A Flexible Privacy-Preserving Framework for Singular Value Decomposition Under Internet of Things Environment

Shuo Chen, Rongxing Lu, Jie Zhang

A Flexible Privacy-preserving Framework for Singular Value Decomposition under Internet of Things Environment

Shuo Chen[1], Rongxing Lu[2], and Jie Zhang[1]

[1] Nanyang Technological University, Singapore, Singapore
chen1087@e.ntu.edu.sg, zhangj@ntu.edu.sg
[2] Faculty of Computer Science, University of New Brunswick, Canada
rxlu@ieee.org

Abstract. The singular value decomposition (SVD) is a widely used matrix factorization tool which underlies many useful applications, e.g. recommendation system, abnormal detection and data compression. Under the environment of emerging Internet of Things (IoT), there would be an increasing demand for data analysis. Moreover, due to the large scope of IoT, most of the data analysis work should be handled by fog computing. However, the fog computing devices may not be trustable while the data privacy is the significant concern of the users. Thus, the data privacy should be preserved when performing SVD for data analysis. In this paper, we propose a privacy-preserving fog computing framework for SVD computation. The security and performance analysis shows the practicability of the proposed framework. One application of recommendation system is introduced to show the functionality of the proposed framework.

1 Introduction

With the prosperous development of communication and computation technologies, the Internet of Things (IoT) is no longer a fantasy nowadays. The big advantage of IoT is that through analyzing the huge amount of information collected from the physical world, the server is capable of making better decisions which would produce considerable benefits. It is estimated that the number of devices connected to the Internet would be about 50 billion by 2020 [1]. It could be anticipated that rather than being conducted on the cloud or inside the intranet of companies, the data analysis work would be performed everywhere and anytime in the future due to the ubiquitousness of IoT. As the amount of data analysis tasks increases in IoT, the singular value decomposition (SVD), which is widely used in different data analysis applications [2, 12, 13, 15, 21], will be performed frequently. However, the traditional way of performing SVD, i.e. calculating the SVD in central server, may not be practical in future IoT due to the vast number of IoT devices. If all the data is transmitted to a central server for computation, it would lead to considerable computation and communication

resource consumption in the server, which would further severely impact the quality of service (QoS) of IoT applications.

To ease the burden of the IoT server and guarantee the QoS, a new technique called fog computing, which is proposed by Cisco [4], is suitable to be applied. The main idea of fog computing is to provide storage, computing and networking services between environmental devices and the central server. The fog devices which are in close proximity to end devices normally possess a certain amount of storage and computation resource. With the equipped resource, the fog devices could process the collected data locally so as to loose the workload of the server. In specific, there are three tiers in the fog computing architecture: environmental tier, edge tier and central tier. In the environmental tier, there are billions of heterogeneous IoT devices collecting and uploading information of the physical world, e.g. medical sensors in eHealth and the mobile phone of each person. The data collected by IoT devices will be transmitted to the edge tier. The fog devices in the edge tier could perform the application-specific operations on received data locally and send the results to the server in the central tier. Owing to the processing of fog devices, the volume of data sent to server could be reduced to a large extent. Since the fog devices are spread in a highly distributed environment, it is impractical for an institution which owns the central server to provide and maintain all those fog devices. Therefore, it is reasonable to assume that the fog devices would be supplied by third parties.

Under the context of fog computing, one could perform the SVD operations on the fog devices. However, another problem which would appear is the privacy issue. The third parties which control the fog devices may not be trustworthy while in many IoT applications, the data collected from the environment is considered as private by the users, e.g. the vital signs in eHealth, the location of vehicles, and the power usage in a smart grid. Performing the SVD on plaintext with fog devices is infeasible if the privacy is a primary concern from the perspective of data owners. Therefore, how to take advantage of fog computing to locally process data in a privacy-preserving way is a challenging issue.

In this paper, we propose a flexible fog computing framework for performing SVD with privacy preserved. The homomorphic encryption technique called Paillier encryption [18] is applied to protect the data privacy. The framework is designed to be capable of supporting different applications based on the SVD computation. The main contributions of this paper are three-fold.

- First, we propose a fog computing framework for privacy-preserving SVD computation to ease the burden of server and protect the data privacy from the fog devices which may not be trustable.
- Second, there is only one communication round between the data providers and data processors in our work while most of the existing works require iterative communications, which brings heavy overhead.
- Third, one application is introduced in details to demonstrate the functionality of the framework. It has been shown that the proposed framework could be easily adopted by the applications which build the trust of unknown entities based on the third-party recommendations.

The remainder of this paper is organized as follow. In Section 2, the preliminaries of our scheme are introduced. The system model, security requirements and design goals are described in Section 3. In Section 4, the proposed framework is presented in details. The security analysis and performance evaluation are discussed in Section 5 and 6. One application based on the proposed privacy-preserving SVD framework is illustrated in Section 7. In Section 8, we discuss the related work, and finally conclude our current work in Section 9.

2 Preliminaries

In this section, the Paillier Cryptosystem [18] and Singular Value Decomposition [9] which are the basis of the proposed framework are reviewed.

2.1 Paillier Cryptosystem

The Paillier Cryptosystem enables the addition operation on plaintext through the manipulation of ciphertext. This homomorphic property is extensively desired in many privacy-preserving applications [16, 20, 25]. In this paper, this feature allows fog devices to process the user data in encrypted form without leaking the data content. The knowledge of Paillier Cryptosystem required for this work is introduced as follow and more details could be referred to [18].
Key Generation: Given one security parameter κ, the public key $\boldsymbol{PK} = (n, g)$ and private key $\boldsymbol{SK}$ could be generated, where the bit length of n is 2κ.
Encryption: Given a message $m \in \mathbb{Z}_n$, randomly choose a number $r \in \mathbb{Z}_n^*$, the ciphertext could be calculated as $c = E(m, r) = g^m \cdot r^n \bmod n^2$.
Decryption: Given a ciphertext $c \in \mathbb{Z}_{n^2}^*$, the plaintext $m = D(c, \boldsymbol{SK})$.
Homomorphic Property: $E(m_1, r_1) \cdot E(m_2, r_2) = E(m_1 + m_2, r_1 \cdot r_2)$.

2.2 Singular Value Decomposition

SVD is a powerful and popular matrix factorization tool that underlies plenty of useful applications, e.g. abnormal detection [12, 15], recommendation system [2, 21] and data compression [13]. Let $\mathbb{A}$ be an $l \times N$ matrix, the SVD of $\mathbb{A}$ is of the form $\mathbb{U}\Sigma\mathbb{V}^T$ where T means conjugate transpose. Σ is an $l \times N$ rectangular diagonal matrix of which diagonal entries are the singular values of $\mathbb{A}$. $\mathbb{U}$ is an $l \times l$ unitary matrix and $\mathbb{V}$ is an $N \times N$ unitary matrix. The columns of $\mathbb{U}(\mathbb{V})$ are the left(right)-singular vectors of $\mathbb{A}$.

Another widely used matrix factorization tool is the eigenvalue decomposition. It is closely related to SVD as shown below:

$$\mathbb{A} \cdot \mathbb{A}^T = \mathbb{U}\Sigma\mathbb{V}^T\mathbb{V}\Sigma^T\mathbb{U}^T = \mathbb{U}\Sigma\Sigma^T\mathbb{U}^T, \quad \mathbb{A}^T \cdot \mathbb{A} = \mathbb{V}\Sigma^T\mathbb{U}^T\mathbb{U}\Sigma\mathbb{V}^T = \mathbb{V}\Sigma^T\Sigma\mathbb{V}^T \tag{1}$$

Equation (1) shows that $\mathbb{U}$ is the eigenvectors of $\mathbb{A} \cdot \mathbb{A}^T$, $\mathbb{V}$ is the eigenvectors of $\mathbb{A}^T \cdot \mathbb{A}$ and the singular values in Σ are the square root of the eigenvalues of $\mathbb{A} \cdot \mathbb{A}^T$ and $\mathbb{A}^T \cdot \mathbb{A}$. We will show that the above relation could be utilized to achieve the privacy-preserving SVD in the later sections.

3 System Model, Security Requirements and Design Goals

In this section, we describe the system model, discuss the security requirements and identify the design goals on privacy-preserving SVD.

3.1 System Model

In this work, we mainly focus on how to utilize the fog computing to compute the SVD of the uploaded data with privacy preserved. Specifically, there are four categories of entity in the system model, namely server, first layer fog device, second layer fog device and environmental device as shown in Fig. 1.

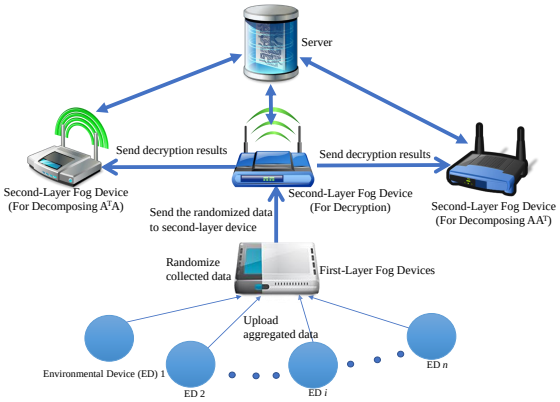


Fig. 1. System Model

Server: Server is a fully trustable entity located in the remote control tier. It is responsible for initializing the whole system and distributing key materials to others. The other operations the server may conduct are application-specific. Two examples will be given in Section 7.

Environmental Device (ED): EDs are the devices distributed in the environmental layer of IoT environment. The analysis on the data uploaded by EDs could enable better decision-making.

First Layer Fog Device (FD): FDs are the fog devices which communicate with EDs directly. FDs process the collected data and upload the results to the second layer fog devices.

Second Layer Fog Device (SD): SDs are the fog devices which communicate with FDs. Compared to FDs, SDs are closer to the server and do not contact with EDs directly. In the proposed framework, there are three SDs playing different roles for SVD operation. One of them is responsible for decrypting the messages from FDs. The other two are in charge of decomposing $\mathbb{A} \cdot \mathbb{A}^T$ and $\mathbb{A}^T \cdot \mathbb{A}$. We denote the one for decryption, decomposing $\mathbb{A} \cdot \mathbb{A}^T$ and decomposing $\mathbb{A}^T \cdot \mathbb{A}$ as SD_d, SD_u and SD_v respectively.

Note that the hierarchical distribution of fog devices is a characteristic inherited from the traditional network architecture. For example, the switchers could function as the first layer fog devices and the gateways in the higher layer could serve as the second layer fog devices.

3.2 Security Requirements

Security is fundamental for the effectiveness of proposed framework. In this work, the server and EDs are assumed to be trustable. The fog devices, i.e. FDs and SDs, are assumed to be honest-but-curious [8, 23] which means they will follow the specified procedures faithfully while being curious about the uploaded data. In addition, FDs and SDs are assumed not to collude with each other. The non-collusion assumption could be realized similarly as the EigenTrust scheme [14]. Briefly speaking, for each SVD computation, the server chooses fog devices based on distributed hash table. Due to the large number of fog devices, it is infeasible for the device providers to determine whether they would be selected for the same computation and negotiate for collusion in advance.

Based on the above assumptions, the confidentiality as the security requirement should be fulfilled, i.e. even FDs and SDs process the collected data, they could not learn anything about the actual value of data. For authenticity and integrity, since there are many existing signature schemes, e.g. Boneh-Lynn-Shacham (BLS) short signature [3], this work just focuses on confidentiality.

3.3 Design Goals

According to the aforementioned system model and security requirements, the proposed framework should achieve the following objectives.

- *The confidentiality should be guaranteed in the proposed framework.* All the user data contained in the transmitted messages should be protected. The processing in fog devices should not leak data privacy.
- *The framework should be flexible enough to be adopted by different applications.* Instead of being the ultimate goal, SVD is the basis or initial step of many applications, which means the further procedures after SVD could be quite different for various scenarios. Therefore, the design of the framework should consider the flexibility such that the results of SVD could be further utilized to achieve the final purposes of different applications.

4 The Proposed Framework

In this section, the proposed framework for SVD computation is presented in details. The framework is composed of five phases: system initialization, data collection, data randomization, pre-computation and eigenvalue decomposition.

4.1 System Initialization

The server is the trustable entity which bootstraps the whole system. Assume the amount of users supported by the system is N, each user data is l-dimensional and the range for each dimension value is $[0, d]$ where d is a constant. The system parameters are κ, κ_1, κ_2 and κ_3. Let $|\bullet|$ denote the bit length of $\bullet$. Given the parameter κ, the server calculates the $\boldsymbol{PK}$: (n, g), where $|n| = 2\kappa$, and the corresponding $\boldsymbol{SK}$. Given the parameters κ_1, κ_2 and κ_3, let $t = 2^{\kappa_1}$, the server randomly chooses two coprime integers W and S such that $W > max(N, l) \cdot d^2$ and $S > max(N, l) \cdot (d^2 + 2tWd + t^2W^2)$, where $|W| = \kappa_2$ and $|S| = \kappa_3$. Then, the server chooses one superincreasing sequence $\vec{\mathbf{a}} = (a_1 = 1, a_2, \cdots, a_l)$ such that $\sum_{j=1}^{i-1} a_j \cdot (d + tW + tS) < a_i$ for $i = 2, \cdots, l$ and $\sum_{i=1}^{l} a_i \cdot (d + tW + tS) < n$. Finally, the server publishes $\{n, g, \vec{\mathbf{a}}\}$ as public parameters, sends $\boldsymbol{SK}$ to SD_d as secret, and sends (W, S) to FDs, SD_u and SD_v as secret respectively.

4.2 Data Collection

In the environmental tier, the data uploaded from N EDs could form the data matrix $\mathbb{A}$. To compute the SVD of matrix $\mathbb{A}$ is the goal of this framework. The ith column of matrix $\mathbb{A}$ $(d_{1i}, \cdots, d_{li})^T$ is from the ith device ED_i. To upload the data, ED_i performs the following steps:

- *Step-1.* Utilize the superincreasing $\vec{\mathbf{a}}$ to compute

$$m_i = a_1 d_{1i} + a_2 d_{2i} + \cdots + a_l d_{li} \tag{2}$$

- *Step-2.* Choose a random number $r_i \in \mathbb{Z}_n^*$ and compute

$$C_i = g^{m_i} \cdot r_i^n \bmod n^2 \tag{3}$$

- *Step-3.* Send the data $C_i || ED_i$ to the FD which communicates with it.

4.3 Data Randomization

For each FD, it will perform the following steps to randomize the received data.

- *Step-1.* For the ith data C_i, FD chooses $2 \cdot l$ random numbers which are $(z_{1i}, \cdots, z_{li})^T$ and $(r_{1i}, \cdots, r_{li})^T$ from the range $[1, t]$. Then FD computes $rz = \sum_{k=1}^{l} a_k \cdot (z_{ki} \cdot W + r_{ki} \cdot S)$.
- *Step-2.* FD randomizes C_i as

$$C_i{}' = C_i \cdot g^{rz} \bmod n^2 = g^{\sum_{k=1}^{l} a_k \cdot (d_{ki} + z_{ki} \cdot W + r_{ki} \cdot S)} \cdot r_i^n \bmod n^2 \tag{4}$$

- *Step-3.* FD sends the randomized data $C_i{}'$ to SD_d.

4.4 Pre-computation

Upon receiving N data from FDs, SD_d will perform the following steps to compute the randomized $\mathbb{A}\mathbb{A}^T$ and $\mathbb{A}^T\mathbb{A}$.

- *Step-1.* For each $C_i^{'}$, SD_d decrypts it with $\boldsymbol{SK}$ and gets the aggregated data

$$m_i^{'} = \sum_{k=1}^{l} a_k \cdot (d_{ki} + z_{ki} \cdot W + r_{ki} \cdot S) \bmod n \tag{5}$$

- *Step-2.* Through the **Algorithm 1** in [6] which is the detailed version of this work, SD_d could recover the randomized value for each dimension of data i.
- *Step-3.* From each $m_i^{'}$, SD_d could get an l-dimensional randomized data. In total, SD_d could get the randomized $l \times N$ data matrix $\mathbb{A}^{'}$, in which the (i, j)th entry is $d_{ij}^{'} = d_{ij} + z_{ij} \cdot W + r_{ij} \cdot S$. Then SD_d simply computes $\mathbb{A}^{'} \cdot (\mathbb{A}^{'})^T$ and $(\mathbb{A}^{'})^T \cdot \mathbb{A}^{'}$, and sends the two resulting matrices to SD_u and SD_v respectively.

4.5 Eigenvalue Decomposition

SD_u: When receiving $\mathbb{A}^{'} \cdot (\mathbb{A}^{'})^T$, SD_u will perform the following steps to compute the left part of the SVD for matrix $\mathbb{A}$, i.e. matrix $\mathbb{U}$ and $\mathbf{\Sigma}$.

- *Step-1.* For each entry ${e_u}^{'}$ of $\mathbb{A}^{'} \cdot (\mathbb{A}^{'})^T$, SD_u derandomizes the entry as follow:

$$e_u = {e_u}^{'} \bmod S \bmod W \tag{6}$$

  The result e_u is the corresponding entry of matrix $\mathbb{A} \cdot \mathbb{A}^T$.
- *Step-2.* After SD_u recovers the matrix $\mathbb{A} \cdot \mathbb{A}^T$, it performs eigenvalue decomposition for matrix $\mathbb{A} \cdot \mathbb{A}^T$ and gets the matrix $\mathbb{U}$ and $\mathbf{\Sigma}$.

SD_v: Similar as SD_u, SD_v performs eigenvalue decomposition on the recovered $\mathbb{A}^T \cdot \mathbb{A}$ to get the right part of the SVD for matrix $\mathbb{A}$, i.e. matrix $\mathbb{V}$ and $\mathbf{\Sigma}$.

By now, the SVD of matrix $\mathbb{A}$ has been separately held by SD_u and SD_v.

The correctness of derandomization The (i, j)th entry ${e_u}_{ij}^{'}$ of $\mathbb{A}^{'} \cdot (\mathbb{A}^{'})^T$ is implicitly formed as

$$\begin{aligned} {e_u}_{ij}^{'} = \sum_{k=1}^{N} d_{ik}^{'} \cdot d_{jk}^{'} &= \sum_{k=1}^{N} (d_{ik} + z_{ik} \cdot W + r_{ik} \cdot S) \cdot (d_{jk} + z_{jk} \cdot W + r_{jk} \cdot S) \\ &= \sum_{k=1}^{N} d_{ik} d_{jk} + \sum_{k=1}^{N} [(z_{ik} d_{jk} + z_{jk} d_{ik}) W + z_{ik} z_{jk} W^2] \\ &\quad + S \sum_{k=1}^{N} [(r_{ik} d_{jk} + r_{jk} d_{ik}) + (z_{ik} r_{jk} + z_{jk} r_{ik}) W + r_{ik} r_{jk} S] \end{aligned} \tag{7}$$

Since

$$\textstyle\sum_{k=1}^{N} d_{ik} d_{jk} + \sum_{k=1}^{N} [(z_{ik} d_{jk} + z_{jk} d_{ik}) W + z_{ik} z_{jk} W^2] < N(d^2 + 2tdW + t^2 W^2) < S,$$

we have $e_{u_{ij}}^{'} \bmod S = \sum_{k=1}^{N} d_{ik}d_{jk} + \sum_{k=1}^{N}[(z_{ik}d_{jk} + z_{jk}d_{ik})W + z_{ik}z_{jk}W^2]$. Also, we have $\sum_{k=1}^{N} d_{ik}d_{jk} < Nd^2 < W$. Thus, $(e_{u_{ij}}^{'} \bmod S) \bmod W = \sum_{k=1}^{N} d_{ik}d_{jk}$ which is the (i,j)th entry of $\mathbb{A} \cdot \mathbb{A}^T$. Similarly, for the entry $e_{v_{ij}}^{'}$ of matrix $(\mathbb{A}^{'})^T \cdot \mathbb{A}^{'}$, $(e_{v_{ij}}^{'} \bmod S) \bmod W = \sum_{k=1}^{l} d_{ki}d_{kj}$ which is the (i,j)th entry of matrix $\mathbb{A}^T \cdot \mathbb{A}$.

5 Security Analysis

In this section, the privacy leakage during normal procedures and the potential attacks which could be conducted by certain participants to snoop data are analyzed. The resistance of the framework against those attacks is discussed and the principles for system configuration are demonstrated.

5.1 Privacy Leakage under Normal Operations

In the proposed framework, each data is encrypted with Paillier Cryptosystem and SD_d is the only fog device which has the private key for decryption. Therefore, the data of each ED could not be discovered by the other EDs and FDs. For SD_d, it could only get the randomized data, i.e. $d_{ij}^{'} = d_{ij} + z_{ij}W + r_{ij}S$, for $i = 1, \cdots, l$ and $j = 1, \cdots, N$ and learn nothing about the real value since SD_d does not know W and S. For SD_u, it could get $\mathbb{U}$ and Σ during normal operations. However, it needs the correct unitary matrix $\mathbb{V}$ to recover $\mathbb{A}$. Since there are infinite unitary matrices, SD_u could not learn original $\mathbb{A}$ with only $\mathbb{U}$ and Σ. Similarly, SD_v could not recover data matrix $\mathbb{A}$ with only $\mathbb{V}$ and Σ.

Based on the above analysis, the data privacy is preserved when the participants follow the defined procedures. In the following, the possible extra computations performed by participants to discover private data are considered.

5.2 Potential Attacks

Since EDs and FDs only have encrypted data, they could not gain much no matter what operations they perform on the ciphertext. Therefore, we mainly discuss the potential attacks from SD_d, SD_u and SD_v in this part.

• **SD_d**: As mentioned above, the information SD_d gets is the randomized data $d_{ij}^{'} = d_{ij} + z_{ij}W + r_{ij}S$, for $i = 1, \cdots, l$ and $j = 1, \cdots, N$. What SD_d needs to do is to find the value of S and W and recover the original data as

$$d_{ij} = d_{ij}^{'} \bmod S \bmod W \tag{8}$$

Since d_{ij} is mixed with the random combination of S and W, it is infeasible for SD_d to determine S and W without additional information. Therefore, we consider the situations in which SD_d knows some of the user data. With the knowledge of user data, the possible operations SD_d could do are as follows:

– *Step-1.* For each known data, SD_d converts the corresponding randomized data to the form $zW + rS$ by computing $d^{'} - d$. Let LC denote the set of converted data and $LC_i = z_iW + r_iS$ denote the ith element of LC.

– *Step-2.* SD_d performs the brute force attack, i.e. tries all possible S. For each try, SD_d performs (modulo S) operation on each LC_i. Then SD_d computes the greatest common divisor (GCD) of the resulting set. If the GCD is larger than 1, it is the value of W and the currently selected S is the correct S.

The rationale behind this attack is: the probability of k randomly chosen integers being coprime is $\frac{1}{\varsigma(k)}$, where $\varsigma(x)$ is Riemann zeta function [17]. When k is large, the probability that they are not coprime is negligible. Thus, after the modulo operations on LC, only when the chosen S is correct, the elements of the resulting set are of the form zW and have a GCD larger than 1, which is W.

Note that, in some cases, SD_d could still form a set, in which the elements are of the form $zW + rS$ with high probability, even it does not know any user data. For example, if the data matrix is sparse, most of the randomized data is already of the desired form. Another case is that data range is not large enough compared to the amount of data, SD_d could compute $DV = d'_{ij} - d'_{i'j'}$ for all possible pairwise combinations $(d'_{ij}, d'_{i'j'})$ and some of the resulting DVs will be of the desired form. For those cases, SD_d could perform the brute force attack.

Parameter Selection. To resist the brute force attack in the possible cases, $|S|$, i.e. κ_3, should be at least equal to 80. Moreover, SD_d could compute $LDV = LC_i - LC_j$ for all possible combinations (LC_i, LC_j). If certain combinations have the same zW inside, those resulting LDVs would be of the form $(r_i - r_j)S$. SD_d could learn S efficiently by computing the GCD of those LDVs even when $|S| \geq 80$. To avoid the case, the randomly chosen z_{ij} should be different with each other with high probability. The z_{ij} is chosen from the range $[1, t]$, and the total number of z_{ij} is $l \cdot N$. According to the generalized birthday problem [22], the probability of at least two chosen z_{ij} match is $1 - \exp^{\frac{-(lN)^2}{2t}}$. Thus, the probability of no match is $\exp^{\frac{-(lN)^2}{2t}}$ and the parameter κ_1 which determines t could be selected accordingly. Note that if FDs could cooperatively choose the set of z_{ij} such that there is no match, then the range t only needs to be larger than $l \cdot N$.

• **SD_u**: SD_u could get $\mathbb{U}$ and $\mathbf{\Sigma}$. To recover matrix $\mathbb{A}$, SD_u needs to find the unitary matrix $\mathbb{V}$. Let λ denote the rank of $\mathbb{A}$. The first λ elements of each row in $\mathbb{V}$ correspond to a column of $\mathbb{A}$, so if SD_u knows the left λ columns of $\mathbb{V}$, it could recover the original data. Since there are infinite unitary matrices, SD_u could not determine the correct $\mathbb{V}$ if it has no additional information. Thus, we assume that SD_u could get N' original data in some cases. Note that using one data, i.e. one column of $\mathbb{A}$, could form l equations for the same row of $\mathbb{V}$. Solving the equations from one data, SD_u could get the first λ elements of that row.

When $N' < N-1$, since each row of $\mathbb{V}$ is linearly independent with each other, obtaining one row does not help to learn the other rows. Thus, SD_u could not utilize the known data to learn the rest unknown data.

When $N' = N - 1$, SD_u could determine the first λ elements of $(N - 1)$ rows of $\mathbb{V}$, then the first λ elements of the last unknown row could be determined due to $\sum_{i=1}^{N} v_{ij}^2 = 1$. The last unknown user data could be recovered accordingly.

• **SD_v**: The purpose of SD_v is to find the unitary matrix $\mathbb{U}$. Different from the case of SD_u, the first λ elements of each row in $\mathbb{U}$ correspond to a row of $\mathbb{A}$, i.e. the

data value of a certain dimension from all users. If SD_v knows the left λ columns of $\mathbb{U}$, it could recover the original data. Similarly, we assume that SD_v knows $N^{'}$ linearly independent data in some cases. We have "linearly independent" here because linearly dependent data would not produce new linearly independent equations. Thus, only the number of linearly independent data matters. Each data, i.e. one column of $\mathbb{A}$, could form one equation for each row of $\mathbb{U}$.

When $\lambda < l$, if $N^{'} = \lambda$, SD_v could form λ linearly independent equations for each row of $\mathbb{U}$. Then through solving equations, SD_v could determine the left λ columns of $\mathbb{U}$ and thus recover the whole $\mathbb{A}$ which contains the other unknown user data. On the other hand, if $N^{'} < \lambda$, SD_v could not recover the other unknown linearly independent data due to the lack of enough linearly independent equations. However, for the data which is linearly dependent with the known data, SD_v could recover them because the columns of $\Sigma\mathbb{V}^T$ have the same linear relationships as those existing among user data.

When $\lambda = l$, there is an additional condition for solving the equations of $\mathbb{U}$, i.e. $\mathbb{U}$ is a unitary matrix. Specifically, the l rows of $\mathbb{U}$ could be regarded as the coordinate axis of l-dimensional space whose rotation degree of freedom is $l-1$. For each linearly independent data known to SD_v, the rotation degree of freedom of the coordinate axis reduces by 1. Therefore, if $N^{'} = l-1$, the rotation degree of freedom reduces to 0, i.e. the coordinate axis is fixed. Moreover, since $\sum_{j=1}^{l} u_{ij}^2 = 1$, each row of $\mathbb{U}$ could be seen as a point locating on the unit sphere of l-dimension. Thus, the set of intersection points between the fixed coordinate axis and the l-dimensional unit sphere is the solution of $\mathbb{U}$.

Based on the above analysis, the proposed framework could resist the potential attacks launched by SD_d through properly choosing z_{ij} and S. For SD_u, only when $(N-1)$ user data is obtained, it could learn the last unknown data. For SD_v, if $\lambda < l$, the framework could resist not more than $(\lambda - 1)$ linearly independent user data leakage and it could resist not more than $(\lambda - 2)$ linearly independent user data leakage if $\lambda = l$.

6 Performance Evaluation

In this section, we evaluate the performance of the proposed fog computing framework in terms of the capacity and efficiency. The capacity demonstrates the number of required ciphertexts for different matrix sizes while the efficiency indicates the computational complexity and communication overhead.

6.1 Capacity

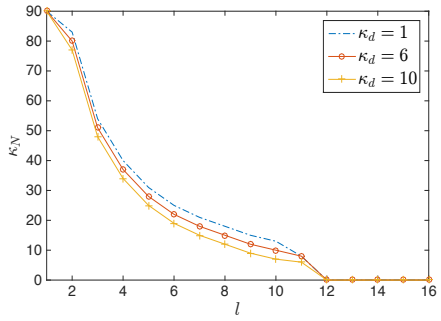
In the proposed framework, the aggregated randomized data is of the form $m_i^{'} = \sum_{k=1}^{l} a_k \cdot (d_{ki} + z_{ki} \cdot W + r_{ki} \cdot S)$. To guarantee the aggregated data could be decrypted correctly, $m_i^{'}$ should be less than n, i.e. the constraint $\sum_{k=1}^{l} a_k \cdot (d + tW + tS) < n$ must be fulfilled. At the same time, the superincreasing sequence $\vec{\mathbf{a}}$ has the constraint: $\sum_{j=1}^{i-1} a_j \cdot (d + tW + tS) < a_i$ for $i = 2, \cdots, l$. Moreover, in order to derandomize the data, W and S need to fulfill: $W > max(N, l) \cdot d^2$ and $S >$

$max(N,l)\cdot(d^2+2tWd+t^2W^2)$. To resist the potential attack from SD_d in special cases, t should be chosen based on N and l, and κ_3 should not be less than 80.

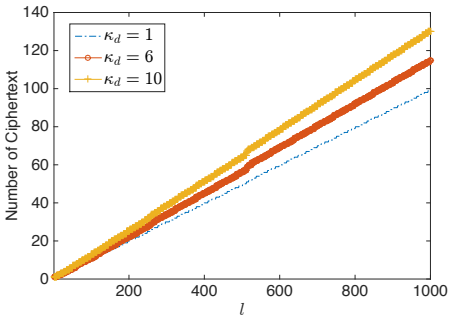
Let κ_N, κ_l and κ_d denote the bit length of N, l and d respectively. For simplicity, assume that FDs could cooperatively select the z_{ij} such that no match happens. Then $\kappa_1 = \kappa_N + \kappa_l + 1$ is enough. To meet $W > max(N,l)\cdot d^2$, we have $\kappa_2 > max(\kappa_N,\kappa_l) + 2\kappa_d$. Then due to $S > max(N,l)\cdot(d^2+2tWd+t^2W^2)$, $\kappa_3 > max(\kappa_N,\kappa_l)+2\kappa_1+2\kappa_2 > 3\cdot max(\kappa_N,\kappa_l)+4\kappa_d+2\kappa_N+2\kappa_l+2$. For the sequence $\vec{\mathbf{a}}$, $|a_2| > \kappa_3+\kappa_1$ and $|a_3| > 2(\kappa_3+\kappa_1)$. It is easy to find that $|a_i| > (i-1)(\kappa_3+\kappa_1)$ and $|\sum_{k=1}^{l} a_k\cdot(d+tW+tS)| > l(\kappa_3+\kappa_1)$. Thus, the bit length of aggregated data:

$$|m_i^{'}| = \begin{cases} l[3max(\kappa_N,\kappa_l)+4\kappa_d+3\kappa_N+3\kappa_l+3] & \text{, if } max(\kappa_N,\kappa_l)+2\kappa_1+2\kappa_2 > 80. \\ l(80+\kappa_N+\kappa_l+1) & \text{, else.} \end{cases}$$

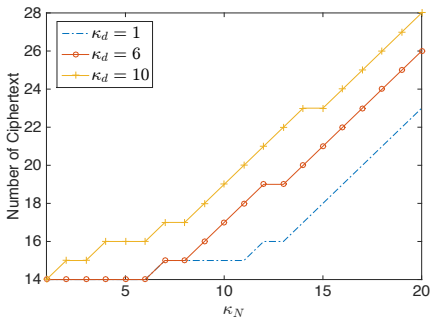
It is obvious that the data dimension has a great influence on the aggregated data length. Given different κ_d and l, the number of users which one ciphertext with $|n| = 1024$ could support is evaluated as shown in Fig. 2(a).



(a) $|n| = 1024$



(b) $\kappa_N = 10$



(c) $l = 150$

Fig. 2. Capacity of the proposed framework

From Fig. 2(a), it could be seen that the increase of dimensionality could dramatically decrease the number of users which one ciphertext could support, while the impact of data range d is not that significant. One ciphertext could support large number of users with low dimensional data, e.g. 2^{37} users with

4-dimensional data and 2^{15} users with 8-dimensional data. To support higher dimensional data for the same amount of users, each ED needs to use multiple ciphertexts to aggregate data, e.g. to support 2^{15} users with 16-dimensional data needs 2 ciphertexts each of which aggregates 8 dimensions. Given different l and κ_N, the number of required ciphertexts with $|n| = 1024$ is evaluated in Fig. 2(b) and Fig. 2(c) respectively. It could be seen that each ED needs to use $O(l \cdot log(N))$ ciphertexts for uploading data.

6.2 Efficiency

As analyzed above, each ED may need more than one Paillier ciphertext for aggregating user data. Let N_C denote the number of required ciphertexts for each ED. In the following, the computational complexity and communication overhead of the proposed framework are analyzed.

Computational Complexity: Because the crypto-operations are much heavier than the computations on plaintext, the amount of crypto-operations is the main concern in this part. Since the fog computing platform in current stage possesses the resource comparable to that of a smart phone, we have implemented the Paillier Cryptosystem on an Android mobile phone. The model number of the phone is Huawei Honor 3C (H30-U10) with the system parameters as: ARM Cortex-A7 4-core CPU @1.3GHz, 2GB memory and 4.2.2 Android version. When $|n| = 1024$, the average running time (1000 iterations) for the exponentiation in $\mathbb{Z}_{n^2}$ is 55.493 milliseconds and the time for the multiplication in $\mathbb{Z}_{n^2}$ is 0.201 milliseconds. It is obvious that the cost of multiplication is negligible compared to the cost of exponentiation. According to the procedures of proposed framework, the computational cost for different entities is as shown in Table 1.

Table 1. Computational Cost of the Proposed Framework

Entity	Computational Cost (milliseconds)
ED	$2 \times 55.493 \cdot N_C$
FDs	$55.493 \cdot N \cdot N_C$
SD_d	$55.493 \cdot N \cdot N_C$

Note that SD_u and SD_v only perform computations on plaintext, and server is only in charge of system initialization. Therefore, their computation cost is negligible compared to the other entities. Another notable thing is that the evaluation implicitly assumes the IoT environment devices are as powerful as a smart phone. This is true for the IoT applications which use mobile phones or vehicles to upload environmental information. However, for the applications utilizing low power sensors as EDs, the Paillier operations are still too heavy. To circumvent this issue, the sensor may transmit its data to nearby more powerful device for conducting the crypto-operations. For example, the wristband could connect with the mobile phone for processing and uploading data.

Communication Overhead: In this part, the communication overhead during SVD computation is evaluated. Note that for Paillier Cryptosystem, the ciphertext space is $\mathbb{Z}_{n^2}$. Thus, the bit length of one ciphertext is $2|n|$. The overhead of each communication flow is as shown in Table 2.

Table 2. Communication Overhead of the Proposed Framework

Communication Flow	Bit Length of Message
ED$\rightarrow$ FD	$N_C \cdot 2\|n\|$
FDs $\rightarrow$ SD$_d$	$N \cdot N_C \cdot 2\|n\|$
SD$_d$ $\rightarrow$ SD$_u$	$l^2(2\kappa_1 + 2\kappa_3 + \kappa_N)$
SD$_d$ $\rightarrow$ SD$_v$	$N^2(2\kappa_1 + 2\kappa_3 + \kappa_l)$

7 Applications

In this section, we discuss the potential IoT applications which could utilize the proposed framework. Basically, the proposed framework could be applied if the application possesses the following characteristics: 1) the application collects the environmental information for data analysis; 2) the data analysis is based on SVD; 3) the number of data analysis tasks is huge; 4) the environmental information is considered as privacy by the application users. Actually, the last two characteristics are the motivation of this work. The large amount of data analysis tasks motivates us to analyze data on fog computing platform. The privacy concern requires the analysis being privacy-preserving. In the below, we describe a recommendation system as an example to demonstrate the functionality of the proposed framework. More applications which indicate the flexibility of the proposed framework could be found in the detailed version [6].

7.1 Localized Recommendation System

Imagine that there are tens of restaurants in the local region where you live. You have already been some of them and want to try a new one, let's say restaurant p, tonight. Before you go there, you would like to get a reputation score about p from other people in the same region. If the score is too low, you may change the plan. We call the recommendation of local resource as localized recommendation. The advantages of localized food recommendation system over the centralized food review sites could be referred in [6]. Since the personal taste is considered as privacy by many people and the recommendation tasks could appear in different regions frequently and concurrently, to get the local reputation score, we should utilize the proposed framework to conduct the SVD-based collaborative filtering as described in [21]. The detailed procedures are as follows:

– *Step-1.* The user c uploads his rating vector to FD with his mobile phone and informs FD that he is interested in restaurant p. FD collects the rating

vectors from other users inside this region. Note that to remove sparsity, each user fills in the ratings of unknown restaurants with his average rating.

- *Step-2.* FD uploads randomized data to SD_d. SD_d, SD_u and SD_v conduct some extra steps for normalizing the data matrix and perform SVD to get $\mathbb{U}$, $\mathbb{V}$ and Σ. Due to page limit, we omit the normalization steps here. Please refer to the detailed version [6] for the extra normalization steps.
- *Step-3.* SD_u and SD_v reduce $\mathbb{U}$, $\mathbb{V}$ and Σ to k dimension, and compute $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}$ and $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T$ respectively, i.e. SD_u holds $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}$ and SD_v holds $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T$. Let $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}(c)$ denote the row of $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}$ which contains the information of user c and $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T(p)$ denote the column of $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T$ which contains the information of restaurant p. The reputation score of restaurant p for user c is computed as $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}(c)\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T(p)$. Note that we use z-scores for normalization, so we do not need to add the user average back as in [21].
- *Step-4.* SD_u and SD_v send $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}(c)$ and $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T(p)$ to SD_d for reputation score computation. To prevent SD_d from inferring information, SD_u and SD_v randomize $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}(c)$ and $\Sigma_k^{\frac{1}{2}}\mathbb{V}_k^T(p)$ with W and S respectively. For example, let u_i denote the ith entry of $\mathbb{U}_k\Sigma_k^{\frac{1}{2}}(c)$, SD_u randomizes it as $u_i + z_iW + r_iS$.
- *Step-5.* SD_d multiplies the two randomized vectors and sends the result $score_p{}'$ to FD. Since FD knows W and S, it could recover the reputation score as $score_p = score_p{}' \bmod S \bmod W$ and send $score_p$ to user c.

From the above description, it has been shown that the proposed framework could utilize the result of SVD operation to compute the reputation score of unknown restaurants for a specific user. It is straightforward that other similar applications which build the trust of unknown entities based on the third-party recommendations could also adopt the framework. Also note that in the above example, the server does not participate in the process, which means the proposed framework completes all the workload in the edge tier.

8 Related Works

In literature, there are a few works which are related to privacy-preserving SVD computation. Polat et al. [19] proposed a SVD-based collaborative filtering scheme in which the data privacy is protected by randomized perturbation. However, their scheme has been proven unsecure by [24]. Note that the randomization in this work does not have the feature of the randomized perturbation in [19]. Thus, the technique in [24] is infeasible for our work. Canny et al. [5] proposed a collaborative filtering scheme which achieves the SVD computation with privacy-preserving. However, their scheme is specifically designed for the recommendation application. Han et al. [10] proposed a secure protocol for SVD computation. However, their scheme could only support the computation between two parties. Hegeds et al. [11] proposed a private SVD computation for low rank approximation in distributed P2P systems. Compared to our work,

the works in [5, 10, 11] have limited applications and require considerable iterations for convergence which brings heavy overhead. Duan et al. [7] proposed a privacy-preserving framework which supports the computation of the learning algorithms which could be expressed as iterative form. Their work could support many learning algorithms while also requiring multiple rounds for the convergence of algorithms, which brings considerable overhead. For example, their scheme needs 83 minutes to compute the SVD for the Enron Email Data set which is a 150×150 matrix while our work would need $N_C = 15$ ciphertexts to aggregate the 150-dimensional data for each of the 150 users and only takes 499 seconds in total. Note that the evaluation in [7] sums up the computation time for all users even the computation of each user is actually performed concurrently. For fair comparison, our evaluation also accumulates the computation time of all users.

9 Conclusions

In this paper, a flexible fog computing framework for privacy-preserving SVD computation has been proposed. The framework divides the SVD calculation into two eigenvector decomposition operations and distributes the two tasks to different fog devices. The security analysis shows that the user data privacy is preserved during transmission, aggregation and eigenvector decomposition. The possible attacks from the second layer fog devices are also analyzed and the resistance of the framework is discussed. The performance analysis has indicated the capacity of the framework and shows that the data dimension is the most important factor influencing the efficiency of the system. Moreover, one application is given as an example to demonstrate the functionality of the proposed framework. Compared with the existing works, our framework could support large scope of applications with relatively small resource consumption.

References

1. Cisco delivers vision of fog computing to accelerate value from billions of connected devices (Jan 2014), `http://newsroom.cisco.com/release/1334100/Cisco-Delivers-Vision-of-Fog-Computing-to-Accelerate-Value-from-Billionsof-Connected-Devices-utm-medium-rss`, press release
2. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: Icml. vol. 98, pp. 46–54 (1998)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of cryptology 17(4), 297–319 (2004)
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing. pp. 13–16. ACM (2012)
5. Canny, J.: Collaborative filtering with privacy. In: Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on. pp. 45–57. IEEE (2002)
6. Chen, S., Lu, R., Zhang, J.: A flexible privacy-preserving framework for singular value decomposition under internet of things environment. arXiv preprint arXiv:1703.06659 (2017)

7. Duan, Y., Canny, J., Zhan, J.: P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users. In: Proceedings of the 19th USENIX Conference on Security. pp. 14–14. USENIX Security'10, USENIX Association, Berkeley, CA, USA (2010)
8. Goethals, B., Laur, S., Lipmaa, H., Mielikinen, T.: On private scalar product computation for privacy-preserving data mining. In: International Conference on Information Security and Cryptology. pp. 104–120. Springer Berlin Heidelberg (2004)
9. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU Press (2012)
10. Han, S., Ng, W.K., Philip, S.Y.: Privacy-preserving singular value decomposition. In: 2009 IEEE 25th International Conference on Data Engineering. pp. 1267–1270. IEEE (2009)
11. Hegedűs, I., Jelasity, M., Kocsis, L., Benczúr, A.A.: Fully distributed robust singular value decomposition. In: Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on. pp. 1–9. IEEE (2014)
12. Idé, T., Kashima, H.: Eigenspace-based anomaly detection in computer systems. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 440–449. ACM (2004)
13. Kalman, D.: A singularly valuable decomposition: the svd of a matrix. The college mathematics journal 27(1), 2–23 (1996)
14. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web. pp. 640–651. ACM (2003)
15. Lee, Y.J., Yeh, Y.R., Wang, Y.C.F.: Anomaly detection via online oversampling principal component analysis. IEEE Transactions on Knowledge and Data Engineering 25(7), 1460–1470 (2013)
16. Lu, R., Liang, X., Li, X., Lin, X., Shen, X.S.: Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. Parallel and Distributed Systems, IEEE Transactions on 23(9), 1621–1631 (2012)
17. Nymann, J.: On the probability that k positive integers are relatively prime. Journal of Number Theory 4(5), 469–473 (1972)
18. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
19. Polat, H., Du, W.: Svd-based collaborative filtering with privacy. In: Proceedings of the 2005 ACM symposium on Applied computing. pp. 791–795. ACM (2005)
20. Sang, Y., Shen, H., Tian, H.: Privacy-preserving tuple matching in distributed databases. Knowledge and Data Engineering, IEEE Transactions on 21(12), 1767–1782 (2009)
21. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender system-a case study. Tech. rep., DTIC Document (2000)
22. Wagner, D.: A generalized birthday problem. In: Annual International Cryptology Conference. pp. 288–304. Springer (2002)
23. Y., L., B., P.: Privacy preserving data mining. In: Annual International Cryptology Conference. pp. 36–54. Springer Berlin Heidelberg (2000)
24. Zhang, S., Ford, J., Makedon, F.: Deriving private information from randomly perturbed ratings. In: Proceedings of the 2006 SIAM International Conference on Data Mining. pp. 59–69. SIAM (2006)
25. Zhong, S.: Privacy-preserving algorithms for distributed mining of frequent itemsets. Information Sciences 177(2), 490–503 (2007)