED BY LEI ET AL.

In this section, we first describe the secure algorithm proposed by Lei et al. The input data of MMC is a matrix $X \in \mathbb{R}^{m \times n}$ and a matrix $Y \in \mathbb{R}^{n \times s}$.

The completed algorithm proposed by Lei et al.[1] contains five sub-algorithms, namely KeyGen, MMCEnc, MMCSolve, MMCDec, ResultVerify. The detail of the algorithm as follow:

KeyGen: Given a security parameter 1^λ, which specifies key space $\mathrm{K}_\alpha, \mathrm{K}_\beta$ and K_γ, the client selects three set-s of non-zero random numbers: $\{\alpha_1, \alpha_2, \cdots, \alpha_m\} \leftarrow K_\alpha$, $\{\beta_1, \beta_2, \cdots, \beta_n\} \leftarrow K_\beta$, $\{\gamma_1, \gamma_2, \cdots, \gamma_s\} \leftarrow K_\gamma$. Then, the client generates three random permutation: $\pi_1 \leftarrow R\text{and}P(1, \cdots m)$, $\pi_2 \leftarrow R\text{and}P(1, \cdots n)$, $\pi_3 \leftarrow R\text{and}P(1, \cdots s)$.

MMCEnc: The client first generates invertible matrices P_1, P_2, P_3, where $P_1(i,j) = \alpha_i \delta_{\pi_1(i),j}, P_2(i,j) = \beta_i \delta_{\pi_2(i),j}$, $P_3(i,j) = \gamma_i \delta_{\pi_3(i),j}$. Then, the client computes $X' = P_1 X P_2^{-1}$ and $Y' = P_2 Y P_3^{-1}$. Later, the encrypted MMC problem $\Phi_K(X', Y')$ will be outsourced to the cloud. For P_1, P_2 and P_3, their inverse matrices can be calculated as follows:

$$\begin{cases} P_1^{-1}(i,j) = (\alpha_j)^{-1} \delta_{\pi_1^{-1}(i),j} \\ P_2^{-1}(i,j) = (\beta_j)^{-1} \delta_{\pi_2^{-1}(i),j} \\ P_3^{-1}(i,j) = (\gamma_j)^{-1} \delta_{\pi_3^{-1}(i),j} \end{cases}$$

MMCSolve: Given the MMC problem $\Phi_K(X', Y')$, the cloud can call any method to compute $Z' = X'Y'$. Then the cloud sends matrix Z' back to the client.

MMCDec: Given the returned matrix Z' from the cloud, the client computes $Z = P_1^{-1} Z' P_3$.

ResultVerify: The client computes $P = X \times (Y \times r) - Z \times r$, where r is a random vector. If $P = (0, \cdots 0)^T$ holds, the client accepts Z as the correct solution; Otherwise, rejects it and claims a failure to the cloud.

Security Requirements is the privacy for the input/ output data of the computation task of the client C. Informally, it means that the server S cannot learn anything from outsourcing process in the protocol in the sense of indistinguishability argument.

In the following theorem, we will show the privacy are not protected well in Leiąŕs algorithm scheme, since it reveals the number of zero element in the input matrix to cloud server.

Theorem 1. *In Lei et al.'s scheme, the number of zero element in matrix X' (Y') is the same as that in matrix X (Y) .*

Proof. Let

$$X = \begin{bmatrix} \mathrm{x}_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix},$$

$P_1(i,j) = \alpha_i \delta_{\pi_1(i),j}$, one can obtain

$$P_1 X = \begin{bmatrix} \alpha_1 x_{\pi_1(1),1} & \cdots & \alpha_1 x_{\pi_1(1),n} \\ \vdots & \ddots & \vdots \\ \alpha_i x_{\pi_1(i),1} & \cdots & \alpha_i x_{\pi_1(i),n} \\ \vdots & \ddots & \vdots \\ \alpha_n x_{\pi_1(n),1} & \cdots & \alpha_n x_{\pi_1(n),n} \end{bmatrix}.$$

Note that $P_2^{-1}(i,j) = (\beta_j)^{-1} \delta_{\pi_2^{-1}(i),j}$, this leads to

$$X' = P_1 X P_2^{-1}$$
$$= \begin{bmatrix} \frac{\alpha_1}{\beta_1} x_{\pi_1(1),\pi_2(1)} & \cdots & \frac{\alpha_1}{\beta_j} x_{\pi_1(1),\pi_2(j)} & \cdots & \frac{\alpha_1}{\beta_n} x_{\pi_1(1),\pi_2(n)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\alpha_i}{\beta_1} x_{\pi_1(i),\pi_2(1)} & \cdots & \frac{\alpha_i}{\beta_j} x_{\pi_1(i),\pi_2(j)} & \cdots & \frac{\alpha_i}{\beta_n} x_{\pi_1(i),\pi_2(n)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\alpha_n}{\beta_1} x_{\pi_1(n),\pi_2(1)} & \cdots & \frac{\alpha_n}{\beta_j} x_{\pi_1(n),\pi_2(j)} & \cdots & \frac{\alpha_n}{\beta_n} x_{\pi_1(n),\pi_2(n)} \end{bmatrix}$$

Thus, we have

$$X'(i,j) = (\alpha_i / \beta_j) X(\pi_1(i), \pi_2(j))$$

for any i, j. Then for $X(\pi_1(i), \pi_2(j)) = 0$, we have $X'(i,j) = 0$.

We give an numerical example to illustrate the inherent flaw. Assume that $X = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 3 & 4 \end{bmatrix}$, the client generates the secret keys: $\{\alpha_1, \alpha_2\} = \{1, 2\}$, $\{\beta_1, \beta_2, , \beta_3\} = \{3, 4, 5\}$, $\pi_1 = [2, 1]$, $\pi_2 = [3, 1, 2]$. Then the client has that $P_1 = \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$, $P_2 = \begin{bmatrix} 0 & 0 & 3 \\ 4 & 0 & 0 \\ 0 & 5 & 0 \end{bmatrix}$, $P_2^{-1} = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{5} \\ \frac{1}{3} & 0 & 0 \end{bmatrix}$. The client computes $X' = P_1 X P_2^{-1} = = \begin{bmatrix} \frac{4}{3} & 0 & \frac{3}{5} \\ \frac{4}{3} & \frac{1}{2} & 0 \end{bmatrix}$. Thus, the number of zero element of X' is same as that of matrix X.

We prove that the privacy of X of MMC is not protected. Let matrices P_1, P_2, P_1', P_2' be four matrices generated by C, where $P_1(i,j) = \alpha_i \delta_{\pi_1(i),j}, P_2(i,j) = \beta_i \delta_{\pi_2(i),j}, P_1'(i,j) = \alpha'_i \delta_{\pi'_1(i),j}, P_2'(i,j) = \beta'_i \delta_{\pi'_2(i),j}$. Given two matrices $X = (x_{i,j})$ and $X' = (x'_{i,j})$ which are chosen by the adversary A, C computes $T = P_1 X P_2^{-1}$ and $T' = P_1' X' P_2'^{-1}$ where

$$T(i,j) = (\alpha_i / \beta_j) X(\pi_1(i), \pi_2(j))$$

and

$$T'(i,j) = (\alpha_i' / \beta_j') X(\pi'_1(i), \pi'_2(j)).$$

According to the number of zero element in matrices X and X', the adversary A can distinguish T from T'.

Similarly, we can prove that the number of zero element in matrix Y' is the same as that in matrix Y and the privacy of X of MMC is not protected. Thus, the cloud can obtain the number of zero element in matrices X and Y in Lei et al.'s scheme [1] and the client cannot protect the privacy of the original matrices X and Y.

5. THE PRIVACY PRESERVING SCHEME FOR OUTSOURCING MMC

In what follows, we describe our new proposed privacy preserving scheme for outsourcing MMC which can hide the numbers of zero element in matrix from the cloud. We first describe the notation that is used in this letter.

Privacy-preserving Matrix Transformation allows the client to hide the data of the matrix in the sense of computational indistinguishability. Specifically, assume that the values of matrix A are within the range $[-K, K]$, where $A \in \mathbb{R}^{n \times n}$ and $K = 2^l (l > 0)$ is a positive constant. Then the client hides the privacy of matrix A by adding a random matrix as $\hat{A} = A + Z$, where the random matrix Z is constructed as follows:

$$Z = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix} \qquad (2 \leq k \ll n)$$

where $u_1, u_2, \cdots, u_k \in \mathbb{R}^{m \times 1}$ are the vectors of uniformly distributed random variables ranging from -2^p and 2^p ($p > 0$), $v_1, v_2, \cdots, v_k \in \mathbb{R}^{n \times 1}$ are the vectors of arbitrary positive constants ranging from 2^l and 2^{l+q} ($q > 0$).

Our scheme is also consisted of five stages, namely KeyGen, MMCEnc, MMCSolve, ResultVerify, MMCDec.

KeyGen: The client specifies a positive integer k ($2 \leq k \ll \min(m,n,s)$) and then picks $4k$ vectors $\{a_1, \cdots, a_k \in \mathbb{R}^{m\times 1}\}$, $\{b_1, \cdots, b_k \in \mathbb{R}^{n\times 1}\}$, $\{c_1, \cdots, c_k \in \mathbb{R}^{n\times 1}\}$, $\{d_1, \cdots, d_k \in \mathbb{R}^{s\times 1}\}$, where $\{a_1, \cdots, a_k\}$, $\{c_1, \cdots, c_k\}$, $\{b_1, \cdots, b_k$, and $d_1, \cdots, d_k\}$ are constructed as shown u, v in the notion **Privacy-preserving Matrix Transformation**, respectively.

MMCEnc: The client first calculate two matrices Z_1 and Z_2 as follows.

$$Z_1 = \begin{bmatrix} a_1 & \cdots & a_k \end{bmatrix} \begin{bmatrix} b_1^T \\ \vdots \\ b_k^T \end{bmatrix}, Z_2 = \begin{bmatrix} c_1 & \cdots & c_k \end{bmatrix} \begin{bmatrix} d_1^T \\ \vdots \\ d_k^T \end{bmatrix}$$

Then, the client computes $X' = X + Z_1$ and $Y' = Y + Z_2$. Later, the encrypted MMC problem $\Phi_K(X', Y')$ will be outsourced to the cloud.

MMCSolve: Given the encrypted MMC problem $\Phi_K(X', Y')$, the cloud can call any method to compute $Z' = X'Y'$. Then the cloud sends matrix Z' back to the client.

ResultVerify: The client computes $P = X' \times (Y' \times r) - Z \times r$, if $P = (0, \cdots 0)^T$ holds, the client accepts Z as the correct solution; Otherwise, rejects it and claims a failure to the cloud.

MMCDec: Given the returned matrix Z' from the cloud, the client computes $XY = Z' - S$. The client computes S as

$$\begin{aligned} S &= (X + Z_1)Z_2 + Z_1 Y \\ &= (X' \begin{bmatrix} c_1 & \cdots & c_k \end{bmatrix}) \begin{bmatrix} d_1^T \\ \vdots \\ d_k^T \end{bmatrix} + \begin{bmatrix} a_1 & \cdots & a_k \end{bmatrix} (\begin{bmatrix} b_1^T \\ \vdots \\ b_k^T \end{bmatrix} Y) \end{aligned}$$

Note that S is computed through matrix-vector multiplication and the complexity is $O(n^2)$.

THEOREM 2. *The proposed scheme is correct.*

Proof. If the client and the cloud follow the scheme honestly, we have

$$\begin{aligned} Z &= Z' - S \\ &= X'Y' - S \\ &= (X + Z_1)(Y + Z_2) - S \\ &= XY + (X + Z_1)Z_2 + Z_1 Y - S \\ &= XY \end{aligned}$$

This implies that the decryption process will always yield the correct result and hence the proposed scheme is correct.

6. SECURITY ANALYSIS

We prove the privacy for X of MMC. Let $Z = (z_{i,j})$, $Z' = (z'_{i,j})$ be two matrices generated by the client. Given two matrices X= ($\mathrm{a}_{i,j}$), X'= ($\mathrm{a}'_{i,j}$) which are chosen by the cloud server, the client computes $T = X + Z = (t_{i,j})$ and $T' = X' + Z' = (t'_{i,j})$, where

$$t_{ij} = x_{ij} + z_{i,j}$$

and

$$t'_{i,j} = x'_{i,j} + z'_{i,j}.$$

The Z, Z' are constructed as shown **Privacy-preserving Matrix Transformation**, thus $t_{i,j}$ and $t'_{i,j}$ are are computationally indistinguishable. Therefore, the advantage of the cloud server to distinguish between T and T' is negligible.

Similarly, we can prove the privacy for Y of MMC. Since matrices Z_1 and Z_2 are kept private by the client, the cloud cannot derive any sensitive information about the element of matrices X and Y from the transformed matrices X' and Y'.

Remark 1: When $k = 1$, X' and Y' are computationally indistinguishable from a random matrix. However, the transform can leak the linear relation between the rows and columns of Z if X or Y is a large sparse matrix. That is, when $k = 1$, it leaks the numbers of the zero element. In our scheme, let $2 \leq k \ll \min(m,n,s)$

Remark 2: Note that Z_1, Z_2, r in our scheme can be only used one time. Thus such parameters are freshly generated for each time of outsourcing MMC.

Remark 3: It is clear that Lei's scheme reveals the rank of the input matrix to cloud server since $Rank(P_1 X P_2^{-1}) = Rank(X)$. Note that $Rank(X+Z) \neq Rank(X)$ and $Rank(X) - kRank(X+Z) \neq Rank(X) + k$, Our scheme can hide the rank of the input matrix in some sense.

7. EFFICIENCY ANALYSIS AND PERFORMANCE EVALUATION

We first present the computational complexity analysis for the proposed scheme in theory. Our efficiency analysis follows the methodology of Serigo et. al [12], which defines the computational complexity of a party as the number of floating-point (flops) operations (additions, subtractions, multiplications, and divisions), bitwise operations and encryptions that the party needs to perform. In the **KeyGen**, the client generates $4k$ vectors $a_1, \cdots, a_k$, $b_1, \cdots, b_k$, $c_1, \cdots, c_k$, $c_1, \cdots, c_k$. To get b_1, the client takes $m \cdot S$ bitwise operations, where S is the number of bitwise operations to generate a random number by methods such as Mersenne Twister[22]. Thus, to get $4k$ vectors, the client takes $k(m + n + n + s) \cdot S$ bitwise operations. In the **MMCEnc**, the client generates the matrices Z_1 and Z_2 which takes $k(mn + ns)$ floating-point operations, and computes $X' = X + Z_1$ and $Y' = Y + Z_1$ which takes $mn+ns$ multiplications operations. Therefore, in the **problem generation**, the computational complexity for the client is $k(m+2n+s) \cdot S + (k+1)(mn+ns) \cdot flops$. The **MMCSolve** is conducted by the cloud. In the **ResultVerify**, the client first generates a random vector r, which takes $s \cdot S$ bitwise operations, then takes $ns + mn + ms$ floating-point operations to compute $P = X' \times (Y' \times r) - Z \times r$. In the **MMCDec**, the client computes $S = X'Z_2 + Z_1 Y$, which takes $k(mn + 2ms + ns)$ floating-point operations. Table 1 presents the comparison of computation complexity for the client, privacy-preserving between Lei et al. and our proposed scheme.

We evaluate the performance of our proposed scheme through experiments. The algorithms are all programmed with Matlab R2014a. The client-side workstation is configured as a virtual machine equipped with a 64 GB RAM and 8 cores (each runs at 2.1GHz). Fig 2 clearly illustrates the total running time of the client of our algorithm with that of [1]. Each element in matrices and vectors randomly located in $(0, 1)$. Although accordance to our theoretical results, we observe that the running time for the client of Lei et al.[1]

Table 1: Comparison of securely outsourcing Matrix Multiplication

	Lei et al.	Our Scheme
Problem Generation	$(m+n+s)\cdot S+$ $(3(m+n+s)+2mn+2ns)$	$k(m+2n+s)\cdot S+$ $(k+1)(mn+ns)\cdot flops$
Problem Verify	$s\cdot S+ns+mn+ms\cdot flops$	$s\cdot S+ns+mn+ms\cdot flops$
Problem Solve	$2(mn+ns)\cdot flops$	$k(mn+2ms+ns)\cdot flops$
Privacy-preserving	$\times$	✓

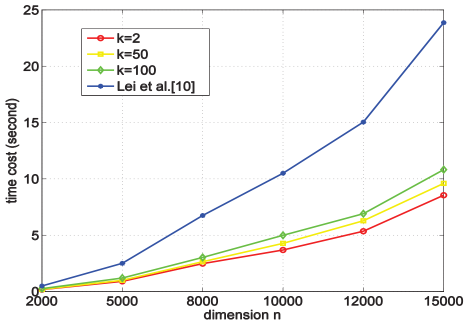

Figure 2: The total running time of our scheme compared with that of [1](m:n:s=4:5:6)

is more efficient than that of our scheme. The experiment results demonstrate that our algorithm is more efficient compared to that of [1].

Remark 3: The choice of k is a tradeoff between security and efficiency. In the proposed algorithm, we suggest that the client can choose k as $k \leq 1\% \min(m,n,s)$.

8. CONCLUSIONS

In this paper, we have presented the security flaw of the algorithm proposed by Lei et al. Although Lei et al. claimed that their scheme is more efficient than others and can achieve security, we have demonstrated that the algorithm cannot protect the number of zero element in original matrices. Therefore, the algorithm cannot preserve the privacy of the outsourced data. We then propose a new privacy-preserving algorithm for outsourcing matrix multiplication computation (MMC) to the cloud. By delegating the most expensive computation of MMC to the cloud, our algorithm relieves the client of its high computation burden. Moreover, with a series of carefully-designed random matrices, our algorithm can properly protect the privacy of input/output data of outsourced MMC. Particularly, it can hide the number privacy of zero elements in the original matrix. Extensive experiments demonstrate that our algorithm achieves higher efficiency than the existing scheme in the client-side computation.

9. ACKNOWLEDGMENTS

This work is supported by the National Nature Science Foundation of China (NSFC) under grant 61379144 and 61572026, Open Foundation of State Key Laboratory of Cryptology (No:MMKFKT201617)and the Foundation of Science and Technology on Information Assurance Laboratory.

10. REFERENCES

[1] 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.

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

[3] Lei X, Liao X, Huang T, et al. Cloud Computing Service: the Case of Large Matrix Determinant Computation[J]. IEEE Transactions on Services Computing, 2015, 8(5):688-700.

[4] Chen X, Li J, Ma J, et al. New algorithms for secure outsourcing of modular exponentiations[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(9): 2386-2396.

[5] Wang Y, Wu Q, Wong D, et al. Securely outsourcing exponentiations with single untrusted program for cloud storage[C]//European Symposium on Research in Computer Security. Springer International Publishing, 2014: 326-343.

[6] Chen X, Susilo W, Li J, et al. Efficient algorithms for secure outsourcing of bilinear pairings. Theor. Comput. Sci. 562: 112-121 (2015)

[7] Ren Y, Ding N, Wang T, et al. New algorithms for verifiable outsourcing of bilinear pairings[J]. Science China Information Sciences, 2016, 59(9): 99103.

[8] Lai J, Deng R, Guan C, Weng J. Attribute-based encryprion with verifiable outsourced decryptiončőIEEE Transactions on Informarion Forensics and Security, 2013čň8(8)čž1343-1354čő

[9] Sherman S. M. Chow: A Framework of Multi-Authority Attribute-Based Encryption with Outsourcing and Revocation. SACMAT 2016: 215-226

[10] Wang C, Ren K, Wang J, et al. Harnessing the Cloud for Securely Solving Large-Scale Systems of Linear Equations[C] IEEE International Conference on Distributed Computing Systems. 2011:549-558.

[11] Chen X, Huang X, Li J, et al. New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations. IEEE Trans. Information Forensics and Security, 2015, 10(1): 69-78

[12] Salinas S, Luo C, Chen X, et al. Efficient secure outsourcing of large-scale linear systems of equations[C] Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015.

[13] Yu Y, Luo Y, Wang D, et al. Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations[C]//Communications (ICC), 2016 IEEE International Conference on. IEEE, 2016: 1-6.

[14] 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.

[15] Wang C, Ren K, Wang J. Secure and practical outsourcing of linear programming in cloud computing[C] INFOCOM, 2011 Proceedings IEEE. IEEE, 2011:820-828.

[16] Atallah M J, Pantazopoulos K N, Rice J R, et al. Secure outsourcing of scientific computations *[J]. Advances in Computers, 2002, 54(01):215-272.

[17] Benjamin D, Atallah M J. Private and Cheating-Free Outsourcing of Algebraic Computations[C] Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on. IEEE, 2008:240 - 245.

[18] Atallah M J, Frikken K B. Securely outsourcing linear algebra computations[C] Acm Symposium on Information. ACM, 2010:48-59.

[19] Lindell Y, Pinkas B. Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality, 25(2):761ĺC766, 2008.

[20] Katz J. Introduction to modern cryptography, second edition. Crc Press, 2014.

[21] Gennaro R, Gentry C, Parno B. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers[C] Advances in Cryptology-CRYPTO 2010, Cryptology Conference, Santa Barbara, Ca, Usa, August 15-19, 2010. Proceedings. 2010:465-482.

[22] Matsumoto, Makoto, Nishimura, et al. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator[J]. Acm Transactions on Modeling and Computer Simulation, 1998, 8(1):3-30.