Secure and verifiable outsourcing protocol for non-negative matrix factorisation

Zhenhua Liu and Bin Li*

School of Mathematics and Statistics, Xidian University, Xi’an, 710071, Shaanxi, China
Email: zhualiu@hotmail.com
Email: zanilyqaz@163.com
*Corresponding author

Qi Han

School of Telecommunications Engineering, Xidian University, Xi’an, 710071, Shaanxi, China
Email: hanqiwildginger@gmail.com

Abstract: With the rapid development of cloud computing services, resource-constrained clients can outsource their expensive computation tasks, such as scientific computations, to untrusted cloud servers. Furthermore, it is essential for these clients to protect their sensitive data and verify the validity of the returned computation results. In this paper, we focus on outsourcing protocol of non-negative matrix factorisation, which is an expensive computation task and has been widely applied to image processing, face recognition, text analysis, and so on. The permutation technique is employed to transform the original problem into a new one in our proposed protocol so as to protect the privacy, and the matrix 1-norm technique is utilised to verify the result returned from the cloud server in order to reduce the verification cost. Based on these two techniques, we construct a secure and verifiable outsourcing protocol for non-negative matrix factorisation. Moreover, the theoretical analysis and the experimental results show that our proposed protocol brings great computation savings for resource-constrained clients and fulfills the goals of correctness, security, verifiability and high efficiency.

Keywords: cloud computing; permutation; outsourcing; cloud security; verifiability; matrix factorisation.


Biographical notes: Zhenhua Liu received his PhD in Cryptography from Xidian University in 2009. Currently, he is working in Xidian University. His current research interests include public key cryptography, cloud security and secure outsourcing computations.

Bin Li is a postgraduate student of Xidian University. His research interests include outsourcing computation and privacy protection.

Qi Han received his BS in Mathematics in 2012 from Xidian University. He is currently working toward his PhD in Cryptography in Xidian University. Besides, he is a visiting PhD student in University of Waterloo, Canada. His research interests include applied cryptography, network security.

This paper is a revised and expanded version of a paper entitled ‘Secure and verifiable outsourcing protocol for non-negative matrix factorization’ presented at The First International Workshop on Cyber Security (CS-2015), Xi’an, China, 16–18 October 2015.

1 Introduction

Non-negative matrix factorisation (NMF), a relatively novel paradigm for dimensionality reduction, was first introduced by Paatero and Tapper (1994) as the concept of positive matrix factorisation. In scientific research and engineering applications, it can help the researchers and engineers obtain the low rank matrix approximation of primitive data matrix, and save the computing resources. Thus, it has been shown
to be an effective tool in many fields of scientific and engineering research. Moreover, non-negative matrix factorisation is widely used in the signal processing (Cichocki et al., 2006), computer vision (Hoyer, 2002), and data mining (Xu et al., 2003). However, non-negative matrix factorisation is also a computation-intensive task if the data is large. For example, a typical double precision 50,000 × 50,000 matrix resulted from signal processing would occupy up to 20 G bytes storage space. But, ordinary client’s computing devices (e.g., mobile devices, laptop) can not satisfy this storage requirement. An economical solution for the resource-constrained client is to outsource this task to the cloud server who has powerful computational resources and capacities.

The outsourcing service we mentioned above is one of the most significant services of cloud computing (Fortis et al., 2015; Giuseppe et al., 2015). In fact, outsourcing of computations may be necessary when a resource-constrained client needs to execute a task, but does not have the enough computational resources to perform it. By outsourcing the workloads into the cloud server, the resource-constrained client can enjoy the unlimited computing resources in a pay-per-use manner and the computational power of cloud customers is no longer limited by their resource-constraint devices (Pad, Phone, etc.). In addition, outsourcing services can help enterprises and individuals save their cost on computing devices and human capital. Therefore, there is no doubt that outsourcing computation paradigm significantly benefits to various companies and individuals.

Despite the multitudinous benefits, outsourcing computational problems to the commercial public cloud also inevitably results in some new security challenges (Brunette and Mogull, 2009).

1 The outsourced computational workloads always contain some sensitive and important information that should be hided from the cloud server. For instance, financial records, health information, research data, etc. Thus, the client’s input/output data privacy security is the first challenge. In other words, the cloud server should learn nothing about the client’s input/output data. So the client should encrypt their sensitive data and sends the encrypted data to the cloud server.

2 The second challenge is efficiency. On the one hand, an essential factor is that the amount of local work performed by the client must be cheaper than performing the original computational problem by itself. Otherwise, there is no need for the client to resort to the cloud server. On the other hand, the amount of work performed by the cloud server should be as close as possible to that needed to compute the original problem by the client itself.

3 The third challenge is the verification of the computation results returned from the cloud server. There are many possible reasons for the cloud server to return an invalid result. Such as, the cloud server might contain a software bug that will fail on a constant number of invocations. Moreover, the cloud server might decrease the amount of the computation because of some financial incentives and return a computationally indistinguishable (false) result to the client. An outsourcing protocol must be designed in such a way that the client can detect whether the result is correct or not (Zhao et al., 2015). Consequently, we are motivated to design a secure, efficient and verifiable outsourcing protocol for non-negative matrix factorisation.

1.1 Related works

Over the past few years, there were plenty of works on the secure outsourcing computation. Most of these works focus on outsourcing protocols of the following two fields: scientific computations and cryptographic computations. Our system framework is inherited from these works. Atallah et al. (2001) proposed a framework for secure outsourcing of scientific computations such as matrix multiplication, matrix inversion and differential equations. Benjamin and Atallah (2008) and Atallah and Frikken (2010) proposed two secure matrix multiplication outsourcing algorithms, respectively. The former builds upon the assumptions of two non-colluding servers, while the latter achieves provable security using Shamir’s secret sharing technique and suffers from large amount of computation overhead. Then, Lei et al. (2013) proposed a new secure and verifiable algorithm for outsourcing large matrix multiplication by using permutation technique and Monte Carlo technique. Xiang et al. (2015) proposed three new and secure outsourcing schemes of matrix calculation (e.g., matrix multiplication, the inverse of a matrix and the determinant of a matrix) by using one untrusted cloud server. To construct an outsourcing computation protocol for solving large-scale systems of linear equation \( Ax = b \), Wang et al. (2011) proposed a protocol by using iteration from Jacobi method and additive homomorphic encryption with semantic security, but the solutions that the client obtained are approximate. Chen et al. (2015) proposed a new algorithm for large-scale systems of linear equations. Chen’s new algorithm is superior to Wang’s algorithm in both efficiency and checkability. Furthermore, Wang et al. (2011) proposed a secure outsourcing mechanism for linear programming. However, the computation cost for the client is dominated by matrix-matrix multiplication operations with complexity of \( O(n^\rho) \) for some \( 2 < \rho \leq 3 \). To reduce the computational complexity of the client, Nie et al. (2014) proposed a secure and practical outsourcing mechanism for linear programming computations based on sparse matrix technique, and the local computation complexity for the client is \( O(n^2) \).

At the same time, many researchers have devoted considerable attention to the problem of how to securely outsource expensive cryptographic computations. Matsumoto et al. (1990) proposed the first outsourcing protocol for exponentiation computations. Hohenberger and
Lysyanskaya (2005) firstly proposed a secure algorithm for outsourcing modular exponentiations based on the pre-computation and server-aided computation. With these two techniques, modular exponentiations can be efficiently computed by the user with $O(\log^2)$ multiplications. Chen et al. (2012) proposed a new and efficient algorithm for secure outsourcing of modular exponentiation, which is superior to Hohenberger et al.’s protocol in both efficiency and check-ability. Chen et al. (2015) also proposed an efficient algorithm for secure outsourcing of bilinear pairings in the two untrusted program model.

Non-negative matrix factorisation was first introduced by Paatero and Tapper. It is regarded as a powerful matrix decomposition technique that approximates a non-negative matrix by the product of two low-rank non-negative matrix factors. There are some traditional optimisation algorithms for the problem of non-negative matrix factorisation, such as: Lee et al. (2001), Zdunek and Cichocki (2006), Lin (2007), Kim et al. (2007) and Bonettini (2011). Generally, traditional non-negative matrix factorisation algorithms solve this problem by transforming it into an optimisation problem. If the client utilises these traditional algorithms to deal with large scale of non-negative matrix factorisation problem, it will take a long time and need a large storage space. However, the resource-constrained clients can not execute these traditional algorithms. Therefore, outsourcing this computational task to the cloud server, which has powerful computational and storage resources, is an economical choice for the resource-constrained clients.

### 1.2 Our contributions

This paper presents a secure, efficient and verifiable outsourcing protocol for non-negative matrix factorisation based on a series of efficient and secure transformation techniques. Our contributions are summarised as follows:

1. We motivate and formulate the outsourcing computation of non-negative matrix factorisation for the first time and present an outsourcing protocol
2. We utilise the permutation technique to protect the privacy of input/output data, and the matrix 1-norm technique to improve the efficiency of verification
3. The efficiency analysis further demonstrates that our protocol can efficiently reduce the overheads on the client and achieve great savings for the client.

**Organisation:** The rest of this paper is organised as follows. Some essential preliminaries are presented in Section 2. The detailed blinding techniques and the proposed protocol of non-negative matrix factorisation are given in Section 3. Related security and efficiency analysis is made of in Section 4. Finally, some conclusions are drawn in Section 5.

## 2 Preliminaries

### 2.1 Security definition

We regard an outsourcing computation architecture in which a computationally weak client wants to outsource the computation of a function $F$ on input $x$ to a cloud server. The cloud server returns the result of the function evaluation, i.e., $y = F(x)$, as well as a proof that the computation of $F$ was carried out correctly on the given value $x$. Then, the client verifies the correctness of the result returned from the cloud server. Gennaro et al. (2010) presented the formal definition for securely outsourcing computation in CRYPTO 2010.

**Definition 1 (Secure outsourcing computation):** A secure and verifiable outsourcing computation scheme should consist of the following four algorithms.

1. **KeyGen**($F, \lambda \rightarrow (PK, SK)$): taking the security a parameter $\lambda$ as input, the randomised key generation algorithm generates a public key $PK$ that encodes the target function $F$, which is used by the cloud server to compute $F$. It also computes a matching secret key $SK$, which should be kept private by the client.
2. **ProGen**$_{SK}(x) \rightarrow (\sigma, \tau)$: the problem generation algorithm uses the secret key $SK$ to encrypt the input $x$ as a public value $\sigma$, which is given to the cloud server to compute with, and a secret value $\tau$, which is kept private by the client.
3. **Compute**$_{PK}(\sigma) \rightarrow (\sigma_f)$: given the client’s public key $PK$ and the encoded input $\sigma$, the cloud server computes an encoded version $\sigma_f$ of the function’s output $y = F(x)$.
4. **verify**$_{SK}(\tau, \sigma_f) \rightarrow y \cup \perp$: on input the secret key $SK$ and the secret decoding $\tau$, the verification algorithm converts the cloud server’s encoded output $\sigma_f$ into the output $y$ of the function, or outputs $\perp$, indicating that $\sigma_f$ is not valid.

### 2.2 System model

Generally, an outsourcing computation architecture involves two different entities, as illustrated in Figure 1: the client, who has limited computational resources, wants to outsource the original non-negative matrix factorisation problem to a cloud server; the cloud server, who has large computational resources and storage capabilities. The client has a large-scale non-negative matrix factorisation problem $\phi = (V)$ to be solved. However, he can not carry out such an expensive computation locally because of lacking of computing sources, such as: memory, storage. Then, the client resorts to the cloud server for solving this problem.

More significantly, to achieve input/output privacy, the
client uses a secret key \( SK \) to transform the original problem \( \phi = (V) \) into a new one, written as \( \phi' = (V') \). Then, the client sends this new problem \( \phi' = (V') \) to the cloud server. On receiving \( \phi' = (V') \), the cloud server solves this problem and returns the solution for \( \phi' = (V') \) to the client. At the same time, the cloud server also sends back a proof \( \tau \) to the client, which can prove the correctness of the returned result.

**Figure 1**: Secure and verifiable NMF outsourcing system model (see online version for colours)

On obtaining the result returned from the cloud server, the client decrypts the returned result using the secret key \( SK \). Meanwhile, the client should be capable of checking whether the result is correct or not: if yes, accept it; if no, just rejects it.

### 2.3 Threat model

The security threats faced by this outsourcing system framework mainly come from the behaviour of the cloud server. Generally speaking, there are three levels of model in outsourcing computation scheme.

The first is the ‘lazy-but-honest’ model which was firstly introduced by Golle and Mironov (2001). The rational cloud server honestly follows the protocol and provides a solution for the clients, while it will try to minimise the amount of work that it needs to perform so as to retrieve the payment.

The second is the ‘honest-but-curious’ model, i.e., the semi-honest model that was firstly introduced by Goldreich et al. (1987). In the semi-honest model, the cloud server follows the protocol with the exception that it keeps all its computations records, and it attempts to use these records to learn some sensitive information.

In the ‘fully malicious’ model which is defined in Chen et al. (2015), the cloud server is lazy, curious and dishonest. It may return a random computationally indistinguishable result to the client to save its computation resources, while hoping not to be detected by the client. Consequently, the client must be able to verify the correctness of the result returned from the cloud server in the full malicious model.

In this paper, we assume that the single cloud server is malicious.

### 2.4 Design goals

A secure and verifiable outsourcing protocol for non-negative matrix factorisation should satisfy the following properties:

1. **Correctness**: if both the client and the cloud are honest and follow the protocol, the result that the client obtained from the cloud server should be a correct solution to the original problem.

2. **Soundness**: no false result from a cheating cloud server can pass the verification with non-negligible probability.

3. **Privacy**: no sensitive information of the client’s data should be obtained by the cloud server during performing the non-negative matrix factorisation computation.

4. **Efficiency**: the local computation done by the client should be substantially less than the computation of the original problem \( \phi = (V) \) on his own. In addition the amount of computation on computing \( \phi' = (V') \) should be as close as possible to that on computing the original problem \( \phi = (V) \).

5. **Verifiability**: the client can verify the correctness and incorrectness of the results received from the cloud server with non-negligible probability.

### 2.5 Mathematical backgrounds

**Definition 2** (non-negative matrix factorisation): Non-negative matrix factorisation is a recently developed method for reducing dimension of non-negative matrix. The definition of non-negative matrix factorisation can be found in Liu and Cheng (2013). It can be expressed as follows:

Given a non-negative matrix \( V \in \mathbb{R}^{m \times n} \) and a positive integer \( r \ll \min\{m, n\} \), non-negative matrix factorisation aims to find two non-negative matrix \( W \in \mathbb{R}^{m \times r} \) and \( H \in \mathbb{R}^{r \times n} \) such that \( V \approx W \cdot H \). Without loss of generality, it is assumed that \( m \ll n \).

**Definition 3** (Permutation function): Permutation function is well studied in group theory and combinatorics. In mathematics, the notion of permutation relates to the act of rearranging, or permuting. The permutation function \( \pi(x) \) can be expressed as follows:

\[
\pi(x) = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix}
\]

**Definition 4** (Kronecker delta function): The Kronecker delta function \( \delta_{xy} \) in Atallah et al. (2001) is defined as

\[
\delta_{xy} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases}
\]
In this paper, let \( I \) be an \( n \times n \) identity matrix. For a matrix \( X \), let \( x(i, j), x_{ij} \) or \( x_{ij} \) denote the entry in \( i^{th} \) row and \( j^{th} \) column of matrix \( X \).

### 3 Protocol construction

A secure and verifiable outsourcing protocol for non-negative matrix factorisation consists of five steps:

1. The step of key generation Key Generation.
2. The step of transformation of original problem non-negative matrix factorisation transformation.
3. The step of solving the transformed problem \( \phi' \) the transformed problem solving.
4. The step of the decryption of the returned result non-negative matrix factorisation decryption.
5. The step of result verification non-negative matrix factorisation verification.

#### 3.1 Key generation

In the first step, the protocol generates a secret key \( SK \) for the client by doing the followings:

a. The client takes as input a security parameter, which species key space \( K_\alpha \) and \( K_\beta \).

b. The client picks two sets of random positive numbers:

\[
\{\alpha_1, \alpha_2, \cdots, \alpha_m\} \leftarrow K_\alpha, \\
\{\beta_1, \beta_2, \cdots, \beta_m, \cdots, \beta_n\} \leftarrow K_\beta,
\]

where \( 0 \notin K_\alpha \cup K_\beta \).

c. The client invokes Algorithm 1 to generate two random permutations \( \pi_1 \) of the integers \((1, 2, \ldots, m)\) and \( \pi_2 \) of the integers \((1, 2, \ldots, m, \ldots, n)\).

Algorithm 1 is due to Durstenfeld (1964). It is usually called the modern version of Fisher-Yates shuffle which is a classic algorithm to generate a random permutation. In fact, the asymptotic time complexity of Algorithm 1 has been optimal. According to Black (2005), the time complexity of Algorithm 1 is \( O(n) \), while the time complexity of Fisher-Yates shuffle algorithm is \( O(n^2) \). Therefore, we utilise Algorithm 1 to generate these two random permutations.

d. The client takes

\[
SK = \{(\alpha_1, \alpha_2, \cdots, \alpha_m), \{\beta_1, \beta_2, \cdots, \beta_m, \cdots, \beta_n\}, \pi_1, \pi_2\}
\]

as secret key.

#### 3.2 Non-negative matrix factorisation transformation

To achieve the input privacy, the client should transform the original problem \( \phi = (V) \) into a new one \( \phi' = (V') \). The client proceeds with the following three sub-steps:

a. The client takes the original matrix \( V \) and the secret key \( SK \) as input.

b. The client generates an \( m \times m \) matrix \( P_1 \) and an \( n \times n \) matrix \( P_2 \), where

\[
P_1(i, j) = \alpha_i \delta_{\pi_1(i), j}, \quad P_2(i, j) = \beta_j \delta_{\pi_2(i), j}.
\]

According to the following Lemma 1, these two matrices are invertible.

c. The client computes \( V' = P_1 \cdot V \cdot P_2^{-1} \), and then sends it to the cloud server.

**Lemma 1:** In the step of non-negative matrix factorisation transformation, both \( P_1 \) and \( P_2 \) are invertible matrices. Furthermore, their inverse \( P_1^{-1} \) and \( P_2^{-1} \) are expressed as follows:

\[
P_1^{-1}(i, j) = (\alpha_i)^{-1} \delta_{\pi_1^{-1}(i), j}, \quad \text{for } i = 1, 2, \ldots, m.
\]

\[
P_2^{-1}(i, j) = (\beta_j)^{-1} \delta_{\pi_2^{-1}(i), j}, \quad \text{for } j = 1, 2, \ldots, m, \cdots, n.
\]

**Proof:** Since that \( 0 \notin K_1 \cup K_2 \), the determinants of \( P_1 \) and \( P_2 \) satisfies \( \det(P_1) \neq 0 \) and \( \det(P_2) \neq 0 \).

Therefore, \( P_1 \) and \( P_2 \) are invertible. Moreover, the transformed matrix \( V' \) can be simply expressed according to the following theorem.

**Theorem 2:** In the step of non-negative matrix factorisation transformation, the entry in \( i^{th} \) row and \( j^{th} \) column of matrix \( V' = P_1 \cdot V \cdot P_2^{-1} \) can be denoted as:

\[
V'(i, j) = \left(\frac{\alpha_i}{\beta_j}\right) V(\pi_1(i), \pi_2(j)).
\]

**Proof:** Let

\[
V = \begin{pmatrix}
v_{1,1} & \cdots & v_{1,n} \\
\vdots & \ddots & \vdots \\
v_{m,1} & \cdots & v_{m,n}
\end{pmatrix}
\]

Since that \( P_1(i, j) = \alpha_i \delta_{\pi_1(i), j}, \) we can obtain the following formula

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Random permutation generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set ( \pi = I_n ) (Identical permutation)</td>
</tr>
<tr>
<td>2</td>
<td>for ( i = n ) down to 1</td>
</tr>
<tr>
<td>3</td>
<td>Set ( j ) to be a random integer with ( 1 \leq j \leq i ).</td>
</tr>
<tr>
<td>4</td>
<td>Swap ( \pi[j] ) and ( \pi[i] ).</td>
</tr>
<tr>
<td>5</td>
<td>end for</td>
</tr>
</tbody>
</table>
Due to \( P_2^{-1}(i, j) = (\beta_j)^{-1} \delta\tau_z^2(i, j) \) for \( i = 1, 2, \ldots, m, \ldots, n \), it holds that

\[
P_1 \cdot V \cdot P_2^{-1} = \begin{pmatrix} \alpha_1^\prime V_{\tau_z^1}(i, 1) & \ldots & \alpha_n^\prime V_{\tau_z^1}(i, n) \\ \vdots & \ddots & \vdots \\ \alpha_m^\prime V_{\tau_z^1}(m, 1) & \ldots & \alpha_n^\prime V_{\tau_z^1}(m, n) \end{pmatrix}
\]

Therefore, \( V' = P_1 \cdot V \cdot P_2^{-1} \) can be written as:

\[
V'(i, j) = \left( \frac{\alpha_1}{\beta_j} \right) V_{\tau_z^1}(i, \tau_z^2(j)).
\]

Consequently, according to Theorem 2, the clients need not to perform matrix multiplications \( P_1 = P_1 \cdot V \), \( P_2 = P_1 \cdot V \), and can use the equation (1) efficiently compute \( V' \), which take time \( O(mn) \).

### 3.3 The transformed problem solving

On receiving the matrix \( V' \), the cloud server then utilises traditional non-negative matrix factorisation optimisation algorithm to get an approximate factorisation \( V' = W' \cdot H' \), where \( W' = (w_{i,j})_{m \times n} \gg 0 \), \( V' = (v_{i,j})_{m \times n} \gg 0 \). Next, the cloud server sends the result to the client. At the same time, the cloud server sends back a proof \( \tau \) to the client and the proof \( \tau \) can prove that the returned result is indeed correct.

### 3.4 Non-negative matrix factorisation decryption

The step of decryption comprises the following two sub-steps:

a) Firstly, the client receives the result \( W' \) and \( H' \) returned from the cloud server and checks the non-negativity of every entry in \( W' \) and \( H' \). In other words, every entry in \( W' \) and \( H' \) should be non-negative.

b) Secondly, if \( W' = (w_{i,j})_{m \times n} \gg 0 \) and \( H' = (h_{i,j})_{m \times n} \gg 0 \) holds, then the client uses the secret key \( SK \) to compute \( W = P_1^{-1} \cdot W' \) and \( H = H' \cdot P_2 \), which are reviewed as the original non-negative matrix's approximate factorisations. According to the following Theorem 3, the client can efficiently compute \( W \) and \( H \) (via time \( O(mn) \) and \( O(nr) \), respectively).

**Theorem 3:** In the step of non-negative matrix factorisation decryption, \( W = P_1^{-1} \cdot W' \) and \( H = H' \cdot P_2 \) are the original non-negative matrix \( V' \)'s approximate factorisations. Furthermore, they are denoted as

\[
W(i, j) = \left( \frac{1}{\alpha_{\tau_z^1(i,j)}} \right) w_{\tau_z^1(i,j)}, \quad H(i, j) = \left( \frac{1}{\beta_{\tau_z^2(i,j)}} \right) h_{\tau_z^2(i,j)}.\]

**Proof:** Firstly, according to the transformation and factorisation steps, we have

\[
V' = P_1 \cdot V \cdot P_2^{-1} = W' \cdot H', \quad P_1 \cdot W \cdot H \cdot P_2^{-1}
\]

Thus, we can obtain \( W = P_1^{-1} \cdot W' \) and \( H = H' \cdot P_2 \). It is to say that the original non-negative matrix \( V \) has approximate factorisations \( W \) and \( H \), i.e., \( V = W \cdot H \).

Then, according to Lemma 1,

\[
P_1^{-1}(i, j) = (\alpha_j)^{-1} \delta\tau_z^2(i, j), \quad P_2(i, j) = \beta \delta\tau_z^2(i, j).
\]

And similarly to Theorem 2, we can deduce the equality

\[
W(i, j) = \left( \frac{1}{\alpha_{\tau_z^1(i,j)}} \right) w_{\tau_z^1(i,j)}, \quad H(i, j) = \left( \frac{1}{\beta_{\tau_z^2(i,j)}} \right) h_{\tau_z^2(i,j)}.
\]

### 3.5 Non-negative matrix factorisation verification

Generally, it is hard to prove the optimality of the solutions of non-negative matrix factorisation, and thus its verification is not an easy problem. However, we can employ the special structure of this problem to perform the step of verification. We need to define a cost function to quantify the quality of the approximation factorisation. Lee et al. (2001) pointed out that the square of the Euclidean distance between the original matrix \( V \) and approximate factorisation \( W \cdot H \) is regarded as one of the most useful measures to evaluate the quality of the approximation factorisation. Let \( A = W \cdot H \), the square of the Euclidean distance between \( V \) and \( A \) is defined as:

\[
\| V - A \|^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (v_{i,j} - a_{i,j})^2,
\]

where \( V = (v_{i,j})_{m \times n} \) and \( A = (a_{i,j})_{m \times n} \). However, the computational cost to compute the square of the Euclidean distance between \( V \) and \( A \) is very expensive when big data is considered. To reduce the cost, we can use the 1-norm of matrix to verify the result returned from the cloud server. According to Cohen and Peng (2014), the 1-norm matrix technique has been used to successfully handle and shrink
big data matrix. They also proved that their algorithm
condenses matrices under the 1-norm as well as it does
under the 2-norm. And thus in the verification step, we also
utilise the 1-norm of matrix $A$, which is defined as:
\[
\|A\|_1 = \max \left\{ \sum_{i=1}^{m} |a_{i,1}|, \sum_{i=1}^{m} |a_{i,2}|, \ldots, \sum_{i=1}^{m} |a_{i,n}| \right\},
\]
where $|a_{ij}|$ is the absolute value of the entry $a_{ij}$ in matrix $A$.
The client receives the result returned from the cloud server
and decrypt the result by using her/his secret key $SK$. And
then, the client can compute the residual error $\varepsilon$, which is
defined in Kim and Park (2008) and where
\[
\varepsilon = \frac{\|V - A\|}{\|V\|}.
\]
In this paper, $\varepsilon$ is reviewed as an error-threshold and should
be small. It is possible for the client to have a prior experience
about the error range. If the error is in the reasonable range, the client should accept the result.
Otherwise, the client will reject.

In this step of verification, the client needs to compute $A = W \cdot H$, which takes time $O(mnr)$. The computation overhead on the client to compute $A$ is a little expensive when big data is considered. To further reduce the computation overhead to compute $A$, we complete the following works.

1. In the step of the transformed problem solving, the cloud server should also return the product of matrices $W'$ and $H'$ back to the client. Let $A' = W' \cdot H'$.
2. To verify the correctness of $A'$ returned from the malicious cloud server, the client chooses the $i$th row of $W'$ and $j$th column of $H'$ randomly, where $1 \leq i \leq m$, $1 \leq j \leq n$. Then, the client computes
\[
c = \sum_{i=1}^{m} (w'_{ij} \cdot h'_{ij}).
\]
If the element in the $i$th row and $j$th column of matrix $A'$ is equal to $c$, then the client considers that $A'$ is indeed correct and the cloud server does not cheat. In fact, parameter $i$ has $m$ options and parameter $j$ has $n$ options. The probability that any incorrect $A'$ generated by cloud server can be passed verification is
\[
\frac{1}{mn}.
\]
It means that the incorrect $A'$ generated by cloud server can be detected with non-negligible probability.

3. Next, the client can obtain $A$ by just computing $P_1^{-1} \cdot A' \cdot P_2$ and it takes time $O(2mn)$ which is much smaller $O(mnr)$ when a big $r$ is considered. We remark that there could be some other better verification methods for non-negative matrix factorisation outsourcing, and it is remained to be studied in our future work.

4 Security and efficiency analysis

4.1 Security analysis

We follow up Wang et al. (2011a, 2011b), Nie et al. (2014) and Chen et al. (2015) which are previous works in the area of outsourcing computation for the security analysis. First of all, we remind that our protocol is similar to one-time pad encryption scheme. Therefore, there is no plaintext-known attack or chosen-plaintext attack. Then, we prove that our protocol can protect the input/output privacy of non-negative matrix factorisation problem.

**Theorem 4:** Our proposed protocol can protect the input/output data privacy.

**Proof:** Firstly, we prove the privacy for input $V$ of non-negative matrix factorisation problem. The original data $V$ is blinded by the following two steps:

1. The position of each entry in the original matrix $V$ is randomly rearranged under two random permutations, i.e.,
\[
T(i, j) = V(\pi_1(i), \pi_2(j)).
\]
2. Furthermore, each entry in matrix $T$ is blinded by multiplying a factor, i.e.,
\[
V'(i, j) = (\alpha \cdot \beta) \cdot V(\pi_1(i), \pi_2(j)) = (\alpha \cdot \beta) \cdot T(i, j).
\]

In the Step 1, the key space consists of all random permutations of $\pi_1$, $\pi_2$ and there are $(m!n!)$ cases of permutations. Each case occurs with probability $\frac{1}{m!n!}$. In addition, it means that even if the malicious cloud server gets the correct $T(i, j)$, the expected time of brute-force attack on the key space to recover the original matrix $V$ is
\[
\frac{m!n!}{2}.
\]

From the Step 2, each entry in $T$ is further blinded. It implies that even if the malicious cloud server has the correct matrix $T(i, j)$ and these two random permutations $\pi_1$, $\pi_2$, the expected time of brute-force attack on the key space to guess $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ and $(\beta_1, \beta_2, \ldots, \beta_n)$ is
\[
\frac{[K_m]_n^2}{2}.
\]
Although, the brute-force attack is one of the most basic methods for cryptanalysis, it will be obstructed by a large key space $K_m$ and $K_n$. Moreover, the step of key generation will be run every time for a new outsourced non-negative matrix factorisation instance. Therefore, the cloud server cannot recover $V$ from $V'$ without the secret key $SK$. 
And then, the privacy of the output $W$ and $H$ can be protected in the non-negative matrix factorisation outsourcing. According to Theorem 3, it is evident that the output data privacy be protected in the same way as the input data privacy. We omitted the detailed analysis.

4.2 Efficiency analysis

We summarise the time complexity of the client side in our proposed protocol and one iteration round of some representative traditional optimisation algorithms in Table 1, where $m$, $n$ is the dimension of matrix $V \in R^{m \times n}$, $r$ is the given positive integer in the above non-negative matrix factorisation problem. In addition, according to Guan et al. (2012), $K$ in Lin (2007), Kim et al. (2007) and Bonettini (2011) is the iteration number of the optimal gradient method algorithm, and the variable $t$ in Kim et al. (2007) and Bonettini (2011) is the inner iteration number of line search procedure.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee et al. (2001)</td>
<td>$O(mnr + mn^2 + nr^2)$</td>
</tr>
<tr>
<td>Zdunek et al. (2006)</td>
<td>$O(mnr + m^2r^3 + n^2r^3)$</td>
</tr>
<tr>
<td>Lin (2007)</td>
<td>$O(mnr) + K \times O(mnr + mn^2 + nr^2)$</td>
</tr>
<tr>
<td>Kim et al. (2007)</td>
<td>$O(mnr) + K \times O(mnr + mn^2 + nr^2)$</td>
</tr>
<tr>
<td>Bonettini (2011)</td>
<td>$O((m+n)r + mn + m + n + r)$</td>
</tr>
<tr>
<td>Ours</td>
<td>$O((m+n)r + mn + m + n + r)$</td>
</tr>
</tbody>
</table>

From Table 1, it is evident that the time complexity of one iteration round of traditional non-negative matrix factorisation optimisation algorithm is no cheaper than $O((m+n)r + mn + m + n + r)$. Moreover, Han et al. (2009) has pointed out that these traditional optimisation algorithms always need many iteration rounds to solve non-negative matrix factorisation problem. Hence, the resource-constrained client in our protocol can obtain great computing savings by outsourcing this problem to the cloud server.

4.3 Experimental result

In this section, we provide a simple experimental valuation of the proposed outsourcing algorithm. We implement our mechanism using MATLAB language with a version of R2013a. The process is conducted on a workstation equipped with Intel(R) Core(TM) 3.30-GHz CPU and 16-GB RAM. We compare the efficiency of our proposed outsourcing protocol with the protocol of Lee et al. (2001), which is a representative multiplicative iterative algorithm for non-negative matrix factorisation. The test benchmark for randomly generated non-negative matrix factorisation only focuses on the problems where $m$ ranges from 30 to 7,290, $n$ ranges from 60 to 14,580.

The comparison of two protocols is depicted in Figure 2. From Figure 2, we can make a conclusion that our proposed outsourcing protocol can bring great savings to the resource-constrained clients. In addition, the experimental results is deserved to be improved and we leave it as our future work.

5 Conclusions

We have proposed a secure and efficient outsourcing protocol for non-negative matrix factorisation based on permutation technique and matrix 1-norm technique. The proposed protocol brings the client great computational savings from secure non-negative matrix factorisation outsourcing, and fulfills the correctness and input/output privacy. However, there are still some interesting and significant future works to investigate, such as to establish formal security framework for non-negative matrix factorisation outsourcing protocol, to establish a better verification method for non-negative matrix factorisation outsourcing protocol, and to design protocols for new scientific computation and engineering problems.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China under Grants No. 61472470, 61100229 and the National Natural Science Foundation of Shaanxi Province under Grants No. 2014JM2-6091 and 2015JQ1007.

References


