Privacy preserving, verifiable and efficient outsourcing algorithm for matrix multiplication to a malicious cloud server

Malay Kumar, Jasraj Meena and Manu Vardhan

Cogent Engineering (2017), 4: 1295783
Privacy preserving, verifiable and efficient outsourcing algorithm for matrix multiplication to a malicious cloud server

Malay Kumar¹*, Jasraj Meena¹ and Manu Vardhan¹

Abstract: Matrix Multiplication is a basic engineering and scientific problem, which has application in various domains. There exists many cryptographic solutions for secure computation of matrix multiplication, but cryptographic preamble makes them infeasible for outsourcing with large input size to the cloud server. In this paper, we propose a privacy-preserving, verifiable and efficient algorithm for matrix multiplication in outsourcing paradigm illustrated by the following scenario: the client is having a large data-set and needs to perform matrix multiplication, but unable to process due to the lack of computing resources. Therefore, the client outsources the computation to the cloud server. We evaluate the algorithm on security, efficiency and verifiability parameters and discuss the implementation details. The result analysis shows that the algorithm is highly efficient and endorses the practical usability of the algorithm. Using this algorithm, we can mostly replace the costly cryptographic operations and securely solve matrix multiplication on a large data-set.

Subjects: Computer Science; Algorithms & Complexity; Computing & IT Security; Computation; Simulation & Modeling

Keywords: matrix multiplication; secure outsourcing; cloud computing; verifiable computation

ABOUT THE AUTHOR

Malay Kumar has received M Tech degree in Computer Engineering from National Institute of Technology, Kurukshetra, Haryana, India, in 2012. He is currently working towards his PhD degree in Computer Science and Engineering from National Institute of Technology, Raipur, Chhattisgarh, India. His current research interest includes secure outsourcing of mathematical and scientific application, encryption schemes, scheduling, distributed systems and cloud computing.

PUBLIC INTEREST STATEMENT

Outsourcing is a common practice widely used as a cost-effective strategy in the business world. It is very often clients take help from some external vendor to accomplish their work. The reason for outsourcing could be many, a client want to access some specialized service or someone offering a service more economical than producing in-house. With the advent of cloud computing a variety of client looking for its services since cloud offers convenient on-demand network access to infrastructure services (CPU, Storage, Network), platform services (programming environment, language, library, services and tool) and software services (application such as web-based email service) in pay-as-per-use manner on a very economical cost. However, despite the tremendous benefit outsourcing data and computation to cloud server generates various privacy and security concerns, which must be handled before using it for real life problems.
1. Introduction

The growing number of smart devices and their increasing desire to execute computationally intensive task makes the outsourcing of computation to the cloud server a promising solution. The outsourcing paradigm enables a resource-constrained client to execute large computation task by offloading their computation load to the massive cloud servers. The availability of cloud servers empowers the clients to execute large computations. Now they are no longer restricted to their limited CPU, storage, and bandwidth (Shiraz, Gani, Khokhar, & Buyya, 2013). Despite the tremendous advantage, this promising paradigm brings many security and privacy concerns such as confidentiality of data (input and output) and the integrity of result, which makes the client reluctant to outsource its computation on the cloud server. In particular, the client data carries sensitive information such as personal health records, financial records, trends of stock, scientific research records, just list a few. Therefore, these types of confidential data needs to be encrypted to maintain the confidentiality and the integrity, before outsourcing to the cloud server. One such approach to address such security concerns is apply some encryption scheme. However, the traditional encryption scheme would not work here. Because the traditional encryption scheme changes the input data into cipher and performing meaningful computation on cipher is very difficult. The second concern is the correctness of the result since the cloud server is not a trusted party. Therefore, it may return a wrong result due to a flaw, bug in the logic or some time intentionally deviates from the algorithmic instructions (in case of malicious cloud). This means there is no guaranty on the integrity of the result. Therefore, an outsourcing algorithm must be proficient in providing privacy to the confidential data (input and output) and verifying the correctness of the result. Further, efficiency is also an important challenge in the process of algorithm design. In outsourcing framework, the client perform some operations such as transformation, verification, and retransformation. Therefore, the execution time needed to perform client operations (transformation, verification, and retransformation) must be substantially lesser than executing the original outsourced problem else the outsourcing has no meaning (Chen, Xiang, & Yang, 2014; Lin & Chen, 2010; Mohassel, 2011).

The particular problem that we address in this paper is matrix multiplication. The matrix multiplication is an elementary operation and useful in many domains such as statistics, finance, oil and gas explorations, machine learning, sensor network, agriculture, predicting rainfall, image encryption, watermarking, telemedicine and others. There are various algorithms available for secure outsourcing of various core problems of linear algebra including matrix multiplication (Atallah & Frikken, 2010; Atallah, Frikken, & Wang, 2012; Fiore & Gennaro, 2012; Mohassel, 2011; Zhang & Blanton, 2014). Many of these algorithms were not initially designed for cloud environment. Therefore, these algorithms do not consider the computational asymmetry of the cloud server and the client. Further, these algorithms were developed using complex cryptographic preambles to encrypt the data-set (input and output), which makes them unsuitable for the computation on the cloud with large data-set. Furthermore, these algorithms does not consider the result verification as an essential requirement. However, for the secure outsourcing in the malicious cloud environment, result verification is a vital part of the algorithm design. These gaps in the present state-of-the-art motivate us to design a secure and efficient outsourcing algorithm for matrix multiplication.

The proposed secure outsourcing algorithm for Matrix Multiplication (MM) on various inputs $x_1, x_2, ..., x_n$ is securely executes the computation on the cloud server, while maintaining the privacy of input/output, correctness, result verification and computational efficiency.

The main contribution of paper includes following:

(a) In this paper, we are introducing orthogonal matrix to protect the privacy of the data-set. The algorithm is capable of multiplying any valid dimension of $(M_1 & M_2)$.

(b) The proposed algorithm required an optimal one round of communication. The client in the proposed algorithm verifies the encrypted result obtained from the cloud server with modest overhead and high probability.
(c) The analytical analysis shows that the algorithm is successfully meeting the challenges of correctness, security, verifiability, and efficiency.

(d) The experimental evaluation validates the proposed algorithm. The result analysis shows that the algorithm is highly efficient and endorses the practical usability of the algorithm. All the experimental results can be reproduced.

The rest of the paper is organized as follows: Section 2 discusses the related work done in the area of secure computation. Section 3 provides the detail discussion on the matrix multiplication outsourcing problem, its mathematical elements, and system model to solve them. Section 4 formulates the proposed matrix multiplication outsourcing algorithm and provide detail discussion on the design of the outsourcing algorithm. Section 5 presents analytical analysis of the proposed outsourcing algorithm on correctness, security, verifiability and efficiency parameter. Section 6 presents the experimental analysis of the proposed outsourcing algorithm. Finally, concluding remark and future direction incorporated in Section 7.

2. Related work
In literature, there are various algorithms available for secure outsourcing of various core problems of linear algebra. Moreover, secure outsourcing is divided into two parts i.e. semi-trusted computing model and untrusted computing model. A detail classification of secure outsourcing algorithms are shown in Figure 1. In the semi-trusted computing model, the cloud server follows the algorithm instructions and produces the correct result, but the cloud server secretly records the accessible data, and attempt to retrieve meaningful information. In the semi-trusted model, the first one is audit based (Belenkiy et al., 2008; Monrose, Wyckoff, & Rubin, 1999; Seshadri et al., 2005), the client or the trusted workers in audit-based approach re-computes some part of computation performed by the untrusted workers. This approach is unfeasible for a computationally weak client because if the client is capable of performing the computation, then there is no need to outsource the computation to the cloud server. This method also required that some workers must be honest or at least non-colluding in nature. The second one is secure-co-processors (Bajikar, 2002; Smith & Weingart, 1999; Yee, 1994), or Trusted Platform Modules (TPMs) method, which required to deploy trusted hardware on the server to provide an isolated execution environment. This model is also impractical, since it is not possible to install TPM on third party cloud server. The last one is secure multiparty computation, the computation is divide among two or more workers without allowing any participated worker to view another individual’s secret data. The resultant of computation is the union of the output of all workers (Dreier & Kerschbaum, 2011; Du, Chen, & Han, 2004; Lindell & Pinkas, 2009; López-Alt, Tromer, & Vaikuntanathan, 2012). This model is not feasible in cloud environment since this model required that the participating workers must have comparable computing capability. However, client transfers the entire computation load to the massive cloud servers in the outsourcing paradigm in cloud environment.

![Figure 1. Taxonomy of secure outsourcing algorithms.](image-url)
In untrusted computing model, the server can deviate from the algorithmic instructions and behave arbitrarily. The solution of this problem applies some encryption techniques to perform secure computation and handle the result verification to verify the correctness of the result. There exists two types of verifiable outsourcing i.e. interactive proof and non-interactive proof. In interactive proof, a weak verifier actively challenges a server (prover). The prover replies a probabilistic proof to the client (verifier) to convince him (verifier) about the truth of the statement that he is unable to compute (Fortnow & Lund, 1993; Goldwasser, Kalai, & Rothblum, 2008; Goldwasser, Micali, & Rackoff, 1989). The second one is the non-interactive proof, where a weak client outsources the computation to a powerful server, the server returns the result along with the proof of verification (Chen, Chang, Hsieh, & Chen, 2014; Chen et al., 2015; Chen, Li, & Ma, 2014; Gennaro, Gentry, & Parno, 2010; Hong, Vaidya, & Lu, 2012; Hu & Tang, 2015; Laud & Pankova, 2015; Lei, Liao, Huang, & Chunqiang, 2013; Lei, Liao, Member, Huang, & Li, 2015; Wang, Ren, Wang, & Urs, 2011; Wang, Ren, Wang, & Wang, 2013; Zhou, Li, & Member, 2016).

Our solution applies non-interactive proof to provide verifiability of computation. Therefore, the attention is more on non-interactive verifiable algorithms. Gentry’s provides a novel Fully Homomorphic Encryption (FHE) scheme (Gentry, 2009). This scheme allows the server to perform arbitrary computation on encrypted input, but the solution suffers from the complexity of FHE that makes it far from practical use for outsourcing (Gennaro et al., 2010; Gentry, 2010; Mohassel, 2011; van Dijk, Gentry, Halevi, & Vaikuntanathan, 2010). Moreover, FHE does not guarantees that the server performs correct computation. Gennaro et al. (2010) has proposed a non-interactive solution for the secure outsourcing of polynomial function. First, they model the polynomial function into Yao’s grappled circuit (Yao, 1982, 1986). Then homomorphically (Gentry, 2009) encrypt the circuit and send to the server for the execution of the polynomial function. The cloud server performs the execution & returns a computationally sound non-interactive proof that can verify in $O(m)$ time. Another elegant solution is proposed by Chung, Kalai, and Vadhan (2010). The client performs pre-processing and create hundreds of problem instances mostly of same type. Then apply homomorphic encryption for privacy on problem instances. The server computes these functions without knowing the actual inputs. Finally, the client verifies the solution of the same type of problem for ensuring their correctness. The main drawback of these two algorithms that they are incurring huge computation load on both client and server due to the complex homomorphic encryption. These algorithms are also require modeling of problem into circuit, which needed to dealing with large amount of parameters. However, one main advantage of these methods that they took constant time for verification.

Furthermore, there are many algorithms available, which address some specific type of problems. Atallah, Pantazopoulos, Rice, and Spafford (2002) have carried out an investigation for many problems such as matrix multiplication, quadrature, a solution of the differential equation, the system of linear equations. Their solution explicitly leaking private information. Besides, the proposed work do not handle the result verification. Further, Benjamin and Atallah (2008) gave an outsourcing algorithm for matrix multiplication. The algorithm developed on the assumption of two non-colluding servers that the servers will not share secret information between them. However, this method is vulnerable to colluding attack. Later on Atallah and Frikken (2010) proposed a secure matrix multiplication outsourcing algorithm based on the theory of secret sharing. This algorithm is an improvement of their previous work in Benjamin and Atallah (2008), regarding single server and computational efficiency. However, this algorithm also suffers from computational overhead due to Shamir’s secret sharing technique. The scalar multiplication complexity expands up to polynomial time. Blanton, Zhang, and Frikken (2013) addressed the large scale computation of biometric computation. The implementation leverages individual structures of distance computation and random sampling. The result verification method can verify the result with modest overhead and high probability. Lei et al. (2013) presented matrix inversion outsourcing algorithm. They uses monomial matrix for the privacy of input/output matrix. The algorithm efficiently outsourcing the matrix inversion to the cloud server, while maintaining the correctness, privacy of the input-output, and verifies the result efficiently. Next, an algorithm addresses linear programming (LP) in Wang, Ren, and Wang (2011). This method...
split the LP solver into two parts; those are the LP solver on cloud and the LP parameters on client-side. The client transforms the input data using a transformation technique and outsources to the LP solver on a cloud server. Then cloud server solves the linear programming problem with the help of LP solver and returns the result. The client verifies the result using the fundamental theorem of duality. Further, another algorithm is presented for LP problem, which reduces the computation and communication time significantly (Hong et al., 2012). There are many algorithms published recently for the “system of linear equation” (Chen et al., 2015; Chen, Xiang, et al., 2014; Wang, Ren, Wang, & Urs, 2011). The algorithm proposed by Wang, Ren, Wang, and Urs (2011) requires \( n \) round of communication between the client and cloud before the client get the convergent solution. The algorithm encrypts the input in \( O(n^2) \) in setup phase and verify the output in \( O(n^2) \). The \( n \) round of communication between the client and the cloud is the main drawback of this method which increases the communication overhead and also add security vulnerability. Next, an algorithm in Chen, Xiang, et al. (2014) identified a new solution for the linear equation, which employs some special linear transformations using diagonal matrices. This algorithm requires one round of communication between the client and the cloud. Further, an algorithm (Chen et al., 2015) utilizes the sparse matrix to propose a new secure outsourcing algorithm of large-scale linear equations in the fully malicious model. The algorithm requires only \( O(1) \) round of communication between the client and cloud server.

3. Details of secure outsourcing algorithm

3.1. Outsourcing algorithm

A secure and verifiable outsourcing algorithm has five steps in the following order: KeyGen, ProbTrans, Compute, Verify, and Retransform.

(a) **KeyGen** \((1^k)\): The algorithm takes input security parameter \( k \) and generates orthogonal matrix, which is used for problem transformation. The key generation step runs for every new problem submission.

(b) **ProbTrans** \((\varphi, k)\): The client encrypts the original problem \( \varphi(M_1, M_2) \) with the key \( k \) and generates the transformed problem \( \varphi_k(M'_1, M'_2) \).

(c) **Compute** \((\varphi_k(M'_1, M'_2))\): The client outsources the transformed problem to the cloud server. Then the cloud server executes the matrix multiplication on the transformed problem and return the result \( R' \) to client.

(d) **Verify** \((R', k)\): Next, the client verifies the encrypted result \( R' \) obtained from the cloud server.

(e) **Retransform** \((R', k)\): The client retransform/decrypts the result \( R' \) (if verification step is successfully passed) to obtain the result \( R \) of matrix multiplication problem.

3.2. System model

The proposed system model for secure outsourcing of matrix multiplication algorithm is shown in Figure 2. The resource-constrained client wants to execute a matrix multiplication problem, but due to lack of computing resources, the client unable to perform such expensive computation. Thus, the client outsources the problem to the massive cloud servers. The client first applies a transformation operation on the input problem \( \varphi(M_1, M_2) \) and produces \( \varphi_k(M'_1, M'_2) \). Now, this transformed problem \( \varphi_k(M'_1, M'_2) \) is outsourced to the cloud server. Further, the cloud server computes the matrix multiplication for the transformed input \( (M'_1, M'_2) \) and produces the result \( R' \). The client receives the result \( R' \) from the cloud server. The client first verifies the encrypted result \( R' \) with the help of secret keys and the transformed input \( (M'_1, M'_2) \). If the result passes the verification test successfully, then only the client re-transforms/decryt the result to obtain the result \( R \) of the problem \( \varphi(M_1, M_2) \) otherwise the algorithm is aborted.
3.3. Threat model

The security threat in outsourcing system model originated from the suspicious behavior of the cloud server. The previous work for the secure outsourcing computation defines three threat model that are “Trusted Model”, “Semi-Trusted Model”, and “Untrusted Model” (Chen et al., 2015; Lei et al., 2013; Lei, Liao, Huang, & Heriniaina, 2014).

3.3.1. Trusted model

The cloud server follows the algorithm instructions correctly and it does not attempt to unauthorized access to the client information. Therefore, no need of encryption and verification.

3.3.2. Semi-trusted model

In this model, the cloud behaves as “honest but curious” or “lazy but honest” or even both. Goldreich, Micali, Goldreich, Micali, and Wigderson (1987) first introduces the honest but curious model. The server follows the algorithm instructions and produces the correct result, but the cloud server tries to secretly record the accessible data and attempt to retrieve meaningful information. However, the proposed algorithm has been design to handle such threat model. The cloud server is not able to extract the original information from the transformed input or the result. The lazy but honest model behaves honestly; it does not record any such information or computation but behave lazily, that the cloud server might not perform on the agreed service level. It might send some invalid result to save computing resources. To, share the remaining resources with other clients to increase the financial gain. The proposed system model for matrix multiplication successfully addresses this security threat. The client can detect the correctness of result with an optimal probability.

3.3.3. Un-trusted model

The third threat model comes from the malicious behavior of cloud server, which is the strongest adversarial model. In this model, the server could be “lazy”, “curious”, and “dishonest”. The cloud server deviates from the algorithm instructions and performs arbitrary. The cloud server may return random indistinguishable result and try to escape and not being detected by the client. The proposed algorithm can detect any such malicious adversary. For this purpose, an effective and efficient result verification technique has been introduced, which verify the correctness of the result.
3.4. Basic idea of matrix multiplication outsourcing

The matrix multiplication operation is represented as:

\[ R = M_1 \ast M_2 \]  \hspace{1cm} (1)

where \( M_1 \) is an \( m \times n \) matrix in \( \mathbb{R}^{m\times n} \), \( M_2 \) is an \( n \times p \) matrix in \( \mathbb{R}^{n\times p} \), and the resultant \( R \) is an \( m \times p \) matrix in \( \mathbb{R}^{m\times p} \). The basic idea of matrix multiplication outsourcing algorithm is as follows: Let the resource-constrained client has a matrix multiplication problem of large input size, but due to lack of computing resources, the client outsources this problem to the cloud server. The MM problem is denoted as \( \varphi = (M_1, M_2) \) and its solution as \( R \). Further, the client applies a transformation operation on input data to preserve the privacy of data. Then, this transformed data is outsourced to the cloud server. Thus, from the viewpoint of the cloud server, this transformed problem is same as any other matrix multiplication problem. However, the cloud server could read this input, but unable to recover the original client information. In this way, the algorithm protects the privacy of the input. Then, the cloud server executes the matrix multiplication on the transformed input and produces the encrypted result \( R' \). Then, the client verifies the integrity of result \( (R') \), once the result passes the verification step, then client retransforms/decrypts and accepts the result.

4. Privacy preserving proposed transformation technique for matrix multiplication

In this section, we present the development of linear transformation method to protect the privacy of client’s data. For a matrix multiplication problem \( \varphi = (M_1, M_2) \), the privacy of \( (M_1, M_2) \) called the input privacy and the privacy of \( (R) \) called the output privacy. The client applies an efficient and secure transformation operation on the data \( (M_1, M_2) \) to transformed MM problem \( \varphi_{t}(M_1', M_2') \), the transformation operation maintains the confidentiality of data (input and output).

(a) The equation \( R = M_1 \ast M_2 \) provides a direction to transform the input \( (M_1, M_2) \) into \( M_1' = (M_1 \ast A) \) and \( M_2' = B \ast M_2 \). When the cloud server computes the matrix multiplication for this transformed input, the result is \( R' = (M_1A \ast BM_2) \). However, the original solution is \( R = M_1 \ast M_2 \). If we try to relate the two outcomes of result \( R \) and \( R' \). The matrices \( A \) and \( B \) should be orthogonal matrix i.e. \( A^T = B \).

\[ AA^T = I \]  \hspace{1cm} (2)

where \( A^T \) is the transpose of matrix \( A \) and \( I \) is identity matrix. Now, \( A \) is an orthogonal matrix, if \( M_1' = (M_1 \ast A) \) and \( M_2' = (A^T \ast M_2) \), then the transformed solution \( R' = (M_1A \ast A'A^T) \). However, an issue has been raise, that this method is unable to provide privacy to the output result. The cloud server could easily read the output result since \( R = R' \).

(b) We further explore the method to find a better solution, which provide privacy to the both input and output. In the pursuit of solution, if we transform the input matrix \( M_1' = (D_1 \ast M_1 \ast A) \) and \( M_2' = (A^T \ast M_2 \ast D_2) \), and putting these transformed values to the Equation (1), the output result \( R' = (D_1 \ast M_1 \ast A) \ast (A^T \ast M_2 \ast D_2) = (D_1M_1AA^T M_2D_2) = (D_1M_1A^T M_2D_2) \), where \( D_1, A, \) and \( D_2 \) matrices are serve as the secret keys, \( D_1 \) & \( D_2 \) are diagonal matrices \( (n \times n, p \times p) \) in \( \mathbb{R}^{m\times m} \) and \( \mathbb{R}^{p\times p} \), while \( A \) is an orthogonal matrix of dimension \( m \times m \) in \( \mathbb{R}^{m\times m} \) respectively. The result \( R' \) is similar to the original solution of the problem and we can easily establish a relation between \( R' \) and \( R \) i.e.

\[ R' = D_1RD_2 \]  \hspace{1cm} (3)

(c) The proposed method discussed in section (b) is analyzed to see whether its efficiency can be further improved. We look into the matrix property for better results. It has found if we restrict the key matrices \( A, D_1 \) & \( D_2 \) to diagonal matrix. The efficiency of the proposed algorithm will be improved. Thus, the client will achieve more performance gain. When the orthogonal matrix \( A \) restrict to be diagonal, the orthogonal property required to be relaxed to some extent, that means the entries of diagonal elements of matrix \( A \) are either \( r \) or -\( r \), where \( r \) is a real number. Thus, the diagonal matrix \( A \) is such that, \( A^T A = r^2 I \). Now apply the new values of key matrices to
transform the matrix multiplication problem. The transformed problem
\( \varphi_k \left(M'_1, M'_2 \right) \) = \((D_1M_1A), (A^TM_2D_2))\), when applied these transformed value to the matrix
multiplication in Equation (1) the relation between the resultant matrix \( R \) and \( R' \) such that,

\[
R' = ((D_1M_1A) \ast (A^TM_2D_2)) \Rightarrow (D_1M_1r^TM_2D_2) \text{, where } r \text{ is a real number,} \\
\Rightarrow r^2(D_1M_1M_2D_2) \Rightarrow r^2D_1RD_2
\]

The output result in both the Equations (3) and (4) are not leaking information to the cloud server. Moreover, the method discussed in section (c) improves the efficiency of the algorithm, but lagging in the security than the method in section (b). Therefore, there is a trade-off between security and efficiency of the proposed algorithm. Based on the requirement of security and efficiency the client either use method (b) or method (c). Further, it is worth noting that at the place of diagonal matrix \( D_1 \) & \( D_2 \), permutation matrix can be utilized. However, the permutation matrix takes more time for computation than the diagonal matrix but provides greater security. Thus, for the application, which requires more security can choose either one of the combinations as per their requirement.

4.1. Details of matrix multiplication algorithm construction

There are five sub-algorithms in the proposed matrix multiplication outsourcing algorithm that are
\textbf{KeyGen} (\( 1^k \)), \textbf{ProbTrans} (\( \varphi, k \)), \textbf{Compute}\( \varphi_k(M'_1, M'_2) \), \textbf{Verify} (\( R', k \)), and \textbf{Retransform} (\( R', k \)). Since the proposed algorithm has developed for malicious cloud environment, the key generation algorithm run each time for a new problem submission. This effective mechanism (new keys for a new problem) diminishes the chance of known-plain-text and chosen-plaintext attack.

4.1.1. \textbf{KeyGen} (\( 1^k \))

The client invokes this algorithm with an input security parameter \( \lambda \), this algorithm first generates an identity matrix. These matrices are masked with the non-zero random numbers \( \alpha \rightarrow \{ \alpha_1, \alpha_2, \ldots, \alpha_n \}, \beta \rightarrow \{ \beta_1, \beta_2, \ldots, \beta_n \} \) which produces two diagonal matrices \( (D_1, D_2) \) and then the orthogonal matrices \( A \). The orthogonal matrix \( A \) has been generated using Gram-Schmidt algorithm (Björck, 1994; Estep & Higham, 2004). In this way the key matrices are generated, which are \( (k = A, D_1, D_2) \).

\begin{algorithm}
\caption{KeyGen}
1. \textbf{Input:} \( \lambda \)
2. The client creates two set of non-zero random numbers \( \alpha \rightarrow \{ \alpha_1, \alpha_2, \ldots, \alpha_n \}, \beta \rightarrow \{ \beta_1, \beta_2, \ldots, \beta_n \} \) where \( \emptyset \neq \alpha \cup \beta \) and security key \( sk \in \mathbb{R} \)
3. \textbf{Output:} \( \alpha, \beta, sk \)
\end{algorithm}

The output \( \alpha, \beta, sk \) become input for KMG algorithm that generates the key matrices \( (A, D_1, D_2) \), where \( A \) is orthogonal matrix and \( (D_1, D_2) \) are diagonal matrices.

\begin{algorithm}
\caption{KMG (Key Matrix Generation)}
1. \textbf{Input} \( I_n \) (Identity Matrix), \( \alpha, \beta, sk \)
2. for \( i = 1 : m \)
   \( D_1 = \alpha_i \ast I_{(i)} \)
end for
3. for \( j = 1 : n \)
   \( A = sk \ast I_{(n)} \)
end for
4. for \( k = 1 : p \)
   \( D_2 = \beta_k \ast I_{(k)} \)
end for
5. \textbf{Output} key Matrix \( A, U_1, U_2 \)
\end{algorithm}
4.1.2. ProbTrans ($\varphi, k$)

For every new input, the sub-algorithm ProbTrans ($\varphi, k$) invokes and transforms the input in order to preserve the privacy. This algorithm transforms the input into $M'_1 = D_1 M_1 A$, $M'_2 = A^T M_2 D_2$. This operation dominates the client side computation cost, but still efficient than computing a general matrix multiplication. The one-time cost of ProbTrans ($\varphi, k$) is $O (mn + np)$, after that the MM outsourcing algorithm runs very efficiently.

<table>
<thead>
<tr>
<th>Algorithm 3. ProbTrans</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input $A, D_1, D_2$, Input Matrix $M_{1\text{input}}, M_{2\text{input}}$</td>
</tr>
<tr>
<td>2. for $i = 1 : m$</td>
</tr>
<tr>
<td>$M'_1 (i, :) = D_1 (i, :) \ast M_1 (i, :)$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>for $i = 1 : n$</td>
</tr>
<tr>
<td>$M'_1 (;, i) = A (i, :) \ast M_1 (;, i)$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>3. for $i = 1 : n$</td>
</tr>
<tr>
<td>$M'_2 (i, :) = A^T (i, :) \ast M_2 (i, :)$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>for $i = 1 : p$</td>
</tr>
<tr>
<td>$M'_2 (;, i) = D_2 (i, :) \ast M_2 (;, :)$</td>
</tr>
<tr>
<td>4. end for</td>
</tr>
<tr>
<td>5. Output encrypted Matrix $M'_1 = D_1 M_1 A$, $M'_2 = A^T M_2 D_2$</td>
</tr>
</tbody>
</table>

4.1.3. Compute$\varphi_i(M'_1, M'_2)$

This sub-algorithm performs the computation on the cloud server. The operation Compute$\varphi_i(M'_1, M'_2)$ computes matrix multiplication as $\varphi_i(M'_1, M'_2) = (D_1 M_1 A) \ast (A^T M_2 D_2)$.

<table>
<thead>
<tr>
<th>Algorithm 4. Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input $\varphi_i(M'_1, M'_2)$</td>
</tr>
<tr>
<td>2. The cloud invokes matrix multiplication algorithm and computes $R'$</td>
</tr>
<tr>
<td>3. $R' = M'_1 \ast M'_2$</td>
</tr>
<tr>
<td>4. Output $R'$</td>
</tr>
</tbody>
</table>

4.1.4. Verify $(R', k)$

The cloud server sends the computed result to client. The client computes $D = (M'_1 \ast (M'_2 \ast x)) - (R' \ast x)$, where $x$ is the column matrix $x \in \{1, 0\}$. This algorithm introduces an extra term $x$ to minimize the complexity of computation. Since the matrix-vector multiplication only cost $O (n^2)$. If the cloud server has correctly executed the Compute$\varphi_i(M'_1, M'_2)$ correctly, the relation $(D = 0)$ should be hold true.

<table>
<thead>
<tr>
<th>Algorithm 5. Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input encrypted Matrices $(M'_1, M'_2, R')$</td>
</tr>
<tr>
<td>2. for $i = 1 : k$</td>
</tr>
<tr>
<td>3. The client generates $x = {0, 1}^n$ vector</td>
</tr>
<tr>
<td>4. $D = R' \ast x - M'_1 \ast M'_2 \ast x$</td>
</tr>
<tr>
<td>5. if $(D = {0, 0, 0, \ldots, 0}^n)$</td>
</tr>
<tr>
<td>\quad return(1)</td>
</tr>
<tr>
<td>\quad else</td>
</tr>
<tr>
<td>\quad return(0)</td>
</tr>
</tbody>
</table>
6. end if
7. end for
8. Output: Client accept the result \( R' \) if it gets (1) else rejects.

4.1.5. Retransform\((R', k)\)
If the matrix multiplication \( \text{Compute}_{\varphi}(M'_1, M'_2) \) passes the verification step successfully then only this step will execute to find the original values of the matrix multiplication \( R = (D_1^{-1} \ast R' \ast D_2^{-1})/sk^2 \) else this step is simply omitted.

Algorithm 6. Retransform
1. Input encrypted Matrices \( (R', D_1, D_2) \)
2. \( D_1 \) and \( D_2 \) are square invertible matrices and only cost linear time for inversion.
3. The client computes \( D_1^{-1} \) \& \( D_2^{-1} \)
4. \( R = (D_1^{-1} \ast R' \ast D_2^{-1})/sk^2 \)
5. Output Result: \( R \)

5. Analysis of proposed matrix multiplication algorithm
The proposed MM outsourcing algorithm is simultaneously meeting all the four challenges of an outsourcing algorithm that are correctness, security, verifiability, and efficiency. The analytical analysis of the algorithm follows the previous work in (Chen, Chang et al., 2014). Further, we present an analytical analysis of proposed algorithm in a malicious cloud environment.

5.1. Correctness analysis
The matrix multiplication outsourcing algorithm will perform correctly, only if the client and the cloud server follow the algorithm instructions properly and produces correct result. Further, we are providing a mathematical explanation, which verifies our claim. Theorem 1: The proposed algorithm is correct if both the client and cloud follows the algorithm instructions properly.

Proof: The client first transforms the input \( (M_1, M_2) \) into \( M'_1 = (D_1M_1A), \) and \( M'_2 = (A^TM_2D_2), \) then the cloud server will compute \( R' \),

(a) Correctness Analysis, when the key matrix \( A \) is orthogonal and \( D_1 \& D_2 \) are diagonal matrices.

From the Equation (3),

\[
R' = D_1RD_2
\]

Then in the procedure of matrix multiplication re-transformation, the client will compute result \( R \) where,

\[
\Rightarrow R = D_1^{-1}D_1M_1M_2D_2D_2^{-1} = M_1 \ast M_2 \tag{5}
\]

(b) Correctness Analysis, when the key matrix \( A \) is a orthogonal diagonal matrix and \( D_1, D_2 \) are diagonal matrices.

From the Equation (4),

\[
R' = r^2D_1RD_2
\]

The client computes the result \( R \) where,
\[ R = r^2 D_1^{-1} D_1 M_1 D_2 M_2^{-1} D_2^{-1} / r^2 \]

Thus, the client will get the correct result in both the cases, which implies the proposed algorithm is correct.

5.2. Security analysis

In the malicious cloud environment, the cloud server behaves as curious cloud. Therefore, the server may record all the client information (input and output) and then tries to retrieve the original information from the recorded information. However, the cloud server never succeeded to recover the original information. Further, a security analysis is presented, which justify our claim. The security analysis follows the previous work on secure outsourcing on cloud computing (Chen, Chang et al., 2014; Wang et al., 2013).

THEOREM 2 In a malicious cloud model the proposed matrix multiplication outsourcing algorithm able to protect the privacy of client’s input \((M_1, M_2)\) and the processed output \(R\).

Proof The proposed algorithm able to provide privacy to the input and output data-set of the matrix multiplication. We have provided mathematical proof to justify out claim.

First, the input privacy

The client first transforms the original input matrix \(\varphi = (M_1, M_2)\) to \(\varphi_1 = (M'_1, M'_2)\) where \(M'_1 = (D_1 M_1 A)\) and \(M'_2 = (A^T D_2 D_1)\). The cloud server has only access to the transformed problem \(\varphi_2 = (M'_1, M'_2)\) and have no information about \(\varphi = (M_1, M_2)\) and the security keys \((A, D_1, D_2)\). The matrices \(M'_1, M'_2\), are not leaking information to the cloud server about the original matrices \((M_1, M_2)\). The matrices \((A, D_1, D_2)\) are in \(\mathbb{R}^m, \mathbb{R}^n, \mathbb{R}^p\) respectively.

(a) Security analysis, when the key matrix \(A\) is orthogonal and \(D_1 & D_2\) are diagonal matrices.

Let each entry in the orthogonal matrix \(A\) is \(l\) bit long integer. The matrix \(A\) is an \(n \times n\) matrix in \(\mathbb{R}^n\) so, there are approximately \(2^{l(n^2)}\) bit of information, that means there are \(2^{2l(n^2)}\) possible choices of key matrix \(A\). Similarly, \(D_1 & D_2\) are diagonal matrices, they have \((m, n, p)\) bit of information. Further, the key matrices \((A, D_1, D_2)\) combined have \(2^{l(n^2)}\) possibilities, which is a large value in term of \(m, n, p\). The estimated time of brute-force attack on the key space to recover the original matrix \((M_1, M_2)\) is \(2^{\frac{lm}{2}(n^2)}\) possibilities, which is a exponential bound quantity in terms of \((m, n, p)\). Further, the client generates a new key for every new problem submission to cloud server. Thus, the cloud server cloud never recovers meaning full information about the input data.

(b) Security analysis, when the key matrix \(A\) is an orthogonal diagonal matrix and \(D_1, D_2\) are diagonal matrices.

The key matrix \(A\) is a diagonal matrix and its entries are set to either \(r\) or \(-r\), where the \(r\) is a random real number of \(l\) bit long. There are \(2^{n_l}\) possibility of \(A\), whereas for diagonal matrix \(D_1 & D_2\) there are \(2^{n_m}\) possibilities. In this case the client has \(2^{n_l+n_m}\) possible choices for the key matrices, which is still an exponential time bound quantity in terms of \((m, n, p)\). Therefore, in this case also the cloud server could not recover any meaningful information.

Second, the output privacy

Similarly, the output result is also protected in the same way as the input data is protected. The resultant matrix is also not leaking any information to the cloud. Thus, a malicious cloud server even if record the input and the output information, but never able to recover the original input and output information. Besides, the client generates a pair of new securities keys \((A, D_1, D_2)\) for every new prob-
lem submission to the cloud server. Thus, our encryption system is similar to one-time-pad encryption system. Therefore, there is also no chance of known-plain-text attack or chosen-plain-text attack.

5.3. Verifiability analysis
In the malicious threat model, the cloud server may deviate from the actual instruction of the matrix multiplication outsourcing algorithm and return a random arbitrary result. Thus, the matrix multiplication outsourcing algorithm must be equipped with the result verification process, which is able to verify the correctness of the result.

**Theorem 3** A wrong/invalid result never passes the result verification step.

**Proof** If the cloud server performs computation correctly, a wrong result $R'$ never passes the verification test.

Therefore,

$$D = (M_1'M_2' - R') = \{0, 0, \ldots , 0\}^T$$  \hspace{1cm} (7)

If the cloud produces a wrong result, it never passes the verification step. Further, we provide proof to justify the claim. If cloud produces a wrong result, then $D \times x \neq (M_1'M_2' \times x - R' \times x)$, then there exist at least a row $D$ which is not equals to zero.

$$Dx = \{d_1, \ldots , d_p\}^T$$  \hspace{1cm} (8)

Let the row $d_i \neq 0$

$$d_i = \sum_{j=1}^{p} D_{i,j}*x_j = D_{i,1}*x_1 + \ldots + D_{i,k}*x_k + \ldots + D_{i,p}*x_p$$  \hspace{1cm} (9)

So, there exist at least one element in this row which is not equal to zero, let $D_{i,k} \neq 0$,

$$d_i = D_{i,k}*x_k + z, \text{ where } z = \sum_{j=1}^{p} D_{i,j}*x_j - D_{i,k}*x_k$$  \hspace{1cm} (10)

Applying total probability theorem,

$$\Pr (d_i = 0) = \Pr ((d_i = 0)| (z = 0)) \Pr [z = 0] + \Pr ((d_i = 0)| (z \neq 0))\Pr [z \neq 0]$$  \hspace{1cm} (11)

$$\Pr [ (d_i = 0)| (z = 0)] = \Pr [x_k = 0] = 1/2$$

$$\Pr [ (d_i = 0)| (z \neq 0)] \leq \Pr [x_k = 1] = 1/2$$, substituting these values to Equation (11).

$$\Pr (d_i = 0) \leq 1/2 \Pr [z = 0] + 1/2\Pr [z \neq 0]$$  \hspace{1cm} (12)

$$\Pr [z = 0] = 1 - \Pr [z \neq 0]$$, substituting this value to Equation (12)

$$\Pr (d_i = 0) \leq 1/2(1 - \Pr [z \neq 0]) + 1/2\Pr [z \neq 0]$$

$$\Pr (d_i = 0) \leq 1/2$$  \hspace{1cm} (13)

The verification process run $k$ times, the value of $k$ is a tradeoff between efficiency versus verifiability.

$$Pr_k \leq \Pr (d_i = 0) \leq 1/2^k$$  \hspace{1cm} (14)

The proof reveals that even if the cloud produces the wrong result, it never passes the verification test.
The probability to pass the erroneous result is a negligible quantity. Thus, the client able to catch the erroneous result produced by cloud server. In this article, we have performed the experiments by taking $k = 20$, while 20 bit is acceptable, because $1/2^{20}$ is itself a big quantity $2^{20} \approx 1$ million, means the verification process fails to detect a wrong result once in million.

### 5.4. Efficiency analysis

The outsourcing algorithm is divided into two parts that are the client-side and the cloud-side computation. The client performs the following sub-operations the Key Generation $\text{KeyGen}(1^k)$, problem transformation $\text{ProbTrans}((\varphi, k))$, Result Verification $\text{Verify}(R', k)$, and the Result re-transformation $\text{Retransform}(R', k)$, while the cloud server computes the computationally expensive matrix multiplication algorithm. Table 1 presents the theoretical performance analysis of MM outsourcing algorithm. Theorem 4: The secure outsourcing algorithm is $O\left(\frac{1}{m} + \frac{1}{n} + \frac{1}{p}\right)$ efficient implementation of matrix multiplication.

### Proof

In order to perform secure outsourcing of matrix multiplication to cloud server, the client needs to perform some matrix multiplication in procedure of $\text{ProbTrans}((\varphi, k))$, $\text{Verify}(R', k)$, and $\text{Retransform}(R', k)$. However, due to special arrangement of key matrices $((A, D_1, D_2))$ that is only one element is present in each row and column. The multiplication at client-side only cost quadratic times, since the cost of addition is omitted. Thus, the multiplication cost for $M_1 = (D_1, M_1, A)$ and $M_2 = (A'M_2D_2)$ only cost $O(mn + np)$. For the result verification the client performs the following operation $D = (R' + x) - M_1 + (M_2 + x)$. The asymptotic complexity of this operation is only $O(mp)$. Once the result $R'$ computed by cloud server passes the result verification algorithm. The client retransform the result $R'$ execution following operation $R = D_1^{-1}R'D_2^{-1} = M_1 + M_2$. The time complexity required to execute this operation is only $O(mp)$. Thus, the asymptotic time complexity of client-side computation is only $O(mn + np + mp)$, while the cloud server takes $O(mnp)$. Thus, the proposed outsourcing algorithm is $O\left(\frac{1}{m} + \frac{1}{n} + \frac{1}{p}\right)$ times efficient than the general matrix multiplication algorithm.

The client got a clear performance gain due to time complexity gap between client and cloud. As the size of $m, n, \text{and} p$ increases the client will got sufficient performance gain. Therefore, this outsourcing algorithm is efficient and a feasible solution from resource constrained client.

### 5.5. Comparison analysis

In this section, we provide a comparison of the proposed algorithm with the published work (Atallah & Frikken, 2010; Benjamin & Atallah, 2008). The algorithm in (Benjamin & Atallah, 2008) is working on the assumption of two non-colluding servers. Thus, if the servers are colluding they can pass secret information between them that makes this approach vulnerable for colluding attack. Besides, this algorithm works in a semi-trusted model, which makes the algorithm ill-suited for the untrusted behavior of cloud. The algorithm in Atallah and Frikken (2010) is an improvement over the previous algorithm in Benjamin and Atallah (2008). It achieves provable security on the Shamir secret sharing scheme. This algorithm is work over finite field $\mathbb{Z}_p$ and suffers from large communication overhead of secret sharing. Therefore, do not meeting the requirements of outsourcing of large computation. A comparative analysis of proposed algorithm with these state-of-the-art algorithms are presented in Table 2.
6. Experimental analysis

The implementation of the proposed algorithm is based on the mathematical and theoretical analysis discussed in the previous sections. The algorithm is implemented using Matlab language version 2014a on a system simulating both the client and server. The system configuration is “CPU Intel® Core™ I3 (CPUs) ~1.8 GHZ 4 GB Ram”. The client and cloud server having the same computing resources, which reflects the actual running time of the algorithm. If the algorithm is executed on two different systems, the algorithm performance would be case specific. Although, in reality, the cloud servers always have more computing resources that will further reduce the execution time of the problem. The evaluation of the proposed algorithm mainly focused on the client-side execution time and cloud-side execution time. In addition, for the proposed algorithm, the execution cost dominates and the communication cost is significantly small. Therefore, we ignore the communication cost in the performance analysis.

To measure the performance of the algorithm, we are using three standard parameters efficiency, performance-gain and the relative extra cost.

\[
\text{Efficiency } \eta = \frac{\text{OT}}{\text{CSPT}} \quad (15)
\]

Ideally, the efficiency of the algorithm should be close to one. If the efficiency is nearby one, it indicates that the execution time of original problem and the encrypted problem is almost same.

### Table 2. Comparison with other MM outsourcing algorithm

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption/ transformation technique</td>
<td>Homomorphic encryption</td>
<td>Shamer secret sharing</td>
<td>Linear transformation</td>
</tr>
<tr>
<td>Matrix dimension</td>
<td>Square matrix</td>
<td>Square matrix</td>
<td>Any compatible matrix</td>
</tr>
<tr>
<td>Range of matrix elements</td>
<td>Infinite field $\mathbb{R}$</td>
<td>Finite field $\mathbb{Z}$</td>
<td>Infinite field $\mathbb{R}$</td>
</tr>
<tr>
<td>Verifiable computation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data privacy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Experimental analysis</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Suitable for cloud environment</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Table 3. Terms and description

<table>
<thead>
<tr>
<th>Notations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>Matrix dimension $(m, n, p)$</td>
</tr>
<tr>
<td>Original time (OT)</td>
<td>Time to execute the matrix multiplication without encryption and outsourcing paradigm (direct method)</td>
</tr>
<tr>
<td>Cloud server processing time (CSPT)</td>
<td>Time to execute transformed/encrypted matrix multiplication</td>
</tr>
<tr>
<td>Client processing time (CPT$_1$)</td>
<td>Time for key generation + Time for problem transformation</td>
</tr>
<tr>
<td>Client processing time (CPT$_2$)</td>
<td>Result verification time + Time for re-transformation</td>
</tr>
<tr>
<td>Total client processing time TCPT</td>
<td>$TCPT = CPT_1 + CPT_2$</td>
</tr>
<tr>
<td>Efficiency</td>
<td>$\eta = \frac{\text{OT}}{\text{CSPT}}$</td>
</tr>
<tr>
<td>Performance gain</td>
<td>$PG = \frac{\text{OT}}{TCPT}$</td>
</tr>
<tr>
<td>Relative extra cost</td>
<td>$REC = \frac{(TCPT + CSPT - OT)}{\text{OT}}$</td>
</tr>
</tbody>
</table>
The second parameter is performance gain for the client “It represents the actual speed-up the client has gained from outsourcing the problem”.

\[ \text{PG} = \frac{\text{OT}}{\text{TCPT}} \]  

(16)

Theoretically, the performance gain of the client should always be greater than one.

**REC:** The relative extra cost is defined as the amount of extra work done by the client and cloud server in outsourcing paradigm as compared to direct method. REC is also known as overhead of the outsourcing algorithm. Ideally, REC should be near to zero, which indicates that the outsourcing paradigm incur minimum burden on the client and the cloud server.

\[ \text{REC} = \frac{(\text{TCPT} + \text{CSPT} - \text{OT})}{\text{OT}} \]  

(17)

Further, the terms use in the experimental analysis presented in Table 3.

We execute the MM outsourcing algorithm multiple times for each problem instance to get a stable system performance. The experimental performance analysis of the proposed algorithm is shown in Table 4 and Figure 3. We can see in Figure 3 (a) and Table 4, that the efficiency parameter remains close to one, which means the outsourcing paradigm adds minimum overhead on the cloud server for executing an encrypted problem. Further, it is shown in Table 4 that the REC parameter stays 0.36896 for (500 * 600 * 700) problem size and 0.03984 for (5000 * 6000 * 7000) problem size, a decrease of 89.20% in REC parameter as the problem size increases from (500*600*700) to (5000 * 6000 * 7000). The values of REC indicate that the extra work done by the cloud and client reduces as the problem size increases, for larger size problem the overhead reduces to great extent. Figure 3 (b) shows that the REC is monotonically decreasing. Further, Figure 3(c) demonstrates the comparison of execution the times of matrix multiplication using direct method and proposed
Table 4. Performance analysis of MM outsourcing algorithm

<table>
<thead>
<tr>
<th>Matrix dimension</th>
<th>Original time</th>
<th>Cloud time</th>
<th>Client-side computation</th>
<th>Performance gain</th>
<th>Efficiency</th>
<th>Relative extra cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OT (Sec)</td>
<td>CSPT (s)</td>
<td>CPT₁ (s)</td>
<td>CPT₂ (s)</td>
<td>TCPT (sec)</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>n</td>
<td>p</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>m</td>
</tr>
<tr>
<td>500</td>
<td>600</td>
<td>700</td>
<td>0.02868</td>
<td>0.02352</td>
<td>0.00959</td>
<td>0.00615</td>
</tr>
<tr>
<td>1,000</td>
<td>1,200</td>
<td>1,400</td>
<td>0.16505</td>
<td>0.169973</td>
<td>0.03603</td>
<td>0.02277</td>
</tr>
<tr>
<td>1,500</td>
<td>1,800</td>
<td>2,100</td>
<td>0.53048</td>
<td>0.540548</td>
<td>0.08177</td>
<td>0.04815</td>
</tr>
<tr>
<td>2,000</td>
<td>2,400</td>
<td>2,800</td>
<td>1.28088</td>
<td>1.27701</td>
<td>0.14366</td>
<td>0.08036</td>
</tr>
<tr>
<td>2,500</td>
<td>3,000</td>
<td>3,500</td>
<td>2.28362</td>
<td>2.27847</td>
<td>0.23005</td>
<td>0.13167</td>
</tr>
<tr>
<td>3,000</td>
<td>3,600</td>
<td>4,200</td>
<td>4.21932</td>
<td>4.16783</td>
<td>0.33502</td>
<td>0.18341</td>
</tr>
<tr>
<td>3,500</td>
<td>4,200</td>
<td>4,900</td>
<td>6.69722</td>
<td>6.59376</td>
<td>0.47098</td>
<td>0.2596</td>
</tr>
<tr>
<td>4,000</td>
<td>4,800</td>
<td>5,600</td>
<td>9.62245</td>
<td>9.55587</td>
<td>0.61148</td>
<td>0.33243</td>
</tr>
<tr>
<td>4,500</td>
<td>5,400</td>
<td>6,300</td>
<td>14.24576</td>
<td>13.87941</td>
<td>0.80005</td>
<td>0.44301</td>
</tr>
<tr>
<td>5,000</td>
<td>6,000</td>
<td>7,000</td>
<td>18.15748</td>
<td>17.34169</td>
<td>1.00034</td>
<td>0.53889</td>
</tr>
</tbody>
</table>
outsourcing algorithm. The time to execute a problem of size $(5000 \times 6000 \times 7000)$ is $18.15$ s, but when the computation is outsourced to the cloud server using the proposed algorithm, it takes only $1.539$ s. This happen because the client in our algorithm work only $O(mn + np + mp)$ for matrix multiplication rather the computation burden of $O(mnp)$.

Therefore, a computational saving of $O\left(\frac{1}{m} + \frac{1}{n} + \frac{1}{p}\right)$ for the client and as the size of problem $(m, n, p)$ increases the client got substantial computational saving. Furthermore, the performance-gain parameter is in double-digit for the larger size problem and able to attain more than $11.79$ times, which is a very motivating factor to use MM outsourcing algorithm in real-world scenarios. Note that the graphical performance analysis presented in Figure 3 is started from matrix of size $2000$. Below that size refer to the Table 4. Finally, the theoretical and experimental analysis of matrix multiplication outsourcing algorithm proves, that the resource-constrained client can execute large matrix multiplication by offloading their computation load to massive cloud servers using matrix multiplication outsourcing algorithm. The proposed work is very efficient and able to maintain the privacy of data under the malicious cloud environment. Due to the rising demand of outsourcing, the proposed solution would be very helpful for the research community to carry out further research work.

It is worth noting that the experimental performance depends on problem dimension, the underlying execution platform, and algorithm for solving matrix multiplication (Lei et al., 2013). If the cloud exploits other faster matrix operation algorithms, then client speedup will decrease to some extent. However, as long as the size of the input goes sufficiently large, the client get a significant performance-gain due to the apparent computation gap between the client-side computation and server-side computation.

7. Conclusion
In this paper, we formulated and implemented a new outsourcing algorithm for the matrix multiplication problem. The proposed algorithm is entirely different from the previous algorithms, which were based on the complex cryptographic preambles, that makes them ill-suited for the computation outsourcing with large input size to the cloud computing. Remarkably, the proposed computation outsourcing algorithm has not uses any such complex cryptography techniques and developed by utilizing linear transformation method using the concepts of linear algebra. The algorithm is able to meet the challenges of input/output privacy, correctness, verifiability, and efficiency, which is well demonstrated in the analytical analysis of the algorithm. The proposed algorithm needed a onetime setup cost of $O(mn + np + mp)$ on the client system, while executes the matrix multiplication with $O(mnp)$ on the cloud server. That makes the proposed algorithm $O\left(\frac{1}{m} + \frac{1}{n} + \frac{1}{p}\right)$ times efficient implementation of matrix multiplication. In future, it would be remarkable to see novel algorithms, which reduces the overhead cost incur by the verification process and develop newer algorithms both with constant verification time and with greater efficiency. However, finding newer computationally expensive mathematical, scientific and engineering problem and designing outsourcing solution for them, would always be noticeable.


