# Harnessing the Cloud for Secure and Efficient Outsourcing of Non-negative Matrix Factorization

Shiran Pan *†‡, Fangyu Zheng *†, Wen-Tao Zhu †, Qiongxiao Wang *†✉
*State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China
†Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
‡School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{panshiran, zhengfangyu}@iie.ac.cn, {wtzhu, qxwang}@is.ac.cn

***Abstract*—Non-negative matrix factorization (NMF) is the task of factoring a non-negative matrix into the product of two (smaller) non-negative matrices. As an important data analysis technique, NMF finds applications in many areas of science and engineering. Real-world NMF tasks often involve large-scale matrices and incur expensive computation costs. Consequently, we consider the cloud computing paradigm, where a resource-constrained client outsources the NMF task to a powerful but untrustworthy cloud. In many applications, the input of and the solution to the NMF task usually contain the client's private information, which necessitates privacy-preserving outsourcing.**

**In this paper, we first reveal the privacy breaches of a recent NMF outsourcing scheme. Then, for enhanced privacy protection, we employ certain matrix transformations and present a new construction, which can solve the NMF task in a masked yet verifiable manner. Theoretical analysis shows that our proposal is cheating-resistant, recoverable, and privacy-preserving. Experiments on synthetic and real-world data demonstrate that our proposal not only brings significant cost savings to a client but also accelerates the task solving.**

## I. Introduction

In linear algebra, the non-negative matrix factorization (NMF) of a matrix $\boldsymbol{V}$ is to factor it into two matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ with the property that all three matrices have no negative elements. In applications (e.g., document clustering [1], audio signal processing [2], recommender systems [3], etc.) where non-negativity is inherent to the data being considered, NMF plays the role of a fundamental technique for data analysis and helps the clients gain insight into certain properties of the data.

Nowadays, many real-world applications involve the NMF tasks with large-scale input matrices. Solving such tasks often incurs prohibitive computation costs. For example, a modern laptop computer may spend as long as 2 minutes to factor a $6,000 \times 3,000$ matrix. Consequently, in this paper we are interested in connecting the NMF task with cloud computing, where a resource-constrained client turns to a powerful cloud for assistance. In this scenario, the majority of the workload is shifted to the cloud, which aids the client to factor large-scale non-negative matrices. By such outsourcing, the client can be freed from hardware/software constraints and enjoy the nearly unlimited computing resources of the cloud in a pay-per-use manner [4].

Despite tremendous benefits of the cloud computing paradigm, an attending issue with the combination of NMF and cloud computing is that the cloud might be untrustworthy, which arouses some privacy concerns [5]. First, the cloud may provide inaccurate or even invalid computation results for saving its computing resources, hoping not to be detected by the client [6], [7]. Second, the data processed (e.g., the non-negative matrix to be factorized) and generated (e.g., the factorization results) during the computation in the cloud often contains some sensitive information (e.g., image features, textual characteristics, etc.), which may leak the client's privacy and be exploited by the cloud for various purposes [6], [8]. Therefore, it is of significant importance for the client to deploy appropriate mechanisms to verify the results by the cloud and to protect sensitive information. In this work, we address the problem of secure outsourcing of large-scale NMF tasks.

In the literature, there have been many research efforts on secure outsourcing of scientific calculations related to matrix transformations. Atallah et al. [9] presented a framework for securely outsourcing matrix multiplication and matrix inversion. Wang et al. [10] presented efficient mechanisms for secure outsourcing of linear programming computations. Recently, a series of schemes have been proposed for outsourcing large-scale systems of linear equations [6], [8], [11], [12], [13]. Though the existent work is extensive, little is dedicated to the NMF outsourcing. The research effort most related to ours is the outsourcing scheme proposed in [7]; however, we observe that there are some privacy breaches in this scheme (we will be more specific in Section IV). For better privacy assurance, in this work we design a lightweight encryption mechanism to protect the data privacy and accordingly construct a secure and efficient outsourcing scheme for NMF. Our technical contributions can be summarized as follows:

i) We formally define the problem of secure outsourcing of non-negative matrix factorization (NMF). Specifically, we propose a framework for privacy-preserving outsourcing, formulate the basic components of the outsourcing scheme, and identify the design goals. We also reveal two privacy flaws of a recent solution [7] with two concrete examples.

ii) We present an elaborate construction for secure and efficient outsourcing of NMF. Thorough theoretical analysis shows that our proposal is cheating-resistant, recoverable, and privacy-preserving.

iii) We implement our proposal and evaluate it on both synthetic and real-world data. Extensive experimental

978-1-5386-4586-4/18/$31.00 ©2018 IEEE

results obtained from the prototype demonstrate that our construction can benefit a client with significantly reduced computation cost as well as accelerated task solving.

This paper is organized as follows. Mathematical preliminaries are presented in Section II. The problem statement regarding solving NMF tasks with the aid of the cloud is in Section III. Section IV presents the security analysis on the state of the art. Section V describes our proposal. Theoretical analysis is conducted in Section VI, while experimental evaluation is in Section VII. Section VIII concludes the paper.

## II. Preliminaries

### A. Non-negative Matrix Factorization (NMF)

For a non-negative matrix $\boldsymbol{V} \in \mathbb{R}^{m \times n}$, the NMF of $\boldsymbol{V}$ is to find two non-negative factor matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ satisfying

$$\boldsymbol{V} = \boldsymbol{W}\boldsymbol{H},$$

where $\boldsymbol{W} \in \mathbb{R}^{m \times p}$, $\boldsymbol{H} \in \mathbb{R}^{p \times n}$, and $p$ is the factorization parameter and smaller than both $m$ and $n$. Since such task is not exactly solvable in general, the factor matrices are commonly approximated numerically. There are many approximate NMF algorithms and the one proposed in [14], [15] is widely used, which reduces the NMF task to the following optimization problem:

$$\min_{\boldsymbol{W},\boldsymbol{H}} \|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_{\mathrm{F}}^2 \text{ subject to } \boldsymbol{W} \geq 0, \boldsymbol{H} \geq 0,$$

where $\boldsymbol{W}(\boldsymbol{H}) \geq 0$ means all elements in $\boldsymbol{W}(\boldsymbol{H})$ are non-negative, and $\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|\boldsymbol{A}(i,j)|^2}$ is the Frobenius norm of the matrix $\boldsymbol{A} = (\boldsymbol{A}(i,j)) \in \mathbb{R}^{m \times n}$. In this algorithm [14], [15], the elements in $\boldsymbol{W}$ and $\boldsymbol{H}$ are initialized with non-negative random numbers, and then updated in an iterative manner following certain rules until local minimum is achieved (the details of which are beyond the scope of this paper). In practice, the iterative results $(\boldsymbol{W}, \boldsymbol{H})$ satisfying

$$\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_{\mathrm{F}}^2 \leq \epsilon$$

are considered as acceptable factorization results, where $\epsilon$ is a pre-determined error bound. According to this algorithm, the acceptable $(\boldsymbol{W}, \boldsymbol{H})$ is not unique.

The computation complexity of this algorithm for solving an NMF task is approximately $\mathcal{O}(lmnp)$, where $l$ represents the number of iterations. In real-world applications, $l$ needs to be sufficiently large so as to ensure the convergence. When handling large-scale matrices (i.e., $m, n \gg 1$), the NMF task often incurs prohibitive computation costs.

### B. Clustering Property of NMF

The matrix multiplication $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{H}$ can be implemented as computing the column vectors of $\boldsymbol{V}$ as linear combinations of those of $\boldsymbol{W}$, using coefficients supplied by columns of $\boldsymbol{H}$. That is, each column of $\boldsymbol{V}$ can be computed as:

$$\boldsymbol{V}_i = \boldsymbol{W}\boldsymbol{H}_i, \tag{1}$$

where $\boldsymbol{V}_i$ and $\boldsymbol{H}_i$ are the $i$-th column of $\boldsymbol{V}$ and $\boldsymbol{H}$, respectively. Eq. (1) actually reveals the inherent *clustering property* of NMF [16]. Specifically, the clustering of the columns $\{\boldsymbol{V}_1, \cdots, \boldsymbol{V}_n\}$ of input matrix $\boldsymbol{V}$ can be derived from the clustering of the columns $\{\boldsymbol{H}_1, \cdots, \boldsymbol{H}_n\}$ of the factor matrix $\boldsymbol{H}$. It is worthy noting that if one is able to approximately represent $\boldsymbol{V}$ with significantly smaller matrices (i.e., $\boldsymbol{W}$ and $\boldsymbol{H}$), then it can cluster the columns of $\boldsymbol{V}$ more easily (as the dimension of each $\boldsymbol{H}_i$ is significantly smaller than that of $\boldsymbol{V}_i$). The clustering property of NMF is very important and is the basis of many applications.

Here is an example based on a text-mining application. Given 200 documents indexed by 5,000 words, we can represent the text data using a non-negative matrix $\boldsymbol{V}$ with 5,000 rows and 200 columns, where words are in rows and documents are in columns. It follows that the column vector $\boldsymbol{V}_i$ of $\boldsymbol{V}$ represents the $i$-th document. Assume we invoke the NMF algorithm to find 10 latent features (i.e., $p = 10$) in order to generate $\boldsymbol{W}$ with 5,000 rows and 10 columns (named features matrix) and $\boldsymbol{H}$ with 10 rows and 200 columns (named coefficients matrix); the product of $\boldsymbol{W}$ and $\boldsymbol{H}$ has the same dimensions as $\boldsymbol{V}$. Following the treatment of matrix multiplication as in eq. (1), each column of $\boldsymbol{V}$ can be viewed as a linear combination of the 10 column vectors in $\boldsymbol{W}$ with coefficients supplied by $\boldsymbol{H}$. Therefore, we can easily cluster the documents with respect to the 10 features by classifying the column vectors of $\boldsymbol{H}$, which is obviously faster than directing clustering $\{\boldsymbol{V}_i\}$.

### C. Random Permutation Function

A basic mathematical building block for constructing our NMF outsourcing scheme is the random permutation function. A random permutation of a set $\mathcal{S} = \{x_1, \cdots, x_n\}$ is defined as a bijection from $\mathcal{S}$ to itself, denoted as $\pi : \mathcal{S} \to \mathcal{S}$. In Cauchy's two-rows notation [17], the elements of $\mathcal{S}$ are in the first row, and the corresponding image elements are in the second row. The random permutation function can be expressed as:

$$\pi = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix},$$

where $\pi(x_i) = y_i$, $i = 1, \cdots, n$. When $y_i = \pi(x_i) = x_i$, the permutation is called identical permutation. The inverse function of $\pi$ is denoted as $\pi^{-1}$. Alg. 1 illustrates the generation of random permutation.

**Algorithm 1** Generation of random permutation

1: Set $\pi$ to be the identical permutation of $\mathcal{S} = \{x_1, \cdots, x_n\}$.
2: **for** $i = n : 2$ **do**
3: Select a random integer $j$ where $1 \leq j \leq i$;
4: Swap $\pi(x_i)$ and $\pi(x_j)$.
5: **end for**

## III. Problem Formulation

### A. System Model

Denote an NMF task as $\mathcal{T}(\boldsymbol{V})$, which takes as input a non-negative matrix $\boldsymbol{V} \in \mathbb{R}^{m \times n}$. The solution to $\mathcal{T}$ is non-negative matrices $(\boldsymbol{W} \in \mathbb{R}^{m \times p}, \boldsymbol{H} \in \mathbb{R}^{p \times n})$ sat.

$$\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_{\mathrm{F}}^2 \leq \epsilon,$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm (cf. Section II-A) and $\epsilon$ is a pre-determined error bound. In this paper, we consider

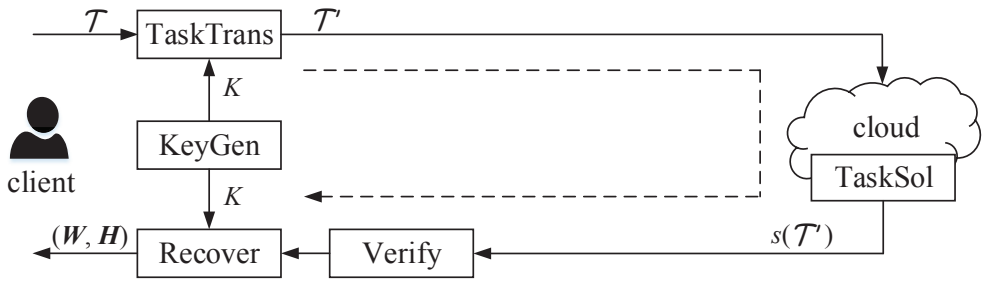



Fig. 1. Architecture of secure outsourcing of non-negative matrix factorization (NMF), which involves two participants: the client and the cloud. The client first generates a secret key $K$ and uses it to transform the original NMF task $\mathcal{T}$ into a masked and outsourced computing task $\mathcal{T}'$. Then, it receives and verifies the intermediate result $s(\mathcal{T}')$ returned by the cloud. Last, the client restores $(\boldsymbol{W}, \boldsymbol{H})$, which is expected to be a solution to $\mathcal{T}$.

an outsourcing framework illustrated in Fig. 1, in which a resource-constrained client securely solves $\mathcal{T}(\boldsymbol{V})$ with the aid of a powerful but untrustworthy cloud. The solid lines indicate certain data flows while the dashed line indicates the workflow.

In this architecture, an NMF outsourcing scheme comprises 5 algorithms (**KeyGen**, **TaskTrans**, **TaskSol**, **Verify**, **Recover**), where only **TaskSol** is conducted by the cloud while others are initiated by the client. First, the client invokes **KeyGen** to generate its secret key $K$. Then it invokes **TaskTrans** to transform a given NMF task $\mathcal{T}(\boldsymbol{V})$ into a masked form $\mathcal{T}'$, which is outsourced to the cloud. Subsequently, the cloud executes **TaskSol** and returns the client the solution to $\mathcal{T}'$. This intermediate result is then checked by the client with **Verify**. If the result passes the verification, the client invokes **Recover** to derive $(\boldsymbol{W}, \boldsymbol{H})$, which is expected to be the solution to the original task $\mathcal{T}$. To prevent the cloud from acquiring sensitive information associated with $\mathcal{T}$, the client employs the secret key $K$ for transforming the task and recovering the final solution.

### B. Threat Model

Following [6], [7], [11], [12], we consider the threat model where the cloud plays the role of a deceptive and curious adversary, which is known as the "fully malicious" model. Specifically, the cloud may return an invalid computation result (for saving its computing resources) while hoping not to be detected by the client. In addition, the cloud may also try to infer, based on the information it receives, the private information associated with the original NMF task. This information includes the input matrix, the factorization results, and the clustering property of the original NMF task (which reveals the clustering of the columns of the input matrix, cf. Section II-B).

### C. Formalized Components

1) $\mathbf{KeyGen}(\lambda) \mapsto K$: Given the security parameter $\lambda$, the client executes this probabilistic key generation algorithm to yield its secret key $K$.
2) $\mathbf{TaskTrans}(\mathcal{T}, K) \mapsto \mathcal{T}'$: The client executes this probabilistic task transformation algorithm to convert with $K$ the original task $\mathcal{T}$ to a masked form $\mathcal{T}'$, which is outsourced to the cloud.
3) $\mathbf{TaskSol}(\mathcal{T}') \mapsto s(\mathcal{T}')$: The cloud invokes this probabilistic task solving algorithm to compute an intermediate result $s(\mathcal{T}')$, which is supposed to be a solution to the masked task $\mathcal{T}'$.
4) $\mathbf{Verify}(\mathcal{T}', s(\mathcal{T}')) \mapsto \texttt{true/false}$: The client invokes this deterministic verification algorithm to check whether $s(\mathcal{T}')$ is a valid solution to $\mathcal{T}'$. If so, the client outputs `true`; otherwise, it outputs `false` and terminates.
5) $\mathbf{Recover}(s(\mathcal{T}'), K) \mapsto (\boldsymbol{W}, \boldsymbol{H})$: The client executes this deterministic recovery algorithm to restore $(\boldsymbol{W}, \boldsymbol{H})$ from $s(\mathcal{T}')$ with $K$. The final $(\boldsymbol{W}, \boldsymbol{H})$ is expected to be a solution to the original task $\mathcal{T}$.

### D. Design Goals

We identify three design goals for NMF outsourcing schemes: cheating resistance, recoverability, and privacy.

**Definition 1** (**Cheating resistance**). *An NMF outsourcing scheme is said to be cheating-resistant if for any $s(\mathcal{T}')$ that is returned by the cloud but not a valid solution to $\mathcal{T}'$, there exists a function $negl(\cdot)$ negligible in the security parameter $\lambda$ satisfying $\Pr(\mathbf{Verify}(\mathcal{T}', s(\mathcal{T}')) \mapsto \texttt{true}) \leq negl(\lambda)$.*

**Definition 2** (**Recoverability**). *An NMF outsourcing scheme is said to be recoverable if for any $s(\mathcal{T}')$ that is returned by a cloud following the designed algorithms, then the $(\boldsymbol{W}, \boldsymbol{H})$ output by* **Recover** *satisfies $\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_{\mathrm{F}}^2 \leq \epsilon$.*

**Definition 3** (**Privacy**). *An NMF outsourcing scheme is said to be privacy-preserving if the cloud cannot infer any of the input matrix, the factorization results, and the clustering property of the original NMF task.*

## IV. Security Analysis on the State of the Art

In this section, we analyze a recent scheme on secure NMF outsourcing [7] from the state of the art by revealing two concrete design flaws.

### A. Review of Duan et al.'s Scheme

Recently, Duan et al. proposed an NMF outsourcing scheme [7] and claimed that a "fully malicious" adversary cannot derive any sensitive information associated with the original NMF task.

In Duan et al.'s scheme, the client first generates as its secret key two permutation matrices $\boldsymbol{P}_1 \in \mathbb{R}^{m \times m}$ and $\boldsymbol{P}_2 \in \mathbb{R}^{n \times n}$, where both $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are binary matrices

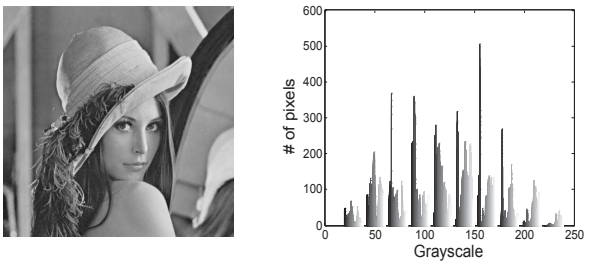
that have exactly one entry of 1 in each row/column and 0's elsewhere. Then, client encrypts the input matrix $\boldsymbol{V}$ to yield $\boldsymbol{V}' = \boldsymbol{P}_1\boldsymbol{A}\boldsymbol{P}_2$, which is submitted to the cloud. According to the property of the permutation matrix [18, §2.3], the matrix multiplication $\boldsymbol{P}_1\boldsymbol{V}$ results in permuting the rows of $\boldsymbol{V}$ while the matrix multiplication $(\boldsymbol{P}_1\boldsymbol{V})\boldsymbol{P}_2$ further permutates the columns of $\boldsymbol{P}_1\boldsymbol{V}$.

Upon receiving $\boldsymbol{V}'$, the cloud is asked to conduct the NMF factorization of $\boldsymbol{V}'$ and return the factorization results $(\boldsymbol{W}', \boldsymbol{H}')$ to the client, which finally restore the factorization of $\boldsymbol{V}$ as $(\boldsymbol{W} = \boldsymbol{P}_1^{-1}\boldsymbol{W}', \boldsymbol{H} = \boldsymbol{H}'\boldsymbol{P}_2^{-1})$. We omit details irrelevant to our security analysis.

### *B. Two Design Flaws in Duan et al.'s Scheme*

**Flaw 1: Leakage of the elements in the input matrix:** In Duan et al.'s scheme [7], the random matrices $(\boldsymbol{P}_1, \boldsymbol{P}_2)$ employed by the client for encrypting the input matrix $\boldsymbol{V}$ are simple permutation matrices. By left-multiplying and right-multiplying $\boldsymbol{V}$ with $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, respectively, the client can only rearrange the position of every element in $\boldsymbol{V}$. Therefore, the cloud knows the values of all elements in $\boldsymbol{V}$ and can further obtain their numerical distribution. In many real-world applications, such statistical distribution reveals some private information associated with the input matrix. For example, for an image matrix $\boldsymbol{V}$ representing a grayscale image, each element of $\boldsymbol{V}$ represents the gray value of each pixel in the image; the numerical distribution of the elements in $\boldsymbol{V}$ is actually the grayscale distribution of the original image, which reveals certain characteristics of the image and is the basis of various applications such as image classification/clustering [19], [20] and image similarity measuring [21].

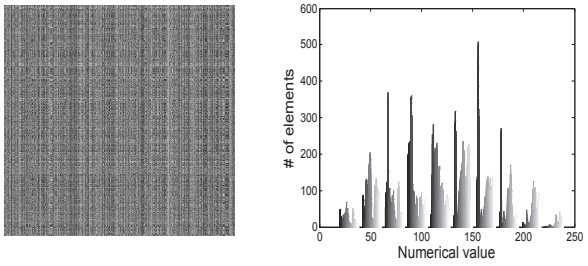
Next, we present a concrete example to demonstrate the above design flaw. For a grayscale image illustrated in Fig. 2(a), its grayscale distribution is shown in Fig. 2(b). Taking as input the image matrix $\boldsymbol{V}$ corresponding to Fig. 2(a), Duan et al.'s scheme generates a masked image matrix $\boldsymbol{V}'$, which is sent to the cloud for factorization. Fig. 3(a) illustrates the obfuscated image corresponding to $\boldsymbol{V}'$, while the numerical distribution of the elements in $\boldsymbol{V}'$ is shown in Fig. 3(b). As can be seen, although the obfuscated image looks "quite different" from the original image, the numerical distribution of elements in $\boldsymbol{V}'$ is exactly the same to the grayscale distribution of the original image, which can be exploited by the cloud for image analysis.




(a) (b)

Fig. 2. (a) Original grayscale image; (b) The grayscale distribution of the image.

**Flaw 2: Leakage of the clustering property of NMF:** As shown in Section II-B, NMF exhibits an inherent clustering property. That is, for a non-negative matrix $\boldsymbol{V}$, one can easily find out the hidden features shared by the column vectors of $\boldsymbol{V}$ from its NMF results, and further use them to cluster these columns vectors.




(a) (b)

Fig. 3. (a) Masked form of Fig. 2(a) that is generated with Duan et al.'s algorithm [7]; (b) The numerical distribution of elements in the masked matrix.

In Duan et al.'s scheme [7], the client restores the NMF factorization of the input matrix $\boldsymbol{V}$ as $(\boldsymbol{W} = \boldsymbol{P}_1^{-1}\boldsymbol{W}', \boldsymbol{H} = \boldsymbol{H}'\boldsymbol{P}_2^{-1})$, where $\boldsymbol{P}_1, \boldsymbol{P}_2$ are permutation matrices and $(\boldsymbol{W}', \boldsymbol{H}')$ form the NMF factorization of the masked matrix $\boldsymbol{V}'$. Following the property of permutation matrices [18, §2.3], $\boldsymbol{P}_2^{-1}$ is still a permutation matrix. Thus, the matrix multiplication $\boldsymbol{H}'\boldsymbol{P}_2^{-1}$ only results in permuting the columns of $\boldsymbol{H}'$ without changing the value of any element in $\boldsymbol{H}'$. Therefore, even though the cloud does not exactly know $\boldsymbol{H}$, it can infer based on $\boldsymbol{H}'$ some information about the clustering of the columns of $\boldsymbol{V}$. As shown in Section II-B, such clustering information may cause privacy leakage.

We next present an example based on a text-mining application following Section II-B and demonstrate the above design flaw. Again assume we have 200 documents indexed by 5,000 words. Then, we can represent these documents with a non-negative matrix $\boldsymbol{V}$ with 5,000 rows and 200 columns, where words are in rows and documents are in columns. In Duan et al.'s scheme [7], the client uses two permutation matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ to generate the masked matrix $\boldsymbol{V}' = \boldsymbol{P}_1\boldsymbol{V}\boldsymbol{P}_2$. Then, the client sends $\boldsymbol{V}'$ to the cloud and receives the factorization results $(\boldsymbol{W}', \boldsymbol{H}')$. Since recovering $\boldsymbol{H} = \boldsymbol{H}'\boldsymbol{P}_2^{-1}$ only results in permuting the columns of $\boldsymbol{H}'$, by clustering the columns of $\boldsymbol{H}'$, the cloud exactly knows how many types the 200 documents can be classified into and the number of documents in each type. This reveals the client's private information.

## V. Proposed Scheme

In this section, we propose to overcome the weaknesses of the solution [7] observed from the state of the art, and develop a new scheme for secure NMF outsourcing. Our proposal is built on the formal definitions presented in Section III-C. It mainly comprises 5 algorithms (**KeyGen**, **TaskTrans**, **TaskSolve**, **Verify**, **Recover**), and is designed to solve an NMF task $\mathcal{T}(\boldsymbol{V})$ and simultaneously meet the design goals proposed in Section III-D with low computation costs. Next, we present the detailed construction of our proposal.

### A. $\mathbf{KeyGen}(\lambda) \mapsto K=(\boldsymbol{M},\boldsymbol{N})$

Given the security parameter, the client invokes Alg. 2 to generate two random invertible sparse matrices $\boldsymbol{M} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{N} \in \mathbb{R}^{n\times n}$, where each row and each column in $\boldsymbol{M}$ (or $\boldsymbol{N}$) has only one non-zero element.

**Algorithm 2** Generation of random invertible sparse matrices

**Input:** Security parameter $\lambda$

**Output:** Invertible sparse matrices $\boldsymbol{M} \in \mathbb{R}^{m\times m}$,$\boldsymbol{N} \in \mathbb{R}^{n\times n}$

1: Invoke Alg. 1 to generate two random permutations $\pi_1$ and $\pi_2$ of integer set $\{1,2,\cdots,m\}$ and $\{1,2,\cdots,n\}$, respectively.

2: Taking as input the security parameter $\lambda$, select two sets of positive integers $K_\alpha$ and $K_\beta$, where the sizes of $K_\alpha$ and $K_\beta$ are both $2^\lambda$. Randomly, uniformly, and independently select two sets of positive numbers: $\{\alpha_1,\cdots,\alpha_m\} \leftarrow K_\alpha$, $\{\beta_1,\cdots,\beta_n\} \leftarrow K_\beta$.

3: Generate matrices $\boldsymbol{M} \in \mathbb{R}^{m\times m}, \boldsymbol{N} \in \mathbb{R}^{n\times n}$ as

$$\begin{cases} \boldsymbol{M}(i,j) = \alpha_i \delta_{\pi_1(i),j},\ 1 \le i,j \le m; \\ \boldsymbol{N}(i,j) = \beta_i \delta_{\pi_2(i),j},\ 1 \le i,j \le n, \end{cases}$$

where $\boldsymbol{M}(i,j)$ and $\boldsymbol{N}(i,j)$ denote the elements in the $i$-th row and $j$-th column of $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively; $\delta_{x,y}$ denotes the Kronecker delta function that returns 1 if $x = y$ or 0 otherwise.

For $\boldsymbol{M}$ and $\boldsymbol{N}$, their inverse matrices $\boldsymbol{M}^{-1} = (\boldsymbol{M}^{-1}(i,j)) \in \mathbb{R}^{m\times m}$ and $\boldsymbol{N}^{-1} = (\boldsymbol{N}^{-1}(i,j)) \in \mathbb{R}^{n\times n}$ are calculated as follows:

$$\begin{cases} \boldsymbol{M}^{-1}(i,j) = \alpha_j^{-1} \delta_{\pi_1^{-1}(i),j},\ 1 \le i,j \le m; \\ \boldsymbol{N}^{-1}(i,j) = \beta_j^{-1} \delta_{\pi_2^{-1}(i),j},\ 1 \le i,j \le n. \end{cases} \tag{2}$$

### B. $\mathbf{TaskTrans}(\boldsymbol{V},K) \mapsto \boldsymbol{V}'$

To protect the privacy of the input matrix $\boldsymbol{V} \in \mathbb{R}^{m\times n}$, the client intends to mask it via left and right multiplications with $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively. Specifically, the client generates $\boldsymbol{V}' = \boldsymbol{M}\boldsymbol{V}\boldsymbol{N}$ and then sends it to the cloud.

### C. $\mathbf{TaskSol}(\boldsymbol{V}') \mapsto (\boldsymbol{W}',\boldsymbol{H}')$

Given the masked matrix $\boldsymbol{V}'$, the cloud conducts the NMF of $\boldsymbol{V}'$ following the algorithm proposed in [14], [15], and then returns the computing results $(\boldsymbol{W}',\boldsymbol{H}')$ to the client.

### D. $\mathbf{Verify}(\boldsymbol{V}',(\boldsymbol{W}',\boldsymbol{H}')) \mapsto \texttt{true/false}$

To detect possible cheating behaviors by the cloud, the client should check the correctness of the solution to the masked task received from the cloud. Specifically, the client needs to verify whether $(\boldsymbol{W}',\boldsymbol{H}')$ forms a non-negative matrix factorization of $\boldsymbol{V}'$, i.e., whether all elements in $\boldsymbol{W}'$ and $\boldsymbol{H}'$ are non-negative and whether $\|\boldsymbol{V}'-\boldsymbol{W}'\boldsymbol{H}'\|_{\mathrm{F}}^2 \le \epsilon$. The client accepts $(\boldsymbol{W}',\boldsymbol{H}')$ and outputs `true` if $\boldsymbol{W}' \ge 0$, $\boldsymbol{V}' \ge 0$, and $\|\boldsymbol{V}'-\boldsymbol{W}'\boldsymbol{H}'\| < \varepsilon$; it terminates and output `false` otherwise.

### E. $\mathbf{Recover}((\boldsymbol{W}',\boldsymbol{H}'),K) \mapsto (\boldsymbol{W},\boldsymbol{H})$

Given the verified $(\boldsymbol{W}',\boldsymbol{H}')$, the client restores with $K$ the NMF results of the input matrix $\boldsymbol{V}$ as

$$\begin{cases} \boldsymbol{W} = \boldsymbol{M}^{-1}\boldsymbol{W}', \\ \boldsymbol{H} = \boldsymbol{H}'\boldsymbol{N}^{-1}, \end{cases} \tag{3}$$

where $\boldsymbol{M}^{-1}$ and $\boldsymbol{N}^{-1}$ are the inverse matrices of $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively, and are calculated following eq. (2) in Section V-A.

## VI. Analytic Evaluation

### A. Cheating Resistance

**Theorem 1.** *The proposed scheme is cheating-resistant.*

*Proof:* For the client to output `true`, the $\boldsymbol{W}'$ and $\boldsymbol{H}'$ returned by the cloud should be non-negative; otherwise, the client rejects $(\boldsymbol{W}',\boldsymbol{H}')$ immediately. Then, the client calculates $\|\boldsymbol{V}'-\boldsymbol{W}'\boldsymbol{H}'\|_{\mathrm{F}}^2$ and compares it with $\epsilon$. Such verification can be efficiently conducted as the computation complexity is as low as $\mathcal{O}(mnp)$, which is relatively lightweight compared to that of solving an NMF task (cf. Section II-A, $\mathcal{O}(lmnp)$). Therefore, if $(\boldsymbol{W}',\boldsymbol{H}')$ returned by the cloud is not a solution to the masked task, it can pass the verification with probability 0, i.e., $\Pr(\mathbf{Verify}(\boldsymbol{V}',(\boldsymbol{W}',\boldsymbol{H}')) \mapsto \texttt{true}) = 0$. ■

### B. Recoverability

The recoverability of our proposal requires that the matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ restored by the client should be both non-negative and satisfy $\|\boldsymbol{V}-\boldsymbol{W}\boldsymbol{H}\|_{\mathrm{F}}^2 \le \epsilon$. For proving the recoverability, we first present the following lemma.

**Lemma 1.** *In* $\mathbf{TaskTrans}$*, for* $\boldsymbol{V}' = \boldsymbol{M}\boldsymbol{V}\boldsymbol{N}$*, it holds that*

$$\boldsymbol{V}'(i,j) = \alpha_i \beta_{\pi_2^{-1}(j)} \boldsymbol{V}(\pi_1(i),\pi_2^{-1}(j)),$$

*where* $\boldsymbol{V}'(i,j)$ *and* $\boldsymbol{V}(i,j)$ *denote the elements on the* $i$*-th row and* $j$*-th column of* $\boldsymbol{V}'$ *and* $\boldsymbol{V}$*, respectively.*

*Proof:* Let

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}(1,1) & \cdots & \boldsymbol{V}(1,n) \\ \vdots & \ddots & \vdots \\ \boldsymbol{V}(m,1) & \cdots & \boldsymbol{V}(m,n) \end{bmatrix}.$$

Following Alg. 2, $\boldsymbol{M}(i,j) = \alpha_i \delta_{\pi_1(i),j},\ 1 \le i,j \le m$. The element on the $i$-th row and $j$-th column of $\boldsymbol{M}\boldsymbol{V}$ is computed as $\sum_{k=1}^{m} \boldsymbol{M}(i,k)\boldsymbol{V}(k,j)$. That is,

$$\boldsymbol{M}\boldsymbol{V} = \begin{bmatrix} \sum_{k=1}^{m} \alpha_1 \delta_{\pi_1(1),k} \boldsymbol{V}(k,1) & \cdots & \sum_{k=1}^{m} \alpha_1 \delta_{\pi_1(1),k} \boldsymbol{V}(k,n) \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{m} \alpha_m \delta_{\pi_1(m),k} \boldsymbol{V}(k,1) & \cdots & \sum_{k=1}^{m} \alpha_m \delta_{\pi_1(m),k} \boldsymbol{V}(k,n) \end{bmatrix}.$$

Since $\delta_{x,y}$ (i.e., the Kronecker delta function) returns 1 if $x = y$ or 0 otherwise, $\delta_{\pi_1(i),k} = 1$ iff $k = \pi_1(i)$, Therefore,

$$\boldsymbol{M}\boldsymbol{V} = \begin{bmatrix} \alpha_1 \boldsymbol{V}(\pi_1(1),1) & \cdots & \alpha_1 \boldsymbol{V}(\pi_1(1),n) \\ \vdots & \ddots & \vdots \\ \alpha_m \boldsymbol{V}(\pi_1(m),1) & \cdots & \alpha_m \boldsymbol{V}(\pi_1(m),n) \end{bmatrix}.$$

Similarly, $\delta_{\pi_2(k),j} = 1$ iff $k = \pi_2^{-1}(j)$. Thus, the element on the $i$-th row and $j$-th column of $\boldsymbol{V}' = \boldsymbol{MVN}$ is computed as

$$\begin{aligned} & \sum_{k=1}^{n} \big(\alpha_i \boldsymbol{V}(\pi_1(i),k)\big)\boldsymbol{N}(k,j) \\ = & \sum_{k=1}^{n} \big(\alpha_i \boldsymbol{V}(\pi_1(i),k)\big)\big(\beta_k \delta_{\pi_2(k),j}\big) \\ = & \alpha_i \beta_{\pi_2^{-1}(j)} \boldsymbol{V}(\pi_1(i), \pi_2^{-1}(j)). \end{aligned}$$

■

**Theorem 2.** *The proposed scheme is recoverable for the NMF task.*

*Proof:* According to eq. (2), each element of $\boldsymbol{M}^{-1}$ is non-negative. Therefore, the matrix product $\boldsymbol{W} = \boldsymbol{M}^{-1}\boldsymbol{W}'$ is still non-negative. It is the similar case for $\boldsymbol{H} = \boldsymbol{H}'\boldsymbol{N}^{-1}$.

Considering the returned matrices $\boldsymbol{W}' = (\boldsymbol{W}'(i,j)), \boldsymbol{H}' = (\boldsymbol{H}'(i,j))$ sat. $\|\boldsymbol{V}' - \boldsymbol{W}'\boldsymbol{H}'\|_{\mathrm{F}}^2 \leq \epsilon$, we next prove that $\|\boldsymbol{V} - \boldsymbol{WH}\|_{\mathrm{F}}^2 \leq \epsilon$. For $\boldsymbol{W}' = \boldsymbol{WM}, \boldsymbol{H}' = \boldsymbol{HN}$, following the proof of Lemma 1,

$$\begin{aligned} & \boldsymbol{W}'(i,j) = \sum_{k=1}^{m} \boldsymbol{M}(i,k)\boldsymbol{W}(k,j) \\ = & \sum_{k=1}^{m} \alpha_i \delta_{\pi_1(i),k} \boldsymbol{W}(k,j) = \alpha_i \boldsymbol{W}(\pi_1(i), j), \end{aligned}$$

and

$$\begin{aligned} & \boldsymbol{H}'(i,j) = \sum_{k=1}^{n} \boldsymbol{H}(i,k)\boldsymbol{N}(k,j) \\ = & \sum_{k=1}^{n} \boldsymbol{H}(i,k)\beta_k \delta_{\pi_2(k),j} = \beta_{\pi_2^{-1}(j)} \boldsymbol{H}(i, \pi_2^{-1}(j)). \end{aligned}$$

According to the definition of Frobenius norm (cf. Section II-A), we have

$$\begin{aligned} & \|\boldsymbol{V}' - \boldsymbol{W}'\boldsymbol{H}'\|_{\mathrm{F}}^2 \\ = & \sum_{i=1}^{m}\sum_{j=1}^{n} |\boldsymbol{V}'(i,j) - \sum_{k=1}^{p} \boldsymbol{W}'(i,k)\boldsymbol{H}'(k,j)|^2 \\ = & \sum_{i=1}^{m}\sum_{j=1}^{n} |\alpha_i \beta_{\pi_2^{-1}(j)} \boldsymbol{V}(\pi_1(i), \pi_2^{-1}(j)) \\ & - \sum_{k=1}^{p} \alpha_i \beta_{\pi_2^{-1}(j)} \boldsymbol{W}(\pi_1(i),k)\boldsymbol{H}(k, \pi_2^{-1}(j))|^2 \\ = & \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_i^2 \beta_{\pi_2^{-1}(j)}^2 \big|\boldsymbol{V}(\pi_1(i), \pi_2^{-1}(j)) \\ & - \sum_{k=1}^{p} \boldsymbol{W}(\pi_1(i),k)\boldsymbol{H}(k, \pi_2^{-1}(j))\big|^2 \\ = & \alpha_i^2 \beta_{\pi_2^{-1}(j)}^2 \|\boldsymbol{V} - \boldsymbol{WH}\|_{\mathrm{F}}^2 \\ \geq & \alpha_{\min}^2 \beta_{\min}^2 \|\boldsymbol{V} - \boldsymbol{WH}\|_{\mathrm{F}}^2, \end{aligned}$$

where $\alpha_{\min} = \min\{\alpha_1, \cdots, \alpha_m\} \geq 1$ and $\beta_{\min} = \min\{\beta_1, \cdots, \beta_n\} \geq 1$. Therefore, $\|\boldsymbol{V} - \boldsymbol{WH}\|_{\mathrm{F}}^2 \leq \frac{1}{\alpha_{\min}^2 \beta_{\min}^2} \|\boldsymbol{V}' - \boldsymbol{W}'\boldsymbol{H}'\| \leq \|\boldsymbol{V}' - \boldsymbol{W}'\boldsymbol{H}'\| \leq \epsilon$. ■

### C. Privacy Preserving

**Theorem 3.** *The proposed scheme is privacy-preserving in the fully malicious model.*

*Proof:* Following the privacy definition in Section III-D, we next prove that the cloud cannot infer any of the input matrix, the factorization results, and the clustering property of the original NMF task.

First, according to **TaskTrans** in Section V-B, the masked matrix is generated as $\boldsymbol{V}' = \boldsymbol{MVN}$. As shown in Lemma 1, the element on the $i$-th row and $j$-th column of $\boldsymbol{V}'$ is $\alpha_i \beta_{\pi_2^{-1}(j)} \boldsymbol{V}(\pi_1(i), \pi_2^{-1}(j))$. Such mask can be viewed as a two-phase operation:

- Phase 1: The position of each element in $\boldsymbol{V}$ is randomly rearranged under two permutations to generate $\tilde{\boldsymbol{V}}$, i.e., $\tilde{\boldsymbol{V}}(i,j) = \boldsymbol{V}(\pi_1(i), \pi_2^{-1}(j))$.
- Phase 2: Each element in $\tilde{\boldsymbol{V}}$ is further masked by multiplying a positive factor to generate $\boldsymbol{V}'$, i.e., $\boldsymbol{V}'(i,j) = \alpha_i \beta_{\pi_2^{-1}(j)} \tilde{\boldsymbol{V}}(i,j)$.

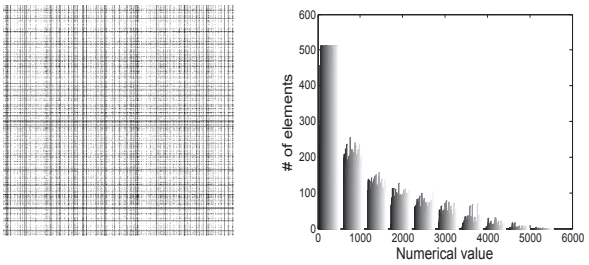
In Phase 1, there $m!$ and $n!$ cases of random permutations $\pi_1$ and $\pi_2$, respectively. This implies that there are $m!n!$ possible ways to rearrange the elements of $\boldsymbol{V}$. In Phase 2, the original matrix is further masked by two positive numbers randomly, uniformly, and independently chosen from $K_\alpha$ and $K_\beta$, respectively. Following **KeyGen**, $|K_\alpha| = |K_\beta| = 2^\lambda$. Therefore, the expected time for the adversary to recover $\boldsymbol{V}$ is $\mathcal{O}\big(m!n!2^{\lambda(m+n)}\big)$, which is a non-polynomially bounded quantity in terms of $m$ and $n$. That is, the privacy of the input matrix is guaranteed. It is worthy noting that the numerical distribution of elements in $\boldsymbol{V}$ is naturally protected with the positive factors from $K_\alpha$ and $K_\beta$.

Second, according to eq. (3), the NMF results of $\boldsymbol{V}$, i.e., $\boldsymbol{W}$ and $\boldsymbol{H}$, are preserved in a similar way for protecting $\boldsymbol{V}$. Following the above analysis, the privacy of the factorization results is also guaranteed.

Last, we show that the clustering property of the original NMF task is protected. According to the elaborate construction of $\boldsymbol{N}$, right-multiplying $\boldsymbol{H}'$ with $\boldsymbol{N}^{-1}$ not only permutates the columns of $\boldsymbol{H}'$ but also masks each $\boldsymbol{H}'(i,j)$ in $\boldsymbol{H}'$ with a random positive number. Therefore, the cloud cannot derive the clustering of the columns of $\boldsymbol{V}$ based on the columns of $\boldsymbol{H}'$. ■

To demonstrate the enhanced privacy protection of our proposal compared to the scheme in [7], here we take the trouble to present a specific example to show that our proposal hides the numerical distribution of the elements in the input matrix. Again we take the corresponding image matrix of Fig. 2(a) as the input of the NMF task and mask the image matrix with our proposal. Fig. 4(a) illustrates the obfuscated image corresponding to masked image matrix while its numerical distribution is shown in Fig. 4(b).

As illustrated above, the numerical distribution of elements in $\boldsymbol{V}'$ (shown in Fig. 4(b)) is significantly different from that of elements in the original image matrix (shown in Fig. 2(b)), which confirms our theoretical analysis.




(a) (b)

Fig. 4. (a) Masked form of Fig. 2(a) that is obfuscated with our proposal; (b) The numerical distribution of the masked matrix.

TABLE I. THE CLIENT'S PROCESSING TIME (IN SECONDS) FOR SOLVING NMF TASKS OF DIFFERENT SCALES

| $m$ | $n$ | $p$ | $t_{\text{client}}$ | $t_{\text{cloud}}$ | $t_{\text{direct}}$ | $\frac{t_{\text{direct}}}{t_{\text{client}}}$ |
|---|---|---|---|---|---|---|
| 2,000 | 1,000 | 100 | 2.12 | 18.69 | 45.27 | 21.35 |
| 4,000 | 2,000 | 200 | 3.57 | 28.72 | 82.39 | 23.08 |
| 6,000 | 3,000 | 300 | 4.87 | 43.28 | 128.56 | 26.39 |
| 8,000 | 4,000 | 400 | 8.47 | 123.64 | 308.91 | 36.47 |

TABLE II. THE CLIENT'S PROCESSING TIME (IN SECONDS) FOR SOLVING NMF TASKS OF DIFFERENT SCALES

| $m$ | $n$ | $p$ | $t_{\text{client}}$ | $t_{\text{cloud}}$ | $t_{\text{direct}}$ | $\frac{t_{\text{direct}}}{t_{\text{client}}}$ |
|---|---|---|---|---|---|---|
| 38,804 | 100 | 20 | 4.48 | 39.67 | 98.77 | 22.05 |
| 38,804 | 100 | 40 | 5.19 | 49.16 | 133.23 | 25.17 |
| 38,804 | 100 | 60 | 5.62 | 57.56 | 166.37 | 29.61 |

### D. Complexity

Now we investigate the computation complexity on the client side.

- In **TaskTrans**, since both $\boldsymbol{M}$ and $\boldsymbol{N}$ are sparse matrices, the client only needs to spend $\mathcal{O}(mn)$ costs calculating $\boldsymbol{MV}$ and $(\boldsymbol{MV})\boldsymbol{N}$, respectively.
- In **Verify**, the client spends $\mathcal{O}(mnp)$ computation costs calculating $\boldsymbol{W'H'}$ and $\mathcal{O}(mn)$ costs checking whether $\|\boldsymbol{V'} - \boldsymbol{W'H'}\|_{\text{F}}^2 \leq \epsilon$.
- In **Recover**, the client spends $\mathcal{O}(mp)$ and $\mathcal{O}(pn)$ costs computing $\boldsymbol{W} = \boldsymbol{M}^{-1}\boldsymbol{W}$ and $\boldsymbol{H} = \boldsymbol{H'N}^{-1}$, respectively.

To sum up, the overall computation complexity is $\mathcal{O}(mnp)$ and is significantly lower than that of locally solving an NMF task, which is $\mathcal{O}(lmnp)$ (cf. Section II-A, $l$ represents the number of iterations for computing the factorization).

## VII. EXPERIMENTAL EVALUATION

In this section, we evaluate our proposal with extensive experiments on both synthetic and real-world data. We implement the client on a laptop computer with an Intel i5-5200U processor running at 2.20GHz and 8GB memory, and emulate the cloud on a desktop workstation with an Intel i7-6700 processor running at 3.40GHz and 8GB memory. The algorithm in [14], [15] is employed for NMF factorization. All proposed algorithms are implemented with MATLAB.

### A. Cost Savings Owing to Our Proposal

To evaluate the computation overhead for solving NMF tasks that involves input matrices with different dimensions, we adopt the processing time as the performance metric. We construct random non-negative matrices $\boldsymbol{V}$'s with dimensions $m \times n$ ranging from $2,000 \times 1,000$ to $8,000 \times 4,000$; any element in $\boldsymbol{V}$'s is randomly, uniformly, and independently sampled from the same interval $[0.0, 1024.0]$. Just for simplicity, $(m, n)$ is instantiated with $m = 2n$ and the factorization parameter $p$ is set to $\frac{m}{20}$. Two scenarios are tested for comparison. In the first scenario, the client deploys our NMF outsourcing scheme; we measures the processing time on both the client and the cloud sides. In the second scenario, the client does not turn to the cloud for assistance and locally conducts the NMF task.

Table 1 illustrates the client's processing time (averaged from repeated but independent experiments) in the two scenarios. In Table 1 $t_{\text{client}}$ is the overhead of the client employing our proposal, which is the sum of the time costs of **TaskTrans**, **Verify**, and **Recover**, and $t_{\text{cloud}}$ is the overhead of the cloud by conducting **TaskSol**; $t_{\text{direct}}$ is the overhead of the client when locally solving the NMF tasks. The results show that in each experiment, the client's processing time in the first scenario (i.e., privacy-preserving cloud computing) is significantly smaller than that in the second scenario (i.e., local computing). To make this more clear, we adopt the ratio of $t_{\text{direct}}$ to $t_{\text{client}}$ as an indicator for cost savings. We make the observation that this indicator increases as the dimensions of the input matrix increase, which means the larger the input matrix is the more performance gain is obtained. Besides, the total elapsed time of our proposal, i.e., $t_{\text{client}} + t_{\text{cloud}}$, is far less than that of the local scenario, i.e., $t_{\text{direct}}$, which implies that our proposal is a pragmatic proposal.

To sum up, our proposal not only brings significant cost savings to the client but also shortens the actual time for the solving NMF tasks.

### B. Real-world Applications

In addition to synthetic data, we also conduct experiments based on real-word data. We consider the factorization of face images, which is also investigated in [14]. Specifically, for evaluating our proposal, we adopt a subset of the face image database named CelebA obtained from http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. The subset we use comprises 100 face images, each of which is of size $218 \times 178$. For matrix factorization, each face image is vectorized as a column of the non-negative matrix $\boldsymbol{V}$ to be factorized. That is, the dimensions $(m, n)$ of $\boldsymbol{V}$ are $(218 \times 178, 100) = (38,804, 100)$. We conduct three independent sets of experiments, in which the factorization parameters are set to 20, 40, and 60, respectively.

The experimental results are presented in Table 2, which show that the client enjoys significant cost savings by deploying our proposal. In addition, the total elapsed time (i.e., the sum of $t_{\text{client}}$ and $t_{\text{cloud}}$) is far less than that of the local scenario.

Furthermore, some selected face images and their reconstructed versions using the factor matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ (with the factorization parameter being 40) are illustrated in Fig. 5. The first row shows the original face images, and the reconstructed ones are shown in the second row. As can be seen, the results

Fig. 5. Selected face images from CelabA and their reconstructed versions with our NMF outsourcing scheme. The factorization parameter is set to 40. The original images are illustrated in the first row while the corresponding reconstructed ones are in the second row.

validate the recoverability of our proposal and demonstrate the good utility of the factorization results restored by the client.

## VIII. Conclusion

In this paper, we are interested in solving a non-negative matrix factorization task, which becomes challenging for a resource-constrained client when the dimensions of the input matrix increase. In practice, large-scale NMF tasks abound in many real-world applications. To confront such challenges, we propose to harness the cloud for secure and efficient NMF outsourcing. To this end, we have formalized the problem of secure NMF outsourcing by identifying the system/threat model, formulating the scheme components, and presenting the design goals. We have also revealed the privacy breaches of a recent NMF outsourcing scheme. For enhanced privacy protection, we have presented a new secure and efficient solution. By employing elaborately constructed matrices for matrix masking, our proposal guarantees the privacy of both the input matrix and the factorization results, and protects the important clustering property of the NMF task. Last, we have conducted extensive experiments on both synthetic and real-world data. The evaluation results show that our scheme brings significant cost savings to the client and simultaneously guarantees good utility.

## Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Basic Research Program of China (973 Program) under Grant 2014CB340603.

## References

[1] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with bregman divergences," in *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS '05)*, 2005, pp. 283–290.

[2] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*, 2011, pp. 69–77.

[3] Y. Bao, H. Fang, and J. Zhang, "TopicMF: Simultaneously exploiting ratings and reviews for recommendation," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*, 2014, pp. 2–8.

[4] S. Marstona, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing - The business perspective," *Decision Support Systems*, vol. 51, pp. 176–189, 2011.

[5] K. Ren, C., and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 24, pp. 69–73, 2012.

[6] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 69–78, 2015.

[7] J. Duan, J. Zhou, and Y. Li, "Secure and verifiable outsourcing of non-negative matrix factorization (NMF)," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '16)*, 2016, pp. 63–68.

[8] S. Salinas, C. Luo, X. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *Proceedings of the 34th IEEE International Conference on Computer Communications (INFOCOM '15)*, 2015, pp. 1035–1043.

[9] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. Spafford, "Secure outsourcing of scientific computations," *Advances in Computers*, vol. 7, pp. 215–272, 2002.

[10] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM '11)*, 2011, pp. 820–828.

[11] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1172–1181, 2013.

[12] C. Wang, K. Ren, K. Wang, and K. M. R. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, 2011, pp. 549–558.

[13] Y. Yu, Y. Luo, D. Wang, S. Fu, and M. Xu, "Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations," in *Proceedings of 2016 IEEE International Conference on Communications (ICC '16)*, 2016.

[14] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negativ matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[15] ——, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS '13)*, 2008, pp. 535–541.

[16] C. Ding, X. He, H. D. Simon, and R. Jin, "On the equivalence of nonnegative matrix factorization and k-means - spectral clustering," in *Proceedings of the SIAM International Conference on Data Mining*, 2008.

[17] Lyndon and C. Roger, *Combinatorial Group Theory*. Springer, 1977.

[18] J. W. Demmel, *Applied numerical linear algebra*. SIAM, 1997.

[19] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification." *IEEE Transactions on Neural Networks*, vol. 10, no. 5, p. 1055, 1999.

[20] O. Chapelle, P. Haffner, and V. Vapnik, "SVMs for histogram-based image classification," *IEEE Transactions on Neural Networks*, 1999.

[21] T. M. Buzug, J. Weese, C. Fassnacht, and C. Lorenz, "Image registration: Convex weighting functions for histogram-based similarity measures," *Proceedings of CVRMed-MRCAS'97, Lecture Notes in Computer Science*, vol. 1205, no. 4, pp. 203–212, 1997.