Secure and Verifiable Outsourcing of Large-scale Nonnegative Matrix Factorization (NMF)

Jia Duan, *Student Member, IEEE,* Jiantao Zhou, *Senior Member, IEEE,* and Yuanman Li, *Member, IEEE*

Abstract—Nowadays, cloud computing platforms are becoming increasingly prevalent and readily available, providing alternative and economic services for resource-constrained clients to perform large-scale computations. This work addresses the problem of secure outsourcing of large-scale nonnegative matrix factorization (NMF) to a cloud in a way that the client can verify the correctness of the results with small overhead. The protection of the input matrix is achieved by a random permutation and scaling encryption mechanism. By exploiting the iterative nature of NMF computation, we propose a single-round verification strategy, which can be proved to be quite effective. Theoretical and experimental results are provided to show the superior performance of the proposed scheme.

Index Terms—Cloud computing, NMF, secure outsourcing, verification.

1 Introduction

In the big data era, quintillion bytes of data are being created daily at an astounding speed. These data are generated from almost everywhere: digital images and videos, purchase transaction records, cell phone GPS signals, sensors for gathering climate information, posts on social media networks, just to name a few. To become competitive, organizations need to efficiently convert the large amount of raw data into significant insights, guiding their strategies for investment, marketing, research, etc.

Nonnegative matrix factorization (NMF), as a fundamental data analysis technique, has proved its usefulness in many applications, ranging from source separation [1], document clustering [2], [3], and feature extraction [4], [5]. Due to the huge volume of data to be processed nowadays, the usage of NMF in practical scenarios has undergone a few challenges. Traditional algorithms and frameworks are not designed to deal with the amount of information present in these large-scale problems. Meanwhile, for small to medium sized organizations, the computations involved in performing the data analysis via NMF could be prohibitive, beyond their computing capabilities.

Fortunately, cloud computing provides a practicable solution for resource-constrained clients to perform large-scale computations [6]. In the outsourced computation framework, the resource-constrained clients can outsource the heavy computational tasks to the cloud server with massive computational power. Both the clients and the cloud can therefore economize large expenditure in hardware, software, deployment and maintenance. However, these cloud services and architectures [7], [8] also bring new security and privacy challenges. First of all, the computational tasks often contain sensitive information that should not be exposed to the cloud server, which usually cannot be fully trusted. Traditional encryption algorithms can only partially solve the protection problem, as it is generally difficult to perform meaningful (complex) computations over the encrypted domain. Furthermore, clients should have a mechanism to verify the correctness of the results, because the cloud may return invalid solutions for financial incentives. Certainly, the overhead induced in performing the verification should be minimized.

Secure outsourcing computation has attracted considerable attention in recent years. Aiming at covering all types of computations, two generic protocols for secure outsourcing computation were proposed by Gennaro *et al.* [9] and Chung *et al.* [10], based on Gentry's work on fully homomorphic encryption (FHE) [11]. Qin *et al.* [12] proposed a framework based on FHE to enable an interested party to detect the global feature of image data without compromising the user's privacy. To this end, the homomorphic properties of FHE were utilized to decompose the existing image feature detection algorithms into circuit-level operations that can be performed in the encrypted domain. Ishimaki *et al.* [13] presented a protocol which can enable a client to perform string search on a genome sequence database without leaking his query to the server. They utilized FHE to encrypt the query and designed an efficient bootstrapping method to reduce the random noise produced by FHE. For the sake of generality, Liu *et al.* [14] proposed a generic calculation toolkit which allows the data owners to outsource the arithmetic operations to the server without the privacy leakage. Unfortunately, the outsourcing frameworks based on FHE are far from practical because of the extremely high complexity of FHE operations. High cost in these general solutions drives researchers to seek more efficient schemes for

Jia Duan, Jiantao Zhou, and Yuanman Li are with the State Key Laboratory of Internet of Things for Smart City, and the Department of Computer and Information Science, University of Macau, China. Emails: {xuelandj@gmail.com; jtzhou@um.edu.mo; yuanmanx.li@gmail.com} (Corresponding author: Jiantao Zhou). *This work was supported in part by the Macau Science and Technology Development Fund under Grants FDCT/022/2017/A1 and FDCT/077/2018/A2, and in part by the Research Committee at the University of Macau under Grants MYRG2016-00137-FST and MYRG2018-00029-FST.*

specific problems, rather than ambitiously targeting generic computations. So far, many secure outsourcing schemes have been proposed, attempting to leverage a public cloud to perform various computations, e.g., modular exponentiations [15], [16], solving linear equations [17], [18], [19], linear programming [20], [21], [22], biometric computations [23], sequence comparisons [24], [25], keyword search over outsourced data [26], [27], image reconstruction [28], and image feature extraction [12], [29].

Among the various proposals for secure outsourcing computation, the ones concerning matrix computations are the most relevant to our work. The problem of secure and verifiable outsourcing of matrix multiplication was first addressed in [30], [31], with redundant execution approaches. Later, Lei *et al.* [32] investigated the same problem by proposing a new matrix encryption scheme via sparse matrix multiplication. As checking multiplication results completely is prohibitive for resource-constrained clients, the sampling-based verification method (i.e., only check the validity of randomly selected entries) was adopted. Furthermore, Atallah *et al.* [33] designed a provably secure protocol for outsourced matrix multiplication, based on Shamir's secret sharing [34]. The verification was conducted through some randomly inserted rows. Besides matrix multiplications, some other matrix-related computations concerning, e.g., matrix determinant calculation [35], and matrix rank decomposition [36] were also discussed.

In this work, we address the problem of secure and verifiable outsourcing of large-scale NMF tasks. Random permutation and scaling based symmetric encryption scheme is adopted to guarantee the security of input matrix. As the factorization is only an approximation of the original matrix and the factorization results are not unique, verifying the correctness of the results becomes a challenging task. To overcome this difficulty, we propose a single-round verification strategy by utilizing the iterative nature of the NMF computation, and prove that the verification strategy is quite effective and efficient. To show the superior performance of our protocol, both theoretical analysis and experimental results are provided.

Difference from conference version: Portions of the work presented in this paper have previously appeared in [37] as a conference version. We have significantly revised and clarified the paper, and improved many technical details compared with [37]. The primary improvements can be summarized as follows. First of all, we define a new criterion for acceptable NMF results, based on which we re-prove the validity of the verification strategy and the correctness of the outsourced NMF protocol. This new criterion eliminates the undesirable effect of the data range on the performance parameter τ, allowing us to determine it for all types of matrices in a universal way. Secondly, we design an improved encryption scheme which can randomly change both the positions and the magnitudes of the entries of the input matrix. Thirdly, we offer an in-depth study on the parameter ϵ, which is crucial for the proposed verification scheme, and present an offline learning method to obtain it in practice. Fourthly, we analyze the security and verifiability of our proposed scheme in a much more formal manner. In addition, though straightforward, we re-implement the encryption scheme by explicitly noticing its permutation nature, which in practice results in significant reduction of the computational complexity on the client side. Finally, all the experiments in our performance evaluation are completely re-done. Especially, we investigate how the dimension parameter r, the performance parameter τ, and the matrix size influence the system performance, which are lacking in [37].

The rest of the paper is organized as follows. An overview of NMF is given in Section 2. In Section 3, we describe the system model, the design goals, and the threat model. Section 4 presents the proposed protocol for secure and verifiable outsourcing of NMF. Section 5 and 6 are devoted to security and system performance analyses. Section 7 offers the experimental results and Section 8 concludes.

2 Nonnegative Matrix Factorization (NMF)

Given a nonnegative matrix $\mathbf{V} \in \mathbb{R}^{m\times n}$, NMF finds nonnegative factors $\mathbf{W}$ and $\mathbf{H}$ such that

$$\mathbf{V} \approx \mathbf{WH} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{m\times r}$, $\mathbf{H} \in \mathbb{R}^{r\times n}$, and r is a dimension parameter. The factorization is usually sought through the following minimization problem

$$\min_{\mathbf{W},\mathbf{H}} D(\mathbf{V}|\mathbf{WH}) \quad \text{subject to } \mathbf{W} \geq 0, \mathbf{H} \geq 0 \tag{2}$$

where $\mathbf{W} \geq 0, \mathbf{H} \geq 0$ means that all the elements of $\mathbf{W}$ and $\mathbf{H}$ are nonnegative. The cost function $D(\mathbf{V}|\mathbf{WH})$ is often designed to be a separable measure of fit between $\mathbf{V}$ and $\mathbf{WH}$, namely

$$D(\mathbf{V}|\mathbf{WH}) = \sum_{i=1}^{m}\sum_{j=1}^{n} d(\mathbf{V}_{i,j}|(\mathbf{WH})_{i,j}) \tag{3}$$

where $\mathbf{V}_{i,j}$ denotes the (i,j)th element of $\mathbf{V}$, and $d(x|y)$ is a scalar cost function of x given y. In this work, we only consider the cost function with separable form as stated above.

In [38], Tan and Fevotte considered $d(x|y)$ to be the β-divergence, a family of cost functions parameterized by a single scalar $\beta \in \mathbb{R}$. The β-divergence can be expressed as

$$d_\beta(x|y) = \begin{cases} \frac{x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}}{\beta(\beta-1)}, & \beta \in \mathbb{R}\setminus\{0,1\} \\ x\log\frac{x}{y} - x + y, & \beta = 1 \\ \frac{x}{y} - \log\frac{x}{y} - 1, & \beta = 0 \end{cases} \tag{4}$$

The widely used cost functions, e.g., squared Euclidean distance, generalized Kullback-Leibler (KL) divergence, and Itakura-Saito (IS) divergence are special cases of β-divergence when β is assigned with different values. To be consistent with the setting of [39], [40], we adopt IS divergence as the cost function, which corresponds to $\beta = 0$.

To solve the non-convex optimization problem in (2), Lee and Seung proposed a multiplicative update algorithm [41], [42], which was later generalized to IS divergence [39]. Specifically, $\mathbf{W}$ and $\mathbf{H}$ are initialized with nonnegative random values, and the following update rules are applied for each entry of $\mathbf{W}$ and $\mathbf{H}$

$$\mathbf{W}_{i,a}^{(k+1)} \leftarrow \mathbf{W}_{i,a}^{(k)} \frac{\left[((\mathbf{W}^{(k)}\mathbf{H}^{(k)})^{\langle -2\rangle} \odot \mathbf{V})(\mathbf{H}^{(k)})^T\right]_{i,a}}{\left[(\mathbf{W}^{(k)}\mathbf{H}^{(k)})^{\langle -1\rangle}(\mathbf{H}^{(k)})^T\right]_{i,a}} \quad \forall\, i,a \tag{5}$$

$$\mathbf{H}_{b,j}^{(k+1)} \leftarrow \mathbf{H}_{b,j}^{(k)} \frac{\left[(\mathbf{W}^{(k+1)})^T((\mathbf{W}^{(k+1)}\mathbf{H}^{(k)})^{\langle -2\rangle} \odot \mathbf{V})\right]_{b,j}}{\left[(\mathbf{W}^{(k+1)})^T(\mathbf{W}^{(k+1)}\mathbf{H}^{(k)})^{\langle -1\rangle}\right]_{b,j}} \quad \forall\, b,j \tag{6}$$

where $\mathbf{W}_{i,a}^{(k)}$ and $\mathbf{H}_{b,j}^{(k)}$ denote the (i,a)th and the (b,j)th elements of the resulting factors $\mathbf{W}^{(k)}$ and $\mathbf{H}^{(k)}$ in the kth iteration. Also, the operator $\odot$ denotes Hadamard element-wise product and $\mathbf{A}^{\langle s\rangle}$ denotes the matrix with element-wise exponentiation to non-zero entries, while the zero elements remain unchanged. This iteration process continues until local minimum is achieved.

It was shown in [39] that the computational complexity of running the above iterative algorithm for one iteration is $O(mnr)$, and hence, the complexity of performing the whole NMF is of order $\#\texttt{iterations} \times O(mnr)$, in which $\#\texttt{iterations}$ represents the number of iterations needed to achieve local minimum. In theory, the value of $\#\texttt{iterations}$ depends on the matrix characteristics, and cannot be regarded as a hidden constant. In practice, it is difficult to estimate the value of $\#\texttt{iterations}$, and a large value, e.g., 10^4, is used to ensure convergence. Alternatively, Arora [43] derived a more accurate complexity estimate $O((mn)^{r^2 2^r})$ for running the entire NMF algorithm (not single iteration). Therefore, when handling large-scale matrices, the incurred complexity is prohibitively high, motivating us to outsource the heavy computations to a cloud.

Ideally, $\mathbf{W}^{(\infty)}$ and $\mathbf{H}^{(\infty)}$, namely, the results after iterating infinite number of times, are the desirable solutions for the factors. In practice, due to the numerical accuracy, the factors $\mathbf{W}^{(k)}$ and $\mathbf{H}^{(k)}$ satisfying

$$\frac{D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})}{D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})} - 1 \leq \tau \tag{7}$$

are still considered as acceptable, where $\tau > 0$ is a pre-defined performance parameter. As the cost function $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})$ is non-increasing with respect to the iteration index k [44], $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)}) > D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})$ always holds. It should be emphasized that we here do not impose a bound on $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})$ when evaluating whether $\mathbf{W}^{(k)}$ and $\mathbf{H}^{(k)}$ are acceptable or not, because its value is highly dependent on the matrix $\mathbf{V}$ to be factorized and the dimension parameter r. In many practical cases, even for the ideal solution, the value of the cost function $D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})$ is still quite large. In other words, it is rather challenging to determine a universal bound on $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})$ to judge the factorization performance. As a result, in all the existing NMF implementations, e.g., [42], [45], the stopping criterion of the iterations is designed by using the $\#\texttt{iterations}$, rather than using the value of $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})$.

Remark: In our conference version [37], the acceptance criterion for factors $\mathbf{W}^{(k)}$ and $\mathbf{H}^{(k)}$ is defined in a different way, i.e.,

$$\frac{1}{mn}\left\{D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)}) - D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})\right\} \leq \tau$$

A major drawback of this condition is that the value of $\frac{1}{mn}\left\{D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)}) - D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})\right\}$ highly depends on the range of the elements in matrix $\mathbf{V}$. Noticing that the range of the matrix entries in different scenarios could be dramatically different, it is almost impossible to determine a universal τ that works well for all types of matrices. This results in a dilemma that many key parameters of the whole system have to be manually tuned for different matrices, severely affecting the practical usefulness of our proposed scheme. In contrast, in the new acceptance criterion given in (7), the left-hand side does not depend on the range of the matrix data, and hence, the undesirable effect of the data range on the determination of the parameter τ can be eliminated. As will be shown in Section 4, many key parameters of our protocol can be obtained by offline training, and are applicable for all types of matrices without the need of manual tuning.

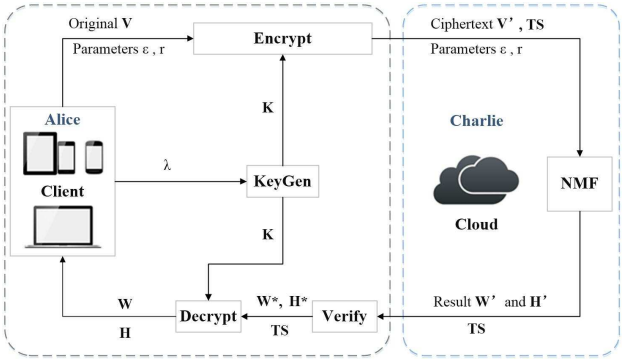


Fig. 1: System model for our proposed scheme.

3 System Model, Design Goals, and Threat Model

3.1 System Model

We consider the secure and verifiable outsourcing of NMF framework depicted in Fig. 1. The client Alice with insufficient computing capabilities intends to outsource NMF of a matrix $\mathbf{V}$ to a cloud server Charlie, who has massive computational power. As Charlie is a third-party server and cannot be fully trusted, Alice encrypts $\mathbf{V}$ into $\mathbf{V}'$ to ensure data secrecy. Alice then passes the encrypted $\mathbf{V}'$, the dimension parameter r, and a stopping criterion parameter ϵ to Charlie for computing the two nonnegative factors $\mathbf{W}'$ and $\mathbf{H}'$ such that

$$\mathbf{V}' \approx \mathbf{W}'\mathbf{H}' \tag{8}$$

We assume that Alice can specify the NMF algorithm to be used by Charlie for completing the outsourced NMF task.

Charlie, upon receiving all the required information from Alice, performs an iterative algorithm, e.g., the one given in (5) and (6), to solve the NMF problem and returns the results $\mathbf{W}'$ and $\mathbf{H}'$ as mentioned above.

With $\mathbf{W}'$ and $\mathbf{H}'$, Alice verifies the correctness of the results using a method to be presented in Section 4. If these

results are acceptable, $\mathbf{W}'$ and $\mathbf{H}'$ are further processed to derive $\mathbf{W}$ and $\mathbf{H}$, respectively, which are the two nonnegative factors for the original matrix $\mathbf{V}$. Otherwise, the results are rejected.

Specifically, the proposed protocol for outsourcing the NMF consists of the following five parts:

- $\mathsf{KeyGen}(\mathsf{NMF}, 1^\lambda) \rightarrow K$: On input a security parameter λ, it produces a private key K to be used for data matrix encryption.
- $\mathsf{Encrypt}(\mathbf{V}, K) \rightarrow (\mathbf{V}', TS)$: The encryption algorithm is run by the client Alice, who takes as input the private key K and the data matrix $\mathbf{V}$. It outputs the encrypted matrix $\mathbf{V}' \in \mathbb{R}^{m\times n}$, and a random nonce TS as a part of the ciphertext.
- $\mathsf{NMF}(\mathbf{V}', TS, r, \epsilon) \rightarrow (\mathbf{W}', \mathbf{H}', TS)$: The NMF computation algorithm is run by the cloud Charlie, who takes as input the ciphertext $(\mathbf{V}', TS)$, the dimension parameter r and the stopping criterion ϵ. It outputs $\mathbf{W}' \in \mathbb{R}^{m\times r}$ and $\mathbf{H}' \in \mathbb{R}^{r\times n}$, which are two nonnegative factors of $\mathbf{V}'$.
- $\mathsf{Verify}(\mathbf{V}', \mathbf{W}', \mathbf{H}', \epsilon) \rightarrow \mathbf{y} \triangleq (\mathbf{W}^*, \mathbf{H}^*) \cup \perp$: The verification algorithm is run by the client Alice, who takes as input $(\mathbf{W}', \mathbf{H}')$, the encrypted matrix $\mathbf{V}'$ and the stopping criterion ϵ, aiming at checking the correctness of the results returned from the cloud. If the verification is successful, it outputs $(\mathbf{W}^*, \mathbf{H}^*)$, and passes them to the decryption algorithm. Otherwise, it rejects the results and produces an error $\perp$.
- $\mathsf{Decrypt}(K, \mathbf{W}^*, \mathbf{H}^*, TS) \rightarrow (\mathbf{W}, \mathbf{H})$: The decryption algorithm is run by the client Alice, who takes as input $(\mathbf{W}^*, \mathbf{H}^*)$ obtained from Verify module, the random nonce TS and the private key K. It outputs two nonnegative factors $\mathbf{W}$ and $\mathbf{H}$ for the original data matrix $\mathbf{V}$.

3.2 Design Goals

To provide confidentiality and verifiability for the above outsourcing framework, we identify the following four design goals, which are consistent with the existing works [17], [18], [19], [35], [46].

- **Confidentiality**: The adversary cannot learn meaningful information concerning the original matrix $\mathbf{V}$ during the process of carrying out the NMF. More formal security definitions will be given in Section 5.
- **Correctness**: The final results $\mathbf{W}$ and $\mathbf{H}$ should be identical or close to those if the NMF computation is conducted locally.
- **Verifiability**: The correct results from a faithful cloud must be verified successfully by the client. No false results from a cheating cloud can pass the verification with non-negligible probability.
- **Efficiency**: The local computation of the client should be substantially less than the original NMF computation on its own. Also, the amount of computation involved in performing NMF over the encrypted $\mathbf{V}'$ should be as close as possible to that over the original $\mathbf{V}$.

Note that some of the design goals are fundamentally conflicting. For instance, if we encrypt $\mathbf{V}$ using the traditional cryptographic algorithm AES, we can achieve very high level of security. However, in this case, the computation over the encrypted matrix would be rather challenging. In other words, we need to strike a balance between the security and the computational efficiency of the encrypted data.

3.3 Threat Model

The security threats faced by the proposed secure and verifiable outsourcing framework mainly come from the behavior of the cloud server Charlie. Generally, there are three types of threat models in generic outsourcing frameworks [35], [47]:

- **Curious Adversary**: Charlie correctly follows the protocol specification; however, he is curious about the information that may be leaked from $\mathbf{V}'$ and any history data that he can access.
- **Lazy Adversary**: Charlie still follows the protocol specification, but may return intermediate results for saving computational resources.
- **Malicious Adversary**: Charlie may not follow the protocol specification and tries to forge results for saving computational resources. In this scenario, the cloud may completely ignore the protocol specification or algorithms run by the cloud.

In these threat models, different adversaries challenge the different security requirements of our proposed framework. Specifically, the curious adversary which is curious about the input and output, mainly challenges the confidentiality of the proposed framework by ciphertext-only attack (COA), chosen-plaintext attack (CPA), chosen-ciphertext attack (CCA) or other cryptographic attacks. The lazy adversary and malicious adversary focus on breaking the verifiability of the proposed framework. They attempt to return the intermediate or forged results for saving the computational resources. There may exist mixed type of attackers. For example, in real applications, an adversary could be curious and malicious, indicating that the adversary intends to challenge both the confidentiality and the verifiability of the proposed framework.

4 The Proposed NMF Outsourcing Protocol

In this section, we provide the details concerning the five modules of our proposed NMF outsourcing framework illustrated in Fig. 1.

4.1 $\mathsf{KeyGen}(\mathsf{NMF}, 1^\lambda) \rightarrow K$

Given the security parameter λ, KeyGen randomly selects a key K from a specified key space $\{0,1\}^\lambda$.

4.2 $\mathsf{Encrypt}(\mathbf{V}, K) \rightarrow (\mathbf{V}', TS)$

In this module, the key K is utilized to drive a PRNG to generate a pseudorandom number sequence, which will be used to produce the encryption matrices to encrypt the data matrix $\mathbf{V} \in \mathbb{R}^{m\times n}$.

Before giving more details of generating the pseudorandom number sequences, we first describe the main

encryption procedure of our method. Assume we have a pseudorandom sequence $\mathbf{r} = \{\mathbf{r}_i\}_{i=1}^{t}$, where $\mathbf{r}_i \in \{0,1\}^{l_2}$ and $t \geq 2m + 2n - 2$. Then, based on the pseudorandom number sequence, we first create two randomly permuted 1-D sequences $\mathbf{p} = \{\mathbf{p}_i\}_{i=1}^{m}$ and $\mathbf{q} = \{\mathbf{q}_j\}_{j=1}^{n}$, which are permuted versions of the sequences $\{1, 2, \cdots, m\}$ and $\{1, 2, \cdots, n\}$, respectively by using the well-known Knuth Shuffle Algorithm [48]. Along with this process, we also generate their inverse sequences $\mathbf{p}^{-1} = \{\mathbf{p}_i^{-1}\}_{i=1}^{m}$ and $\mathbf{q}^{-1} = \{\mathbf{q}_j^{-1}\}_{j=1}^{n}$ such that: if $\mathbf{p}_i = j$ ($\mathbf{q}_i = j$), then $\mathbf{p}_j^{-1} = i$ ($\mathbf{q}_j^{-1} = i$). Note that the inverse sequences $\mathbf{p}^{-1}$ and $\mathbf{q}^{-1}$ will be used in the decryption phase. According to the security parameter λ, we randomly choose α_1 and α_2 such that $\alpha_1 < \lambda$ and $\alpha_2 < \lambda$. Then, we also randomly pick up two sets of random number $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{i=1}^{m} \leftarrow \mathcal{K}_\mu$ and $\boldsymbol{\nu} = \{\boldsymbol{\nu}_j\}_{j=1}^{n} \leftarrow \mathcal{K}_\nu$, where $\mathcal{K}_\mu = \{0,1\}^{\alpha_1}$ and $\mathcal{K}_\nu = \{0,1\}^{\alpha_2}$. The details are provided in the following Algorithm 1.

Algorithm 1 Generating random permuted 1-D sequences, their inverses and random number sets.

Input: Pseudorandom number sequence $\mathbf{r} = \{\mathbf{r}_i\}_{i=1}^{t}$, maximum of pseudorandom number $\mathsf{RMAX} = 2^{l_2} - 1$, key spaces $\mathcal{K}_\mu = \{0,1\}^{\alpha_1}$, $\mathcal{K}_\nu = \{0,1\}^{\alpha_2}$, and length of sequences m and n.

Output: $\{\mathbf{p}, \mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\nu}, \mathbf{p}^{-1}, \mathbf{q}^{-1}\}$.

Initialize $\mathbf{p} = \mathbf{p^{-1}} = \{1, 2, \ldots, m\}$ and $\mathbf{q} = \mathbf{q^{-1}} = \{1, 2, \ldots, n\}$.
for $i = 1$ to $m - 1$ **do**
  Generate a random integer j such that $i \leq j \leq m$, by setting $j = i + \lfloor \frac{\mathbf{r}_i}{\mathsf{RMAX}+1} \cdot (m - i + 1) \rfloor$.
  Swap $\mathbf{p}_i$ and $\mathbf{p}_j$.
  Set $\mathbf{p}_{\mathbf{p_i}}^{\mathbf{-1}} = i$.
  Generate a random number $\boldsymbol{\mu}_i$ from key space $\mathcal{K}_\mu$ by setting $\boldsymbol{\mu}_i = 1 + \lfloor \frac{\mathbf{r}_{i+m-1}}{\mathsf{RMAX}+1} \cdot (2^{\alpha_1} - 1) \rfloor$.
end for
Set $\mathbf{p}_{\mathbf{p_m}}^{\mathbf{-1}} = m$.
Set $\boldsymbol{\mu}_m = 1 + \lfloor \frac{\mathbf{r}_{2m-1}}{\mathsf{RMAX}+1} \cdot (2^{\alpha_1} - 1) \rfloor$.
for $i = 1$ to $n - 1$ **do**
  Generate a random integer j such that $i \leq j \leq n$, by setting $j = i + \lfloor \frac{\mathbf{r}_{i+2m-1}}{\mathsf{RMAX}+1} \cdot (n - i + 1) \rfloor$.
  Swap $\mathbf{q}_i$ and $\mathbf{q}_j$.
  Set $\mathbf{q}_{\mathbf{q_i}}^{\mathbf{-1}} = i$.
  Generate a random number $\boldsymbol{\nu}_i$ from key space $\mathcal{K}_\nu$ by setting $\boldsymbol{\nu}_i = 1 + \lfloor \frac{\mathbf{r}_{i+2m+n-2}}{\mathsf{RMAX}+1} \cdot (2^{\alpha_2} - 1) \rfloor$.
end for
Set $\mathbf{q}_{\mathbf{q_n}}^{\mathbf{-1}} = n$.
Set $\boldsymbol{\nu}_n = 1 + \lfloor \frac{\mathbf{r}_{2m+2n-2}}{\mathsf{RMAX}+1} \cdot (2^{\alpha_2} - 1) \rfloor$.
return $\{\mathbf{p}, \mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\nu}, \mathbf{p}^{-1}, \mathbf{q}^{-1}\}$.

We then produce the encryption matrices $\mathbf{P}, \mathbf{Q}, \mathbf{X}$ and $\mathbf{Y}$ by the sequences $\{\mathbf{p}, \mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\nu}\}$. Specifically, two permutation matrices $\mathbf{P} \in \mathbb{R}^{m \times m}$ and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ are produced as follows.

$$\begin{cases} \mathbf{P}_{i,j} = \delta(\mathbf{p}_i, j) \\ \mathbf{Q}_{i,j} = \delta(i, \mathbf{q}_j) \end{cases} \tag{9}$$

where $\delta(x, y)$ is the Kronecker delta function defined by

$$\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases} \tag{10}$$

Additionally, we also produce two diagonal matrices $\mathbf{X} = diag(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m)$ and $\mathbf{Y} = diag(\frac{1}{\boldsymbol{\nu}_1}, \frac{1}{\boldsymbol{\nu}_2}, \ldots, \frac{1}{\boldsymbol{\nu}_n})$, which can be utilized to adjust the magnitudes of the entries of the input matrix $\mathbf{V}$.

Upon having the encryption matrices $\mathbf{P} \in \mathbb{R}^{m \times m}$, $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{X} \in \mathbb{R}^{m \times m}$ and $\mathbf{Y} \in \mathbb{R}^{n \times n}$, we propose to encrypt the data matrix $\mathbf{V}$ as follows:

$$\mathbf{V}' = \mathbf{PXVYQ}. \tag{11}$$

It should be noted that a similar encryption strategy was also employed in [46].

When designing the encryption scheme, we also need to consider the incurred computational cost, as too complicated encryption and decryption would deviate the purpose of the outsourced NMF computation. In other words, we should ensure that the complexity of performing matrix encryption and decryption is much lower than that of solving the original NMF problem. To implement the matrix encryption in practice, we do not need to left and right multiply the matrices $\mathbf{P}, \mathbf{X}$ and $\mathbf{Y}, \mathbf{Q}$ explicitly; but rather we can employ row-wise and column-wise permutations and scaling specified by $\mathbf{p}$, $\mathbf{q}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ over the input matrix $\mathbf{V}$. Specifically, the entry $\mathbf{V}'_{i,j}$ of the encrypted matrix $\mathbf{V}'$ can be expressed as

$$\mathbf{V}'_{i,j} = \frac{\boldsymbol{\mu}_{\mathbf{p}_i}}{\boldsymbol{\nu}_{\mathbf{q}_j}} \mathbf{V}_{\mathbf{p}_i, \mathbf{q}_j} \tag{12}$$

Though straightforward, this could significantly reduce the computational complexity on the client side, compared with our initial implementations in the conference version [37].

We now explain how to utilize the key K to generate a pseudorandom sequence $\mathbf{r} = \{\mathbf{r}_i\}_{i=1}^{t}$, where $t \geq 2m+2n-2$. Specifically, Encrypt module randomly selects a initialization vector TS (e.g., time stamp) from a given space $\{0,1\}^d$, and combines it with the key K through a cryptographic hash function to produce a seed $s = hash(K||TS)$. Here TS serves as a randomizer and should take a new value for every encryption session. It is important to note that TS does not have to be kept secret, but must change for every session. Such a value is often referred to as nonce, which stands for number used once. A similar strategy employed in the hashed message authentication code [49] can be used to combine the key and the nonce $(K||TS)$. Here, we assume the availability of a cryptographic hash function, which is a pseudorandom one-way function with collision resistance (e.g., MD5, SHA1, SHA2). The security requirements of hash function are non-invertible and collision resistance. Specifically, a cryptographic hash function can be formally defined as below [50]:

Definition 1. *Let $hash : \{0,1\}^{l_0} \rightarrow \{0,1\}^{l_1}$ be a one-way hash function, where $l_0 > l_1$. $\forall x \in \{0,1\}^{l_0}$, $hash(x)$ is uniformly distributed in $\{0,1\}^{l_1}$. Also, for any inputs $x_1, x_2 \in \{0,1\}^{l_0}$ and $x_1 \neq x_2$, we have*

$$Pr[hash(x_1) = hash(x_2)] \leq \epsilon_h \tag{13}$$

where ϵ_h is a negligible function related to l_0 and l_1.

As shown in [51], ϵ_h is lower bounded by $1 - \exp(-2^{l_0 - l_1 - 1}(2^{l_0} - 1))$ and upper bounded by $2^{l_0 - l_1 - 1}(2^{l_0} - 1)$.

Clearly, for any two distinct inputs, the collision resistant hash function collides with probability at most ϵ_h.

Then we exploit a forward-secure stateful PRNG with seed s produced by the hash function to generate random key streams for the subsequent matrix encryption and decryption. Before giving the description of the forward-secure stateful PRNG, we first present the definition of the standard PRNG. Let $G : \{0,1\}^{l_1} \rightarrow \{0,1\}^{p(l_1)}, p(l_1) > l_1$, be a family of deterministic polynomial time computable functions. If its output is computationally indistinguishable from true randomness, then G is called a PRNG [52].

Compared with the standard PRNG described above, a forward-secure PRNG is a stateful PRNG [52]. It is an iterative and stateful algorithm, which produces some output bits as pseudorandom number and updates the seed at each invocation. It can be described as $G_{fs} : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_1} \times \{0,1\}^{l_2}$. At the period i, the input bit string $\mathbf{s}_i \in \{0,1\}^{l_1}$ is considered as the current state. The next state $\mathbf{s}_{i+1} \in \{0,1\}^{l_1}$ and the pseudorandom number $\mathbf{r}_i \in \{0,1\}^{l_2}$ are regarded as the output of the period i. A forward-secure PRNG can be easily constructed with a standard PRNG $G : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_1+l_2}$ by splitting its output into the next state and the actual output. Specifically, it can be done by setting $G_{fs}(\mathbf{s}_i) = G_0(\mathbf{s}_i)||G_1(\mathbf{s}_i)$, where $G_0(\mathbf{s}_i) = \mathbf{s}_{i+1} \in \{0,1\}^{l_1}$ and $G_1(\mathbf{s}_i) = \mathbf{r}_i \in \{0,1\}^{l_2}$. Then G_{fs} is a forward-secure PRNG with G_0 as the next state and G_1 as the pseudorandom output of the current state [52]. In the sequel, we use the notation $G^i_{fs}(\mathbf{s}_1)$ to denote the output pseudorandom random sequence $(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_i)$ up to the period i with the seed $\mathbf{s}_1$. Namely, we have

$$G^i_{fs}(\mathbf{s}_1) = G_1(\mathbf{s}_1)||G_1(\mathbf{s}_2)|| \cdots ||G_1(\mathbf{s}_i) \tag{14}$$

Then, ϵ_{prng}-forward secure PRNG can be formally defined as follows [52]:

Definition 2. *Let $G_{fs} : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_1} \times \{0,1\}^{l_2}$ be a stateful PRNG. If the initial state (seed) $\mathbf{s}_1$ is chosen uniformly at random from $\{0,1\}^{l_1}$, then at any period i, for any probabilistic polynomial time algorithm $\mathcal{D}$, which outputs 1 or 0 as a distinguisher, the probability of distinguishing the output from a truly random sequence is at most ϵ_{prng}. It can be expressed as*

$$\left| Pr[\mathcal{D}(G^i_{fs}(\mathbf{s}_1)) = 1] - Pr[\mathcal{D}(\pi) = 1] \right| \leq \epsilon_{prng} \tag{15}$$

where ϵ_{prng} is a negligible function related to the seed length l_1. The sequence π consists of a truly random sequence $(\mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}'_i)$, with each element randomly chosen from $\{0,1\}^{l_2}$.

The definition indicates that it is negligibly possible to distinguish the sequence output by a forward-secure PRNG from a truly random one.

According to the security analysis in [53], if $hash()$ is defined as **Definition** 1 and the stateful PRNG $G_{fs}()$ is defined as **Definition** 2, for the combined construction $G_{fs}(hash(x))$, the advantage of the distinguisher $\mathcal{D}$ making at most q queries is bounded by $\epsilon_{prng} + \epsilon_h \cdot q^2/2$, which can be written as

$$\left| \Pr[\mathcal{D}(G^i_{fs}(hash(x))) = 1] - \Pr[\mathcal{D}(\pi) = 1] \right| \leq \epsilon_{prng} + \epsilon_h \cdot q^2/2 \tag{16}$$

This indistinguishable property later will be used to prove the security of our proposed scheme.

At last, it should be emphasized that even for the same data matrix $\mathbf{V}$ encrypted in different sessions, the employed key streams could be different, due to the change of the nonce TS. In this sense, the key stream for matrix encryption/decryption can be regarded as being generated by a stream cipher. In fact, in our proposed scheme, the way of generating the key streams is exactly the same as that in standard stream ciphers, and the difference only lies in how to use these key streams for the actual encryption/decryption. In traditional stream ciphers, the produced key stream is typically XORed with the information sequence for encryption purpose; however, the key stream in our scheme is used to generate the permutation matrices $\mathbf{P}$, $\mathbf{Q}$ and the scaling matrices $\mathbf{X}$, $\mathbf{Y}$ for encrypting the data matrix.

4.3 NMF$(\mathbf{V}', TS, r, \epsilon) \rightarrow (\mathbf{W}', \mathbf{H}', TS)$

Upon receiving the encrypted matrix $\mathbf{V}'$, the dimension parameter r, and a stopping criterion ϵ, Charlie intends to factorize $\mathbf{V}'$ into $\mathbf{W}'$ and $\mathbf{H}'$ of sizes $m \times r$ and $r \times n$, respectively. Here, it should be noted that the nonce TS, as a part of ciphertext, does not involve in the factorization computation over the encrypted domain on the cloud side. It would be passed to the client Alice for the decryption operation. To perform the factorization, Charlie can employ one of the existing NMF algorithms specified by Alice, such as multiplicative update (MU) [40], alternating least squares (ALS) [45], and heuristic Algorithm [54], to perform the factorization task. As far as we know, all these existing NMF algorithms are based on iterations. Charlie is also asked to stop the iterative NMF algorithm when the following inequality holds

$$\frac{D(\mathbf{V}'|\mathbf{W}'^{(k)}\mathbf{H}'^{(k)})}{D(\mathbf{V}'|\mathbf{W}'^{(k+1)}\mathbf{H}'^{(k+1)})} - 1 \leq \epsilon \tag{17}$$

where $\mathbf{W}'^{(k)}$ and $\mathbf{H}'^{(k)}$ are the resulting factors in the kth iteration. Here, ϵ denotes the stopping criterion specified by the client Alice, and the discussion of its selection is deferred to the next subsection. After completing the NMF task, Charlie returns $\mathbf{W}' \triangleq \mathbf{W}'^{(k)}$ and $\mathbf{H}' \triangleq \mathbf{H}'^{(k)}$ as the results to Alice.

4.4 Verify$(\mathbf{V}', \mathbf{W}', \mathbf{H}', \epsilon) \rightarrow \mathbf{y} \triangleq (\mathbf{W}^*, \mathbf{H}^*) \cup \perp$

With the received $\mathbf{W}'$ and $\mathbf{H}'$ from the cloud, Alice first checks the validity of the results. Compared with the verification tasks in the existing matrix-related outsourcing frameworks [23], [35], [46], the challenges in our protocol stem from the fact that the factorization is only an *approximation* of the original matrix, rather than strict equality. Such approximation nature inherent in NMF computation makes the sampling-based verification approaches, e.g., [23], [35], [46], invalid. This is because the factorization performance can only be judged from the entire matrix, instead of from some selected entries.

Another straightforward verification strategy is to compute $\mathbf{W}'\mathbf{H}'$ and calculate the cost function value $D(\mathbf{V}'|\mathbf{W}'\mathbf{H}')$. As explained in Section 2, the minimized factorization distortion $D(\mathbf{V}'|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})$ highly depends on the matrix $\mathbf{V}'$ and the dimension parameter r, and hence,

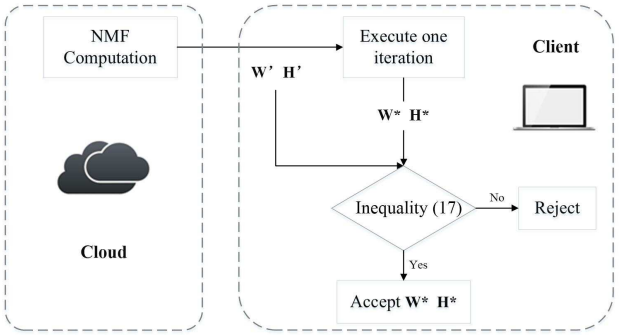


Fig. 2: Verification procedure.

it is difficult to judge the performance of the factorization from a single value of $D(\mathbf{V}'|\mathbf{W}'\mathbf{H}')$. For some practical matrices, even in the ideal case, i.e., iterating the algorithm for infinite number of times, the value of $D(\mathbf{V}'|\mathbf{W}'\mathbf{H}')$ is still quite large, especially when the entries of $\mathbf{V}'$ are taken from a large range. This explains why most of the existing NMF schemes do not use the value of $D(\mathbf{V}|\mathbf{W}^{(k)}\mathbf{H}^{(k)})$ as the stopping criterion [42], [45].

In this work, we propose a simple yet effective way to verify the results with small computational overhead by specifically exploiting the iterative nature of NMF computation. The idea behind our verification strategy is that when the relative decrease of the cost function, i.e., $\frac{D(\mathbf{V}'|\mathbf{W}'^{(k)}\mathbf{H}'^{(k)})}{D(\mathbf{V}'|\mathbf{W}'^{(k+1)}\mathbf{H}'^{(k+1)})} - 1$ is sufficiently small (less than a threshold ϵ), then we can determine that convergence has already been achieved. Specifically, the procedure of performing the efficient verification is illustrated in Fig. 2. By taking $\mathbf{W}'$ and $\mathbf{H}'$ received from Charlie as initial values, Alice further runs the NMF iterative algorithm for only *one* iteration to generate $\mathbf{W}^*$ and $\mathbf{H}^*$. Alice then verifies the results by checking the following inequality

$$\frac{D(\mathbf{V}'|\mathbf{W}'\mathbf{H}')}{D(\mathbf{V}'|\mathbf{W}^*\mathbf{H}^*)} - 1 \leq \epsilon \tag{18}$$

where ϵ is the stopping criterion. If this condition is satisfied, Alice accepts the results; otherwise, rejects them. It should be noted that the encrypted factorizations $\{\mathbf{W}', \mathbf{H}'\}$ and the verification result are all known for the adversary. In other words, the proposed Verify algorithm can be fully accessed by any adversary, or any public third party. It implies that our protocol is essentially publicly verifiable according to the definition given in [55]: any third party (possibly different from the client) can verify the correctness of the results returned from the server.

We now show that when ϵ is appropriately chosen to be sufficiently small, then when (18) holds, we can make sure that the results are acceptable, under the criterion defined in (7). To this end, we first define

$$D'^{(k)} \triangleq D(\mathbf{V}'|\mathbf{W}'^{(k)}\mathbf{H}'^{(k)}) \tag{19}$$

$$N \triangleq \min\left\{k \,\middle|\, \frac{D'^{(k)}}{D'^{(\infty)}} - 1 \leq \tau\right\} \tag{20}$$

where τ is the performance parameter given in (7). We further define

$$\bar{\epsilon} \triangleq \min_{k\in\{1,2,\cdots N-1\}} \left(\frac{D'^{(k)}}{D'^{(k+1)}} - 1\right) \tag{21}$$

Then ϵ is selected in a way such that

$$0 \leq \epsilon < \bar{\epsilon} \tag{22}$$

Under the above settings, we have the following Theorem.

Theorem 1. *If* $\frac{D'^{(k)}}{D'^{(k+1)}} - 1 \leq \epsilon$, *then* $\frac{D'^{(k+1)}}{D'^{(\infty)}} - 1 < \tau$.

Proof. Please refer to Appendix A for the detailed proof. □

This Theorem implies that, on the client side, Alice only needs to check the inequality (18), which ensures that the returned results are acceptable.

We conclude this subsection by a discussion on how to determine the parameter ϵ in (22) in practice. Clearly, we cannot set ϵ directly from (22) in an online fashion, because the value N in (20) depends on $D'^{(\infty)}$, which is not available before iterating for infinite number of times. To resolve this challenge, we here adopt an *offline* training method to determine ϵ. We build up a training set consisting of 100 matrices, among which 50 are random matrices of sizes ranging from 1000×1000 to 10000×8000. The remaining 50 matrices are formed by face images randomly selected from AR database[1], which contains 2600 face images of size 83×60 [56]. Each face image is vectorized to form a column of the data matrix $\mathbf{V}$, whose size ranges from 4980×800 to 4980×2600. We perform NMF for these 100 matrices with dimension parameters $r \in [20, 60]$. For each given performance parameter τ, we calculate $D'^{(k)}$, N, and $\bar{\epsilon}$ from (19), (20), and (21), respectively. In our training, $\mathbf{W}^{(\infty)}$ and $\mathbf{H}^{(\infty)}$ are set as the resulting factors after running the NMF iterative algorithms for 3×10^4 iterations. Let $\bar{\epsilon}(i, \tau)$ be the resulting $\bar{\epsilon}$ for the ith training matrix and for a given τ. We then determine ϵ as the lower bound of $\bar{\epsilon}(i, \tau)$ (subject to a small offset) for all the training matrices, namely

$$\epsilon(\tau) = \min_i \bar{\epsilon}(i, \tau) - \xi \tag{23}$$

where ξ is a small positive offset empirically set as 10^{-8}.

In Fig. 3, we plot the relationship between the performance parameter τ and ϵ obtained above. It can be seen that the value of ϵ becomes bigger with respect to the increasing τ.

The most time-consuming part of the above training process is the 3×10^4 iterations of NMF algorithm to obtain $\mathbf{W}^{(\infty)}$ and $\mathbf{H}^{(\infty)}$ for each given r (dimension parameter), especially when the training matrices are of large sizes. Overall, to train the 10 pairs of (ϵ, τ) by using our offline training method, as illustrated in Fig. 3, it takes around 30 hours on a MAC laptop with Intel Core i5 CPU and 8GB RAM. It should be noted that once the offline training process is completed, for a fixed τ, the corresponding ϵ can be obtained by performing simple look-up table operations.

1. http://www2.ece.ohio-state.edu/ aleix/ARdatabase.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2911282, IEEE Transactions on Services Computing

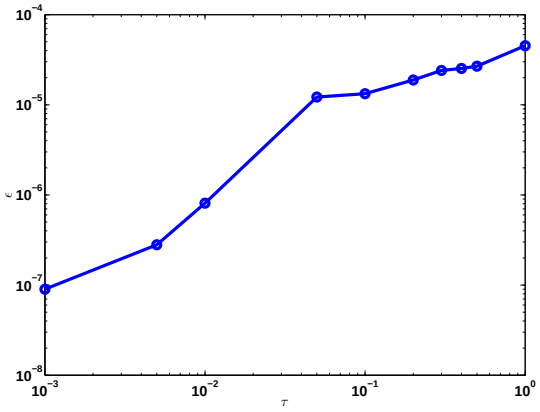


Fig. 3: The relationship between ϵ and τ.

4.5 Decrypt$(K, \mathbf{W}^*, \mathbf{H}^*, TS) \rightarrow (\mathbf{W}, \mathbf{H})$

If the results $\mathbf{W}'$ and $\mathbf{H}'$ successfully pass the verification stage, we then adopt $\mathbf{W}^*$ and $\mathbf{H}^*$ (rather then $\mathbf{W}'$ and $\mathbf{H}'$) as the encrypted factors. To derive the factors for the original matrix $\mathbf{V}$, we apply the following decryption operation

$$\begin{cases} \mathbf{W} = \mathbf{X}^{-1}\mathbf{P}^{-1}\mathbf{W}^* = \mathbf{X}^{-1}\mathbf{P}^T\mathbf{W}^* \\ \mathbf{H} = \mathbf{H}^*\mathbf{Q}^{-1}\mathbf{Y}^{-1} = \mathbf{H}^*\mathbf{Q}^T\mathbf{Y}^{-1} \end{cases} \tag{24}$$

Since the scaling matrices $\mathbf{X}$ and $\mathbf{Y}$ are diagonal, we can easily derive their inverse directly, $\mathbf{X}^{-1} = diag(\frac{1}{\boldsymbol{\mu}_1}, \frac{1}{\boldsymbol{\mu}_1}, \ldots, \frac{1}{\boldsymbol{\mu}_m})$ and $\mathbf{Y}^{-1} = diag(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_n)$.

In practice, we do not need to use the matrix multiplication to conduct the decryption. Rather, we can employ a technique similar to the encryption phase described in Section 4.2, by exploiting the permutation nature. Specifically,

$$\begin{cases} \mathbf{W}_{i,j} = \frac{1}{\boldsymbol{\mu}_i} \cdot \mathbf{W}^*_{\mathbf{p}_i^{-1},j} \\ \mathbf{H}_{i,j} = \boldsymbol{\nu}_j \cdot \mathbf{H}^*_{i,\mathbf{q}_j^{-1}} \end{cases} \tag{25}$$

where $\mathbf{p}^{-1} = \{\mathbf{p}_i^{-1}\}_{i=1}^m$ and $\mathbf{q}^{-1} = \{\mathbf{q}_j^{-1}\}_{j=1}^n$ denote the inverse sequences of $\mathbf{p}$ and $\mathbf{q}$, respectively. Similar to the Encrypt module, we can employ the key K and the nonce TS to generate the same pseudorandom number sequence $\mathbf{r} = \{\mathbf{r}_i\}_{i=1}^t$, which is then used to produce the inverse sequences $\mathbf{p}^{-1} = \{\mathbf{p}_i^{-1}\}_{i=1}^m$ and $\mathbf{q}^{-1} = \{\mathbf{q}_j^{-1}\}_{j=1}^n$ by Algorithm 1.

5 Security Experiments

Before giving the security and verifiability analyses, we first present some important definitions and then describe the security experiments. We use notation $x \xleftarrow{R} \mathcal{S}$ to denote that x is chosen uniformly at random from the set $\mathcal{S}$. A function $f(n)$ is said to be negligible if for sufficiently large n, its value is smaller than the inverse of any polynomial $poly(n)$.

Definition 3. Restricted Matrix Permutation-scaling set: *Let $\mathcal{B} = \{\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_M\}$, where each $\mathbf{B}_i \in \mathbb{R}^{m\times n}$. For a given matrix $\mathbf{V} \in \mathbb{R}^{m\times n}$, we call $\mathcal{B}$ a restricted matrix permutation-scaling set with respect to $\mathbf{V}$ if for any $\mathbf{B}_i \in \mathcal{B}$, there exist permutation matrices $\mathbf{P} \in \mathbb{R}^{m\times m}$, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and diagonal matrices $\mathbf{X} \in \mathbb{R}^{m\times m}$, $\mathbf{Y} \in \mathbb{R}^{n\times n}$ such that $\mathbf{B}_i = \mathbf{PXVYQ}$.*

In other words, the matrix $\mathbf{V}$ can be regarded as a generator to produce all the elements in $\mathcal{B}$ by left and right multiplying a permutation matrix and a diagonal matrix of appropriate sizes, respectively.

Having the intuitions above, we now define the interactive security experiment to evaluate the data secrecy in our secure and verifiable outsourcing scheme. As can be seen from (11), our encryption scheme shuffles the positions of the matrix entries, and also changes their values. Therefore, the following security game is defined under the restricted matrix permutation-scaling set $\mathcal{B}$, in which we consider the impact of the data value and the entry position on the privacy experiment. In the experiment, the adversary $\mathcal{A}$ is allowed to query Encrypt on inputs $\mathbf{V}_i$ of its choice a polynomial number of times, and obtain the corresponding encrypted matrix $\mathbf{V}'_i$ and the nonce TS_i. Essentially, this is a type of CPA. Eventually, $\mathcal{A}$ outputs two matrices $\mathbf{V}^{(0)}$ and $\mathbf{V}^{(1)}$ on which it would like to be challenged, from any restricted matrix permutation-scaling set $\mathcal{B}$. A random bit b is drawn. $\mathcal{A}$ is then given $\mathbf{V}'^{(b)}$, which is the encrypted version of $\mathbf{V}^{(b)}$. Finally, the adversary $\mathcal{A}$ outputs its guess for b and wins if it could correctly determine the value of b, i.e., distinguish between $\mathbf{V}^{(0)}$ and $\mathbf{V}^{(1)}$.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CPA}}(\mathsf{NMF}, \lambda)$
$K \leftarrow \mathsf{KeyGen}(\mathsf{NMF}, 1^\lambda)$, $\mathbf{V}_1 \xleftarrow{R} \mathcal{B}$
$(\mathbf{V}'_1, TS_1) \leftarrow \mathsf{Encrypt}(\mathbf{V}_1, K)$
for $i = 2$ **to** q do
  $\mathbf{V}_i \leftarrow \mathcal{A}(\mathbf{V}'_1, TS_1, \cdots, \mathbf{V}'_{i-1}, TS_{i-1})$
  $(\mathbf{V}'_i, TS_i) \leftarrow \mathsf{Encrypt}(\mathbf{V}_i, K)$
$\{\mathbf{V}^{(0)} \in \mathcal{B}, \mathbf{V}^{(1)} \in \mathcal{B}\} \leftarrow \mathcal{A}(\mathbf{V}'_1, TS_1, \cdots, \mathbf{V}'_q, TS_q)$
$b \xleftarrow{R} \{0, 1\}$
$(\mathbf{V}'^{(b)}, TS^{(b)}) \leftarrow \mathsf{Encrypt}(\mathbf{V}^{(b)}, K)$
$\hat{b} \leftarrow \mathcal{A}(\mathbf{V}'_1, TS_1 \cdots, \mathbf{V}'_q, TS_q, \mathbf{V}'^{(b)}, TS^{(b)})$
if $\hat{b} = b$ return 1, else return 0

For any $\lambda \in \mathbb{N}$, we define the advantage of an adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries in the above security game as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{CPA}}(\mathsf{NMF}, q, \lambda) = \left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CPA}}(\mathsf{NMF}, \lambda) = 1] - 1/2\right| \tag{26}$$

Definition 4. *We say that a verifiable outsourcing scheme of NMF ensures data secrecy against CPA if for any adversary $\mathcal{A}$ and λ, it holds that the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{CPA}}(\mathsf{NMF}, q, \lambda)$ is negligible.*

In addition to CPA, we also consider the security experiment under CCA. According to [57], there are two types of CCA model: non-adaptive CCA (CCA1) [58], and adaptive CCA (CCA2) [59]. Under CCA1 model, the adversary can choose a polynomial number of ciphertexts and use the decryption oracle to obtain the corresponding plaintexts before a challenge ciphertext is given. In other words, the adversary can decrypt arbitrary messages before obtaining the challenge ciphertext. Under CCA2 model, the adversary can adaptively choose the ciphertexts to decrypt before and after a challenge ciphertext is given. The restriction is that the adversary cannot use the decryption oracle to decrypt the challenge ciphertext directly. Compared with CCA1 model, the adversary in CCA2 obtains more advantages

(decryption oracle) after the challenge ciphertext is given.

We now first focus on the security experiment under CCA1, and defer the discussion on CCA2 in the next section. Under CCA1, the adversary is granted oracle access to the decryption module, where $\mathcal{O}_{\mathsf{Dec}}(K, \mathbf{V}'_i, TS_i)$ runs the decryption module $(\mathbf{V}_i) \leftarrow \mathsf{Dec}(K, \mathbf{V}'_i, TS_i)$. Then, $\mathcal{A}$ outputs two matrices $\mathbf{V}^{(0)}$ and $\mathbf{V}^{(1)}$ on which it would like to be challenged, from the restricted matrix permutation-scaling set $\mathcal{B}$. A random bit b is drawn. $\mathcal{A}$ is then given $\mathbf{V}'^{(b)}$, which is the encrypted version of $\mathbf{V}^{(b)}$. Finally, the adversary $\mathcal{A}$ outputs its guess for b and wins if it could correctly determine the value of b, i.e., distinguish between $\mathbf{V}^{(0)}$ and $\mathbf{V}^{(1)}$.

Experiment $\mathbf{Exp}^{\mathsf{CCA1}}_{\mathcal{A}}(\mathsf{NMF}, \lambda)$

$K \leftarrow \mathsf{KeyGen}(\mathsf{NMF}, 1^\lambda)$, $\mathbf{V}_1 \xleftarrow{R} \mathcal{B}$
$(\mathbf{V}'_1, TS_1) \leftarrow \mathsf{Encrypt}(\mathbf{V}_1, K)$
$\mathbf{V}_1 \leftarrow \mathsf{Dec}(K, \mathbf{V}'_1, TS_1)$
for $i = 2$ **to** q do
$(\mathbf{V}'_i, TS_i) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}}(\cdot,\cdot,\cdot)}(\mathbf{V}'_1, TS_1, \cdots, \mathbf{V}'_{i-1}, TS_{i-1})$
$\mathbf{V}_i \leftarrow \mathsf{Dec}(K, \mathbf{V}'_i, TS_i)$
$(\mathbf{V}^{(0)} \in \mathcal{B}, \mathbf{V}^{(1)} \in \mathcal{B}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}}(\cdot,\cdot,\cdot)}(\mathbf{V}'_1, TS_1, \cdots, \mathbf{V}'_q, TS_q)$
$b \xleftarrow{R} \{0, 1\}$
$(\mathbf{V}'^{(b)}, TS^{(b)}) \leftarrow \mathsf{Encrypt}(\mathbf{V}^{(b)}, K)$
$\hat{b} \leftarrow \mathcal{A}(\mathbf{V}'_1, TS_1, \mathbf{V}_1, \cdots, \mathbf{V}'_q, TS_q, \mathbf{V}_q, \mathbf{V}'^{(b)}, TS^{(b)})$
if $\hat{b} = b$ return 1, else return 0

For any $\lambda \in \mathbb{N}$, we define the advantage of an adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries in the above security game (under CCA1 model) as

$$\mathbf{Adv}^{\mathsf{CCA1}}_{\mathcal{A}}(\mathsf{NMF}, q, \lambda) = \left| \Pr[\mathbf{Exp}^{\mathsf{CCA1}}_{\mathcal{A}}(\mathsf{NMF}, \lambda) = 1] - 1/2 \right| \quad (27)$$

Definition 5. *We say that a verifiable outsourcing scheme of NMF ensures data secrecy against CCA1 if for any adversary $\mathcal{A}$ and λ, it holds that the advantage $\mathbf{Adv}^{\mathsf{CCA1}}_{\mathcal{A}}(\mathsf{NMF}, q, \lambda)$ is negligible.*

In addition, to formulate the verifiability of our verifiable computation scheme, we can similarly define an interactive security experiment described next. It should be noted that this experiment focuses on the verifiability of the returned results, rather than the data secrecy. In the experiment, the adversary $\mathcal{A}$ is allowed to query KeyGen and $\mathsf{Encrypt}$ on inputs $\mathbf{V}_i$ of its choice a polynomial number of times, obtain the corresponding encrypted matrix $\mathbf{V}'_i$. It is also granted oracle access to the public Verify algorithm, where $\mathcal{O}_{\mathsf{Verify}}(\mathbf{V}'_i, \mathbf{W}'_i, \mathbf{H}'_i, \epsilon_i)$ runs the verification algorithm $\mathbf{y} \leftarrow \mathsf{Verify}(\mathbf{V}'_i, \mathbf{W}'_i, \mathbf{H}'_i, \epsilon_i)$. Eventually, $\mathcal{A}$ outputs $\mathbf{V}^*$ on which it would like to be challenged, obtains the encrypted $\mathbf{V}'^*$, and produces a result $(\bar{\mathbf{W}}', \bar{\mathbf{H}}')$. The adversary wins if $(\bar{\mathbf{W}}', \bar{\mathbf{H}}')$ makes the inequality (7) invalid, and the verification algorithm $\mathsf{Verify}(\mathbf{V}'^*, \bar{\mathbf{W}}', \bar{\mathbf{H}}', \epsilon)$ does not generate an error $\perp$.

For any $\lambda \in \mathbb{N}$, we define the advantage of an adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries in the above security game as

$$\mathbf{Adv}^{\mathsf{Ver}}_{\mathcal{A}}(\mathsf{NMF}, q, \lambda) = \Pr[\mathbf{Exp}^{\mathsf{Ver}}_{\mathcal{A}}(\mathsf{NMF}, \lambda) = 1] \quad (28)$$

Definition 6. *A verifiable outsourcing scheme of large-scale NMF is secure for any adversary $\mathcal{A}$ and any λ, if $\mathbf{Adv}^{\mathsf{Ver}}_{\mathcal{A}}(\mathsf{NMF}, q, \lambda)$ is negligible in λ.*

Experiment $\mathbf{Exp}^{\mathsf{Ver}}_{\mathcal{A}}(\mathsf{NMF}, \lambda)$

$K \leftarrow \mathsf{KeyGen}(\mathsf{NMF}, 1^\lambda)$
$\mathbf{V}_1 \xleftarrow{R} \mathcal{B}$, $(r_1, \epsilon_1) \leftarrow \mathcal{A}(\mathbf{V}_1)$
$(\mathbf{V}'_1, TS_1) \leftarrow \mathsf{Encrypt}(\mathbf{V}_1, K)$
$(\mathbf{W}'_1, \mathbf{H}'_1, TS_1) \leftarrow \mathsf{NMF}(\mathbf{V}'_1, TS_1, r_1, \epsilon_1)$
$\mathbf{y}_1 \leftarrow \mathsf{Verify}(\mathbf{V}'_1, \mathbf{W}'_1, \mathbf{H}'_1, \epsilon_1)$
for $i = 2$ **to** q do
$(\mathbf{V}_i, r_i, \epsilon_i) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Verify}}(\cdot,\cdot,\cdot,\cdot)}(\mathbf{V}'_1, TS_1, \mathbf{W}'_1, \mathbf{H}'_1, \cdots, \mathbf{V}'_{i-1}, TS_{i-1}, \mathbf{W}'_{i-1}, \mathbf{H}'_{i-1})$
$(\mathbf{V}'_i, TS_i) \leftarrow \mathsf{Encrypt}(\mathbf{V}_i, K)$
$(\mathbf{W}'_i, \mathbf{H}'_i, TS_i) \leftarrow \mathsf{NMF}(\mathbf{V}'_i, TS_i, r_i, \epsilon_i)$
$\mathbf{y}_i \leftarrow \mathsf{Verify}(\mathbf{V}'_i, \mathbf{W}'_i, \mathbf{H}'_i, \epsilon_i)$
$(\mathbf{V}^*, r, \epsilon) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Verify}}(\cdot,\cdot,\cdot,\cdot)}(\mathbf{V}'_1, TS_1, \mathbf{W}'_1, \mathbf{H}'_1, \cdots, \mathbf{V}'_q, TS_q, \mathbf{W}'_q, \mathbf{H}'_q)$
$(\mathbf{V}'^*, TS^*) \leftarrow \mathsf{Encrypt}(\mathbf{V}^*, K)$
$(\bar{\mathbf{W}}', \bar{\mathbf{H}}') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Verify}}(\cdot,\cdot,\cdot,\cdot)}(\mathbf{V}'_1, TS_1, \mathbf{W}'_1, \mathbf{H}'_1, \mathbf{y}_1, \cdots, \mathbf{V}'_q, TS_q, \mathbf{W}'_q, \mathbf{H}'_q, \mathbf{y}_q, \mathbf{V}'^*)$
$\mathbf{y} \leftarrow \mathsf{Verify}(\mathbf{V}'^*, \bar{\mathbf{W}}', \bar{\mathbf{H}}', \epsilon)$
if $\mathbf{y}$ makes the inequality (7) invalid and $\mathbf{y} \neq \perp$, return 1
else return 0

6 Security, Correctness, Verifiability and Complexity Analyses

6.1 Security Analysis

To formulate the data secrecy in our proposed protocol, we proceed with security games and analyze the adversary $\mathcal{A}$'s advantage in winning the experiment.

Theorem 2. *The proposed secure and verifiable outsourcing scheme of large-scale NMF holds indistinguishability under CPA (IND-CPA) privacy according to the* **Definition** 4.

Proof. Please refer to Appendix B for the detailed proof. □

Theorem 3. *The proposed secure and verifiable outsourcing scheme of large-scale NMF holds indistinguishability under CCA1 (IND-CCA1) privacy according to the* **Definition** 5.

Proof. Please refer to Appendix C for the detailed proof. □

Remark: When designing the above security experiments and conducting the proofs, the matrices to be challenged are only picked from the restricted matrix permutation-scaling set $\mathcal{B}$. In other words, we only claim that our proposed scheme is secure **under the restricted matrix permutation-scaling set**, in which all the matrices have the same number of zeros. The adoption of the restricted matrix permutation-scaling set eliminates the straightforward attack strategy by exploiting the matrix statistics, e.g., choosing two challenged matrices with different number of zeros. Meanwhile, the development of a more secure scheme that can resist all statistical attacks and still enjoy efficient processing in the cloud is one of our future works.

In addition, in the above proofs, we do not require the input data matrix to be sparse. This implies that the proposed scheme is CPA-secure and CCA1-secure, even in the case of sparse input matrix. Nevertheless, our scheme is not secure against CCA2 because it is a malleable scheme.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2911282, IEEE Transactions on Services Computing

Specifically, it is possible to transform the challenge ciphertext into another ciphertext and decrypt it directly. Hence, the adversary can generate and decrypt a specific ciphertext which is related to the challenge ciphertext, so that the adversary can distinguish the challenge ciphertext. How to make the NMF outsourcing framework secure against CCA2 is still a challenging task.

6.2 Correctness Analysis

According to the second design goal, we should ensure that the factors $\mathbf{W}$ and $\mathbf{H}$ obtained in Section 2 are acceptable according to the criterion defined in (7). It should be emphasized that the judgement of the acceptance of factorization performance is conducted in the plaintext domain, rather than the ciphertext domain in which the NMF is carried out. To show the correctness of the outsourced NMF protocol, it suffices to prove the following Theorem.

Theorem 4. *Letting* $\mathbf{W}$ *and* $\mathbf{H}$ *be the two factors for* $\mathbf{V}$ *obtained using the method presented in Section 4, we have*

$$\frac{D(\mathbf{V}|\mathbf{WH})}{D(\mathbf{V}|\mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)})} - 1 \leq \tau \tag{29}$$

Proof. Please refer to Appendix D for the detailed proof. □

6.3 Verifiability Analysis

Theorem 5. *The proposed secure and verifiable outsourcing scheme of large-scale NMF is secure according to the* **Definition** 6 *in the presence of malicious and lazy adversaries.*

Proof. Please refer to Appendix E for the detailed proof. □

6.4 Complexity Analysis

Now, we give the analysis of the complexity on both client side and cloud side. On the client side, the complexity mainly comes from the tasks of performing the encryption/decryption and doing the verification. In the encryption/decryption phase, if we naively implement it by using matrix multiplications, then the complexity involved is of order $O(t^2)$, where $t = \max(m, n)$. However, if we adopt the permutation-based implementation, as explained in Section 4.2, the complexity can be reduced to $O(mn)$. In terms of the verification where we need to perform one iteration, the complexity is of order $O(mnr)$ [44].

On the cloud side, the complexity is incurred primarily by the iterative algorithm for solving the NMF problem over the encrypted domain. We now analyze the update rules in both the plaintext and the encrypted domain. For simplicity, we use the update rule of $\mathbf{W}$ in (5) for example. The derivation below can also be extended to the case of $\mathbf{H}$. Here, we utilize $\hat{\mathbf{W}} \triangleq \mathbf{W}^{(k)}$ and $\hat{\mathbf{H}} \triangleq \mathbf{H}^{(k)}$ in the plaintext domain. Specifically, according to the decryption rules $\mathbf{W}' = \mathbf{PXW}$, $\mathbf{H}' = \mathbf{HYQ}$ and the update rule (5) in the plaintext domain, we can write the update rule in the encrypted domain as

$$(\mathbf{PXW}^{(k+1)})_{i,a} \leftarrow (\mathbf{PX}\hat{\mathbf{W}})_{i,a} \cdot \theta \tag{30}$$

$$\theta = \frac{\left[((\mathbf{PX}\hat{\mathbf{W}}\hat{\mathbf{H}}\mathbf{YQ})^{\langle -2\rangle} \odot (\mathbf{PXVYQ}))(\hat{\mathbf{H}}\mathbf{YQ})^T\right]_{i,a}}{\left[(\mathbf{PX}\hat{\mathbf{W}}\hat{\mathbf{H}}\mathbf{YQ})^{\langle -1\rangle}(\hat{\mathbf{H}}\mathbf{YQ})^T\right]_{i,a}} \tag{31}$$

where $\mathbf{P}$, $\mathbf{Q}$ are permutation matrices and $\mathbf{X}, \mathbf{Y}$ are diagonal matrices used in the encryption/decryption phase. Noticing that $\mathbf{P}^{\langle n\rangle} = \mathbf{P}$, $\mathbf{Q}^{\langle n\rangle} = \mathbf{Q}$ and $\mathbf{QQ}^T = \mathbf{I}$, we have

$$\begin{aligned}
\theta &= \frac{\left[((\mathbf{PX}^{\langle -2\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle}\mathbf{Y}^{\langle -2\rangle}\mathbf{Q}) \odot (\mathbf{PXVYQ}))\mathbf{Q}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{PX}^{\langle -1\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\mathbf{Y}^{\langle -1\rangle}\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}} \\
&= \frac{\left[\mathbf{P}((\mathbf{X}^{\langle -2\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle}\mathbf{Y}^{\langle -2\rangle}) \odot (\mathbf{XVY}))\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{PX}^{\langle -1\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\mathbf{Y}^{\langle -1\rangle}\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}} \\
&= \frac{\left[\mathbf{P}(\mathbf{X}^{\langle -2\rangle}\mathbf{X}((\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle} \odot \mathbf{V})\mathbf{Y}^{\langle -2\rangle}\mathbf{Y})\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{PX}^{\langle -1\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\mathbf{Y}^{\langle -1\rangle}\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}} \\
&= \frac{\left[\mathbf{PX}^{\langle -1\rangle}((\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle} \odot \mathbf{V})\mathbf{Y}^{\langle -1\rangle}\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{PX}^{\langle -1\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\mathbf{Y}^{\langle -1\rangle}\mathbf{QQ}^T\mathbf{Y}\hat{\mathbf{H}}^T\right]_{i,a}} \\
&= \frac{\left[\mathbf{PX}^{\langle -1\rangle}((\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle} \odot \mathbf{V})\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{PX}^{\langle -1\rangle}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\hat{\mathbf{H}}^T\right]_{i,a}} \\
&= \frac{\left[\mathbf{P}((\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -2\rangle} \odot \mathbf{V})\hat{\mathbf{H}}^T\right]_{i,a}}{\left[\mathbf{P}(\hat{\mathbf{W}}\hat{\mathbf{H}})^{\langle -1\rangle}\hat{\mathbf{H}}^T\right]_{i,a}}
\end{aligned}$$

where the second equality holds from the fact that the permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ change the locations of entries but not their values. Also, the fifth equality holds from the property of diagonal matrices $\mathbf{X}^{\langle -1\rangle}\mathbf{X} = \mathbf{I}$ and $\mathbf{Y}^{\langle -1\rangle}\mathbf{Y} = \mathbf{I}$. As can be seen from above and the update rule (5), the factor θ in the encrypted domain remains the same as that in the plaintext domain, subject to row permutation caused by $\mathbf{P}$. The above derivation guarantees the correctness and the efficiency of carrying out NMF in the encrypted domain. As demonstrated by Arora *et. al* in [43], the complexity of performing the NMF computation in the plaintext domain is of order $O((mn)^{r^2 2^r})$, and so is the complexity in the encrypted domain.

Therefore, the complexity on the client side in the proposed NMF outsourcing framework is much lower than that involved in solving the original NMF problem. In addition, the efficiency on the cloud side is expected to be well preserved. The experimental results to be given in the next section will validate the superiority of our proposed scheme.

7 Experimental Results

In this section, we present the experimental results to evaluate the system performance. We implement our protocol with MATLAB 2014b, and all the following tests are conducted on a MAC laptop with Intel Core i5 CPU and 8GB RAM.

We first investigate the performance on factorizing synthetic nonnegative matrices. We construct random nonnegative matrices $\mathbf{V}$ of various sizes ranging from 1000×1000 to 20000×16000, using a similar technique in [39]. Specifically, the synthetic data matrix $\mathbf{V}$ is produced by $\mathbf{V} = \mathbf{CD}$, where the ground truth factors $\mathbf{C} \in \mathbb{R}^{m \times r}$ and $\mathbf{D} \in \mathbb{R}^{r \times n}$ are generated as the absolute value of standard Gaussian noise [39].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2911282, IEEE Transactions on Services Computing

TABLE 1: Comparison of Execution Time on Synthetic Matrices (in seconds)

m	r	n	T_{original}	T_{client}	T_{cloud}	$T_{\text{original}}/T_{\text{client}}$	$T_{\text{original}}/T_{\text{cloud}}$
1000	40	1000	59.91	0.6901	59.45	86.81	1.0077
1000	60	1000	72.07	0.7577	70.61	95.12	1.0207
3000	40	2400	342.94	2.9205	352.62	117.43	0.9725
3000	60	2400	379.47	3.0689	383.67	123.65	0.9891
6000	40	4800	1431.71	11.0653	1429.03	129.39	1.0019
6000	60	4800	1567.74	11.6499	1579.87	134.57	0.9923
10000	40	8000	3806.63	25.6522	3820.48	148.39	0.9964
10000	60	8000	4253.82	26.0518	4266.90	163.28	0.9969
20000	40	16000	49965.87	202.8413	50688.35	246.33	0.9857
20000	60	16000	53917.13	209.5975	52916.04	257.24	1.0189

TABLE 2: Comparison of Execution Time on Face and PET Image Databases (in seconds)

Database	m	r	n	T_{original}	T_{client}	T_{cloud}	$T_{\text{original}}/T_{\text{client}}$	$T_{\text{original}}/T_{\text{cloud}}$
CBCL	361	25	2429	107.59	0.7256	109.89	148.28	0.9791
CBCL	361	40	2429	147.73	0.7737	145.29	190.94	1.0168
Olivetti	4096	25	400	242.24	1.0380	246.02	233.37	0.9846
Olivetti	4096	40	400	339.18	1.2532	335.17	270.66	1.0119
ORL	10304	25	400	500.62	1.6262	507.76	307.85	0.9859
ORL	10304	40	400	563.39	1.6593	553.36	339.53	1.0181
PET	573440	3	40	1009.87	6.4457	994.82	156.67	1.0151
PET	573440	5	40	1202.36	6.6524	1225.44	180.74	0.9812

This ensures that the matrix $\mathbf{V}$ can be exactly factorized. We set the performance parameter $\tau = 10^{-1}$, and consequently can obtain the stopping criterion $\epsilon = 1.3267 \times 10^{-5}$ according to the training results presented in Section 4.4. In Table 1, we show the comparison regarding the computational complexity. Here,

- T_{original} denotes the execution time for the client to solve the original NMF locally (without encryption).
- T_{cloud} represents the execution time for the cloud to complete NMF computation over the encrypted $\mathbf{V}'$.
- T_{client} is the total execution time for the client to perform matrix encryption, verification and decryption.

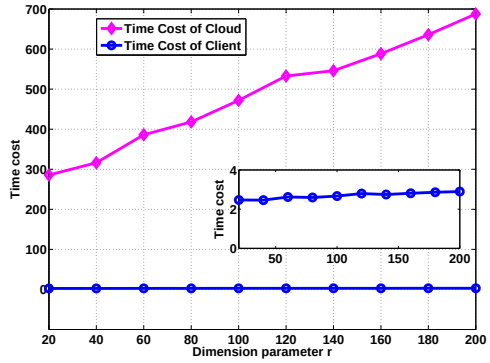
T_{original} $/T_{\text{client}}$ can be considered as the client resource advantage, which measures the performance gain of the client. Generally, this value should be as large as possible. Similarly, T_{original} $/T_{\text{cloud}}$ measures the efficiency of outsourcing protocol, which should be close to 1, according to the last design goal. As can be seen from Table 1, the client resource advantage is up to 257.24, which is quite significant. It can also be observed that the client resource advantage becomes more remarkable with the increasing size of the matrix to be factorized. This is because T_{client} is of order $O(mnr)$, while T_{original} is of order $O((mn)^{r^2 2^r})$. This implies that the proposed protocol is especially valuable for handling large-scale matrices, which are very common in big data era. Furthermore, from the rightmost column of Table 1, we can notice that all the values of $T_{\text{original}}/T_{\text{cloud}}$ are quite close to 1, meaning that the efficiency on the cloud side is well maintained.

We then evaluate the system performance by using real-world data. We consider the factorization tasks for face images, which were also studied in [41], and for hyperspectral positron emission tomography (PET) images, as investigated in [60]. We test on three face image databases: 1) Center for Biological and Computational Learning (CBCL) face database[2], which consists of 2429 face images of size 19×19; 2) Olivetti face database[3], which includes 400 face images of size 64×64, and 3) ORL face image database[4], which is composed of 400 face images of size 112×92. For a given face image database, each image is vectorized as a column, which eventually forms a nonnegative matrix $\mathbf{V}$ to be factorized. The PET image[5] consists of 40 frames, in which each frame is composed of 35 slices of 128×128 3D image. Each frame is vectorized as a column, resulting in a nonnegative matrix $\mathbf{V}$ of size 573440×40. The experimental results over these three face databases and PET database are tabulated in Table 2. As can be seen, the resource advantage on the client side can be as large as 339.53, and the efficiency on the cloud side is well preserved.

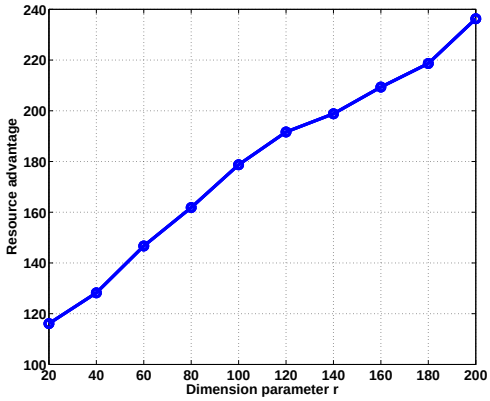
The dimension parameter r plays an important role in the NMF task, balancing the data fitting and the model complexity [38]. To measure the impact of different r on the system performance, we fix the size of $\mathbf{V}$ to be 3000×2400; but vary the dimension parameter r from 20 to 200. Specifically, we generate 100 random matrices of size 3000×2400 by using the aforementioned technique. For each given r, we test the factorization performance over these 100 random matrices, and record the average time costs on the client side and the cloud side. The results are demonstrated in Fig. 4 (a), where each point is the average time cost over all these 100 random matrices. As can be observed, the computational complexity on the cloud side grows almost linearly with respect to the increasing r, because the total number of entries of $\mathbf{W}$ and $\mathbf{H}$ is $r(m+n)$. On the client side, the computational complexity also increases when r becomes larger, primarily due to the increment of the decryption cost; but the resource consumption is much slower than

2. http://cbcl.mit.edu/software-datasets/FaceData2.html
3. http://www.cs.nyu.edu/ roweis/data.html
4. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
5. http://cogsys.imm.dtu.dk/toolbox/nmf

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2911282, IEEE Transactions on Services Computing

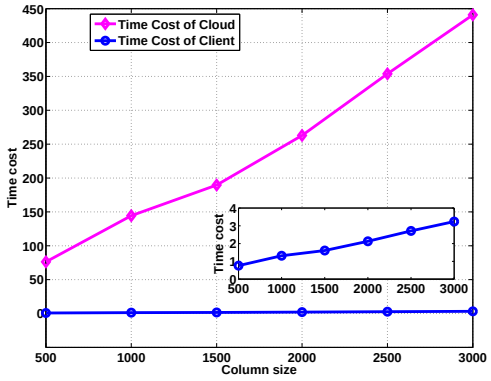


(a)

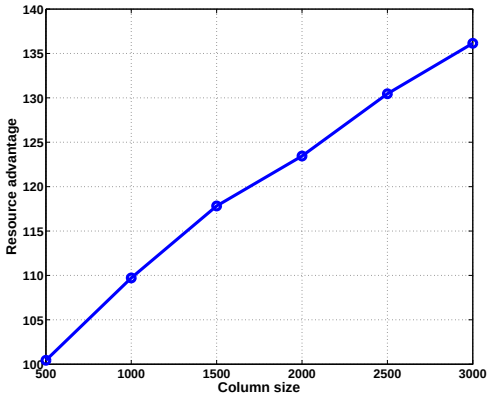


(b)

Fig. 4: (a) Time costs on the cloud side and on the client side, with respect to the dimension parameter r. (b) Client resource advantage v.s. dimension parameter r.



(a)



(b)

Fig. 5: (a) Time costs on the cloud side and on the client side, with respect to the column size. (b) Client resource advantage v.s. the column size.

that on the cloud side. This implies that the client resource advantage, i.e., $T_{\text{original}}/T_{\text{client}}$, improves with respect to the increasing r, as illustrated in Fig. 4 (b).

Furthermore, we study the influence of the matrix size on the system performance. To simplify the task, we fix the row size to be 3000 and the dimension parameter $r = 40$; but vary the column size from 500 to 3000. As can be seen from Fig. 5 (a), the computational complexity on both the cloud side and the client side is increased; but the incremental speed of the cloud side is much faster. This phenomenon, in turn, suggests that the performance gain on the client side boosts with the increasing column size, as demonstrated in Fig. 5 (b).

We also show how the performance parameter τ influences the client resource advantage. To this end, we fix the size of the input matrix to be 3000×2400 and the dimension parameter $r = 40$, while varying τ from 10^{-3} to 1. Generally, a smaller τ implies a higher accuracy of the factorization task, which requires the cloud server to perform more iterations. It should also be noted that τ is not used on the client side, suggesting that the complexity of the client side is independent with τ. In Fig. 6, we give the relationship between the client resource advantage and τ. It can be observed that the gain on resource advantage

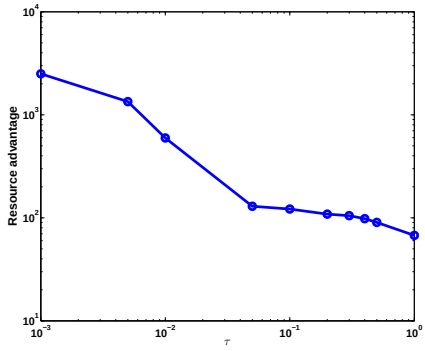
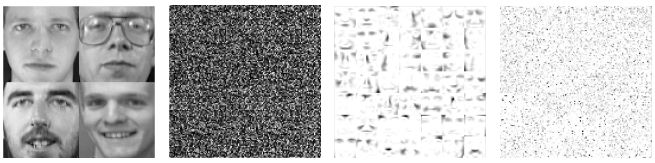


Fig. 6: The client resource advantage under different choices of τ.

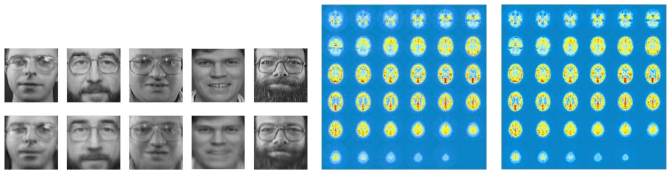
improves with the decreasing τ, which coincides with our analysis above.

We now demonstrate the visual comparison of the original images and the original dictionaries (i.e., the factor $\mathbf{W}$), together with their encrypted counterparts in Fig. 7. To assist the illustration, we here use face images selected from `Olivetti` database. As can be observed from Fig. 7 (a)-(b),



(a) (b) (c) (d)

Fig. 7: Original face images, output dictionary and their encrypted versions. (a) Original Face Images, (b) Encrypted Face Images, (c) Original Dictionary, (d) Encrypted Dictionary



(a) Face Images and reconstructed versions (b) PET images (c) Reconstructed

Fig. 8: Original face images, PET images and their reconstructed versions upon NMF.

our proposed encryption approach is effective in destroying the semantic meaning of the images. In addition, we give the original dictionary matrix $\mathbf{W}$ in Fig. 7 (c), which represents parts of face in [41]. However, from the encrypted version $\mathbf{W}^*$ obtained from the cloud upon the NMF, we can hardly see the structures of the face parts.

Finally, the reconstruction performance from the decrypted factors is given in Fig. 8, for face images and PET images. It can be seen that high quality of reconstruction is still achieved.

8 Conclusions

We have designed a protocol for outsourcing large-scale NMF computation to a cloud in a secure and verifiable manner. A lightweight encryption scheme based on random permutation and scaling has been proposed to encrypt the data matrix, prior to passing it to a cloud. A simple yet effective verification method has been suggested by exploiting the iterative nature of NMF computations. It has been shown that the verification strategy is quite effective to ensure the correctness of the results from the untrusted cloud server.

References

[1] D. El Badawy, N. Q. K. Duong, and A. Ozerov, "On-the-fly audio source separation," in *Proc. IEEE Int. Workshop on Machine Learning for Signal Process.*, 2014, pp. 1–6.

[2] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons, "Document clustering using nonnegative matrix factorization," *J. Inf. Process. Manag.*, vol. 42, no. 2, pp. 373–386, 2006.

[3] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, 2003, pp. 267–273.

[4] H. Lee, A. Cichocki, and S. Choi, "Kernel nonnegative matrix factorization for spectral eeg feature extraction," *Neurocomputing*, vol. 72, no. 13, pp. 3182–3190, 2009.

[5] Z. Yuan and E. Oja, "Projective nonnegative matrix factorization for image compression and feature extraction," in *Proc. 14th Scandinavian Conf. Image Analysis*, 2005, pp. 333–342.

[6] L.-J. Zhang, "Editorial: Big services era: Global trends of cloud computing and big data," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 467–468, 2012.

[7] G. Brunette and R. Mogull, "Security guidance for critical areas of focus in cloud computing v2. 1," *Cloud Security Alliance*, pp. 1–76, 2009.

[8] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, 2014.

[9] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Ann. Conf. Advances in Cryptology*, 2010, pp. 465–482.

[10] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. Ann. Conf. Advances in Cryptology*, 2010, pp. 483–501.

[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[12] Z. Qin, J. Yan, K. Ren, C. W. Chen, C. Wang, and X. Fu, "Privacy-preserving outsourcing of image global feature detection," in *Proc. IEEE Global Commun. Conf.*, 2014, pp. 710–715.

[13] Y. Ishimaki, H. Imabayashi, K. Shimizu, and H. Yamana, "Privacy-preserving string search for genome sequences with fhe bootstrapping optimization," in *Proc. Int. Conf. Big Data*, 2016, pp. 3989–3991.

[14] X. Liu, R. H. Deng, R. K.-K. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[15] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. 2nd Int. Conf. Theory Cryptography*, 2005, pp. 264–282.

[16] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, 2014.

[17] C. Wang, K. Ren, J. Wang, and K. M. R. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, 2013.

[18] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 69–78, 2015.

[19] F. Chen, T. Xiang, X. Lei, and J. Chen, "Highly efficient linear regression outsourcing to a cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 499–508, 2014.

[20] Z. Xu, C. Wang, K. Ren, L. Wang, and B. Zhang, "Proof-carrying cloud computation: The case of convex optimization," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1790–1803, 2014.

[21] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. IEEE Int. Conf. Computer Commun.*, 2011, pp. 820–828.

[22] ——, "Secure optimization computation outsourcing in cloud computing: A case study of linear programming," *IEEE Trans. Computers*, vol. 65, no. 1, pp. 216–229, 2016.

[23] M. Blanton, Y. Zhang, and K. B. Frikken, "Secure and verifiable outsourcing of large-scale biometric computations," *ACM Trans. Inf. Syst. Security*, vol. 16, no. 3, pp. 1–33, 2013.

[24] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Security*, vol. 4, no. 4, pp. 277–287, 2005.

[25] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Proc. 17th Eur. Symp. Res. Computer Security*, 2012, pp. 505–522.

[26] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, 2012.

[27] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, 2017.

[28] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, "Privacy-assured outsourcing of image reconstruction service in cloud," *IEEE Trans. Emerging Topics in Computing*, vol. 1, no. 1, pp. 166–177, 2013.

[29] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 497–506.

[30] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 501–512.
[31] Y. Zhang and M. Blanton, "Efficient secure and verifiable outsourcing of matrix multiplications," in *Proc. 7th Int. Conf. Inf. Security*, 2014, pp. 158–178.
[32] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Inf. Sci.*, vol. 280, pp. 205–217, 2014.
[33] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf. Comput. Commun. Security*. ACM, 2010, pp. 48–59.
[34] A. Shamir, "How to share a secret," *ACM Commun.*, vol. 22, no. 11, pp. 612–613, 1979.
[35] X. Lei, X. Liao, T. Huang, and H. Li, "Cloud computing service: the case of large matrix determinant computation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 688–700, 2014.
[36] X. Lei, X. Liao, X. Ma, and L. Feng, "Securely and efficiently perform large matrix rank decomposition computation via cloud computing," *Cluster Comput.*, vol. 18, no. 2, pp. 989–997, 2015.
[37] J. Duan, J. Zhou, and Y. Li, "Secure and verifiable outsourcing of nonnegative matrix factorization (nmf)," in *Proc. 4th ACM Workshop on Inf. Hiding and Multimedia Security*, 2016, pp. 63–68.
[38] V. Y. Tan and C. Févotte, "Automatic relevance determination in nonnegative matrix factorization with the beta-divergence," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 35, no. 7, pp. 1592–1605, 2013.
[39] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the β-divergence," *Neural Computation*, vol. 23, no. 9, pp. 2421–2456, 2011.
[40] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.
[41] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.
[42] D. D. Lee and H.-S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 556–562.
[43] S. Arora, R. Ge, R. Kannan, and A. Moitra, "Computing a nonnegative matrix factorization–provably," in *Proc. 44th Annu. ACM Symp. Theory Comput.*, 2012, pp. 145–162.
[44] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Trans. Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.
[45] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.
[46] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 78–87, 2013.
[47] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *Journal of Privacy and Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.
[48] D. E. Knuth, *The art of computer programming*. Addison-Wesley, 2006.
[49] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proc. Ann. Int. Cryptology Conf.*, 1996, pp. 1–15.
[50] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Proc. Int. Workshop on Fast Software Encryption*, 2004, pp. 371–388.
[51] S. Goldwasser and M. Bellare, *Lecture notes on cryptography*, 2001. [Online]. Available: http://cseweb.ucsd.edu/~mihir/papers/gb.pdf
[52] M. Bellare and B. Yee, "Forward-security in private-key cryptography," in *Proc.RSA Conf. on Cryptographers Track*, 2003, pp. 1–18.
[53] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs." *Cryptology EPrint Archive*, p. 332, 2004.
[54] A. Cichocki, R. Zdunek, and S.-i. Amari, "Csiszar's divergences for non-negative matrix factorization: Family of new algorithms," in *Proc. Int. Conf. Ind. Compon. Anal. Blind Signal Separation*, 2006, pp. 32–39.
[55] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: verifiable computation from attribute-based encryption," in *Proc. 9th Int. Conf. Theory Cryptography*, 2012, pp. 422–439.
[56] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 2, pp. 228–233, 2001.
[57] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Proc. Ann. Int. Cryptology Conf.*, 1998, pp. 26–45.
[58] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. Ann. ACM Symposium on Theory of Computing*, 1990, pp. 427–437.
[59] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *Proc. Ann. Int. Cryptology Conf.*, 1991, pp. 433–444.
[60] J.-H. Ahn, S. Kim, J.-H. Oh, and S. Choi, "Multiple nonnegative-matrix factorization of dynamic pet images," in *Proc. Asian Conf. Computer Vision*, 2004, pp. 1009–1013.

Jia Duan (S'15) received the B.S. in software engineering from Chongqing University, Chongqing, China, in 2011, and the M.S. degree in software engineering from University of Macau, Macau, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau, China. His research interests include cloud computing, privacy-preserving computations and deep learning.

Jiantao Zhou (M'11) received the B.Eng. degree from the Department of Electronic Engineering, Dalian University of Technology, in 2002, the M.Phil. degree from the Department of Radio Engineering, Southeast University, in 2005, and the Ph.D. degree from the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, in 2009. He held various research positions with the University of Illinois at Urbana-Champaign, the Hong Kong University of Science and Technology, and the McMaster University. He is currently an Associate Professor with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau. He holds four granted U.S. patents and two granted Chinese patents. His research interests include multimedia security and forensics, and multimedia signal processing. He has co-authored two papers that received the Best Paper Award at the IEEE Pacific-Rim Conference on Multimedia in 2007 and the Best Student Paper Award at the IEEE International Conference on Multimedia and Expo in 2016. He is an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.

Yuanman Li (S'15) received the B.S. in software engineering from Chongqing University, Chongqing, China, in 2012. He received the M.S. degree and the Ph.D. degree from University of Macau, Macau, China, in 2015 and 2018 respectively. His research interests include multimedia security, pattern recognition and machine learning.