Efficient Federated Matrix Factorization Against Inference Attacks

DI CHAI, Hong Kong University of Science and Technology, China
LEYE WANG, MOE Key Lab of High Confidence Software Technologies, Peking University, China
KAI CHEN, Hong Kong University of Science and Technology, China
QIANG YANG, WeBank AI Lab, China and Hong Kong University of Science and Technology, China

Recommender systems typically require the revelation of users' ratings to the recommender server, which will subsequently use these ratings to provide personalized services. However, such revelations make users vulnerable to a broader set of inference attacks, allowing the recommender server to learn users' private attributes, e.g., age and gender. Therefore, in this paper, we propose an efficient federated matrix factorization method that protects users against inference attacks. The key idea is that we obfuscate one user's rating to another such that the private attribute leakage is minimized under the given distortion budget, which bounds the recommending loss and overhead of system efficiency. During the obfuscation, we apply differential privacy to control the information leakage between the users. We also adopt homomorphic encryption to protect the intermediate results during training. Our framework is implemented and tested on real-world datasets. The result shows that our method can reduce up to 16.7% of inference attack accuracy compared to using no privacy protections.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; • **Computing methodologies** → *Factorization methods*; • **Information systems** → *Recommender systems;*

Additional Key Words and Phrases: Federated learning, inference attack, matrix factorization

ACM Reference format:
Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2022. Efficient Federated Matrix Factorization Against Inference Attacks. *ACM Trans. Intell. Syst. Technol.* 13, 4, Article 59 (June 2022), 20 pages.
https://doi.org/10.1145/3501812

1 INTRODUCTION

The recommender system has become an indispensable component in our lives to provide personalized services, e.g., suggesting movies and music. A recommender system typically requires collecting users' ratings in one place, then performing analysis and predictions. However, the

The work is partly supported by the NSFC Grant no. 61972008, the PKU-Baidu Fund Project no. 2019BD005, the Hong Kong RGC TRS T41-603/20R and the National Key Research and Development Program of China under Grant No. 2018AAA0101100.

Authors' addresses: D. Chai and K. Chen, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China; emails: {dchai, kaichen}@cse.ust.hk; L Wang, MOE Key Lab of High Confidence Software Technologies, School of Computer Science, Peking University, Haidian District, Beijing, China, 100080; email: leyewang@pku.edu.cn; Q. Yang, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China. & WeBank AI Lab, Nanshan district, Shenzhen, China, 518040; email: qyang@cse.ust.hk.

2157-6904/2022/06-ART59 $15.00
https://doi.org/10.1145/3501812

rating information is very private because it represents users' preferences and indicates users' private attributes, such as age and gender. To this end, the privacy-preserving recommender system has become a hot research topic.

One of the most widely used recommendation technologies is **matrix factorization (MF)** [23], which decomposes the rating matrix into two low-dimensional matrices, capturing the latent factors of users and items. Numerous works have proposed privacy-preserving MF algorithms to address the privacy issues. These works can be categorized into encryption-based methods [20, 26], **differential privacy (DP)** based methods [5], and federated learning (FL) based methods [7]. The encryption-based methods protect users' privacy by encrypting the data before uploading it to the service providers. However, they usually require two non-colluding service providers (i.e., a recommendation server and a crypto-service provider), which are hard to find in the real world and thus have relatively low practicality. The encryption-based methods also sustain large time overhead brought by complex computation under ciphertext or circuits. The differential privacy based methods protect privacy by minimizing the probability of identifying any user's data. It takes a trade-off between recommending accuracy and privacy. However, the differential privacy based methods cannot offer confidentiality of users' rating values and rating item set by definition, and thus are vulnerable to inference attacks. Federated learning methods protect privacy by keeping users' data locally and performing joint model training between all the users. The exchanged intermediate results are protected by the homomorphic encryption, thus they guarantee data confidentiality.

In this paper, we focus on federated matrix factorization because of its following strengths:

- Compared with other technologies, federated learning avoids collecting users' data in one place, thus it is easier to satisfy privacy-preserving regularizations (e.g., GDPR).
- Compared with the encryption-based methods, which need two non-colluding servers, the federated solution only requires one semi-honest server, thus it has more practicality.
- FL has good expandability, making it more flexible. For example, we can apply both homomorphic encryption and differential privacy in FL to get a more secure solution.

The first secure federated matrix factorization is proposed by Chai et al. [7]. They proved that the gradients leak private data. Thus, it is necessary to adopt the homomorphic encryption to enhance the confidentiality of rating values. Two schemes are proposed in their paper: *FullText* and *PartText*:

- *FullText*: Users upload gradients for all the items. The gradients of non-rated items are set to zero to avoid affecting the recommending accuracy. The server cannot specify the zero gradients because the numbers are all encrypted using homomorphic encryption.
- *PartText*: Users only upload gradients for the rated items.

However, neither the proposed *FullText* nor *PartText* solution is practical, they show a polarization of performance between efficiency and privacy preservation. The *FullText* solution leaks nothing to the server, however, it brings a large time overhead. The *PartText* solution minimizes the time consumption, however, it leaks the rated item set, which can be used by the adversary to attack users' private attributes.

To this end, we propose an **efficient** and **inference-attack-safe** federated matrix factorization solution, called EIFedMF. EIFedMF is much faster than *FullText* and can protect users against attribute inference attacks compared with *PartText*. The key idea is that we learn an obfuscation function to obfuscate users' uploading item set, such that the mutual information between the uploaded information and the private attribute is minimized under a data distortion budget, which bounds the utility of the obfuscation results.

Apart from the problem of learning the obfuscation function itself, the federated learning setting also brings lots of challenges to us:

- Most machine learning algorithms require that the data should be collected in one place, thus it cannot be directly used in our system. We need to transfer some of them into the federated setting to build our system.
- Homomorphic encryption is indispensable in our system to protect users' private data, and it brings challenges to our system design because the type of homomorphic encryption's supported operation is limited (e.g., it cannot compare two ciphertexts or compute their absolute values).

In summary, we made the following contributions:

(a) To the best of our knowledge, this is the first efficient federated matrix factorization framework that can protect users against attribute inference attacks. Compared with existing secure MF works, EIFedMF is more practical, i.e., faster than $FullText$ and safer than $PartText$.
(b) We propose an obfuscation based method to protect users against inference attacks in the federated matrix factorization. EIFedMF is well accommodated into the federated learning setting and does not bring any new private data leakage.
(c) We implement the proposed method and evaluate it on real-world datasets. The evaluation results show that our method can reduce the inference attack accuracy by up to 16.7% compared with $PartText$ solution, and only brings 2.4% recommendation loss (i.e., root mean square error), thus it guarantees the recommending utility. Compared with $FullText$ solution, our method can significantly improve the system efficiency by reducing about 40%~50% number of uploading items.

2 PRELIMINARIES

In this section, we briefly introduce some technologies that are closely related to our work, which are homomorphic encryption, federated matrix factorization, and oblivious transfer.

2.1 Homomorphic Encryption

Homomorphic encryption (HE) is one of the proudest technologies in cryptography, and its study can be traced back to the 1970s. Briefly, HE allows computations directly to run on ciphertext without decryption. A typical application of HE is private cloud computing. We say an HE algorithm is homomorphic over operation "$\star$" if the following equation holds [1]:

$$[\![m_1]\!] \star [\![m_2]\!] = [\![m_1 \star m_2]\!] \ \forall m \in M$$

Where $[\![m]\!]$ is the ciphertext of m, M is the set of all possible messages.

Because of HE's nice homomorphic property, it is also widely used in federated learning to protect the exchanged intermediate results between different parties. In this paper, we use the CKKS [8] encryption schema, which supports both homomorphic addition and multiplication in a limited depth.

2.2 Federated Matrix Factorization

Matrix factorization based recommender system is first introduced in the work of Koren et al. [23], and **stochastic gradient decent (SGD)** is used to train the model. Given the rating information $r_{ij} : (i,j) \in \mathcal{M}$, Equation (1) shows the loss function of matrix factorization and Equations (2)–(5) are the parameter updating formulas using SGD.

$$\min_{U,V} \frac{1}{M}(r_{i,j} - \langle u_i, v_j \rangle)^2 + \lambda ||U||_2^2 + \mu ||V||_2^2 \tag{1}$$

$$u_i^t = u_i^{t-1} - \gamma \nabla_{u_i} F(U^{t-1}, V^{t-1}) \tag{2}$$

$$v_i^t = v_i^{t-1} - \gamma \nabla_{v_i} F(U^{t-1}, V^{t-1}) \tag{3}$$

$$\nabla_{u_i} F(U,V) = -2 \sum_{j:(i,j)} v_j (r_{ij} - \langle u_i, v_j \rangle) + 2\lambda u_i \tag{4}$$

$$\nabla_{v_j} F(U,V) = -2 \sum_{i:(i,j)} u_i (r_{ij} - \langle u_i, v_j \rangle) + 2\lambda v_j \tag{5}$$

where $U \in \mathbf{R}^{n\times d}$ is the user profile, $V \in \mathbf{R}^{m\times d}$ is the item profile, λ and μ are small positive values to rescale the penalizer, $j:(i,j)$ means all the items that evaluated by user-i, $i:(i,j)$ means all the users that have evaluated item-j.

Federated matrix factorization allows users to jointly train the model while keep rating data locally. Users send the intermediate results (i.e., gradients) to the server instead of sharing raw rating data. Algorithm 1 shows the procedure of federated matrix factorization.

ALGORITHM 1: Federated Matrix Factorization with HE (*PartText*)

```
Initialize: Server initializes V and holds the public key, Users initialize U and hold the secret key
Output: Converged U and V
Server keeps latest $[\![V]\!]$ for all users' download
User local update:
    Download $\{[\![v_j]\!] | j \in (i,j)\}$ from server
    Decrypt and get $\{v_j | j \in (i,j)\}$
    Local updates: $u_i^t = u_i^{t-1} - \gamma \nabla_{u_i} F(U^{t-1}, V^{t-1})$
    $Gradient_{ij} = -2\gamma u_i (r_{ij} - \langle u_i, v_j \rangle) + 2\lambda v_j$
    Encrypt and send $\{[\![-\lambda Gradient_{ij}]\!] | j \in (i,j)\}$ to the server, where $\lambda$ is the learning rate
Server update:
    Receive $\{[\![-\lambda Gradient_{ij}]\!] | j \in (i,j)\}$ from user-i
    Update : $[\![v_j^t]\!] = [\![v_j^{t-1}]\!] + [\![-\lambda Gradient_{ij}]\!]$ for $j:(i,j)$
```

2.3 Oblivious Transfer

Oblivious transfer (OT) is a secure computation protocol in which a sender transfers one of the potentially many pieces of data to a receiver and guarantees that the sender remains oblivious about which piece of data is transferred. A typical OT protocol is 1-out-of-2 oblivious transfer, which assumes that the sender has two messages and wants to send one of them to the receiver.

In this paper, we use the 1-out-of-n oblivious transfer, which can be naively implemented by doing n times of 1-out-of-2 OT. And we adopt the protocol of Naor et al. [25], which reduces the complexity of 1-out-of-n OT from $O(n)$ to $O(logn)$.

3 PROBLEM FORMULATION

In this section, we introduce the security definition of our system, demonstrate the issues of existing solutions in detail, and formulate our problem.

3.1 Security Definition

In this paper, we assume that all the participants, including the server and users, are semi-honest. By definition, the server and users could be compromised by an adversary, but they will follow the prescribed protocol correctly. We also assume that the server does not collude with any user.

3.2 Neither PartText nor FullText is Practical

According to Algorithm 1, which is exactly the *PartText* setting, each user uploads the encrypted gradients which are subsequently used by the server to update the item vectors. We can notice that,

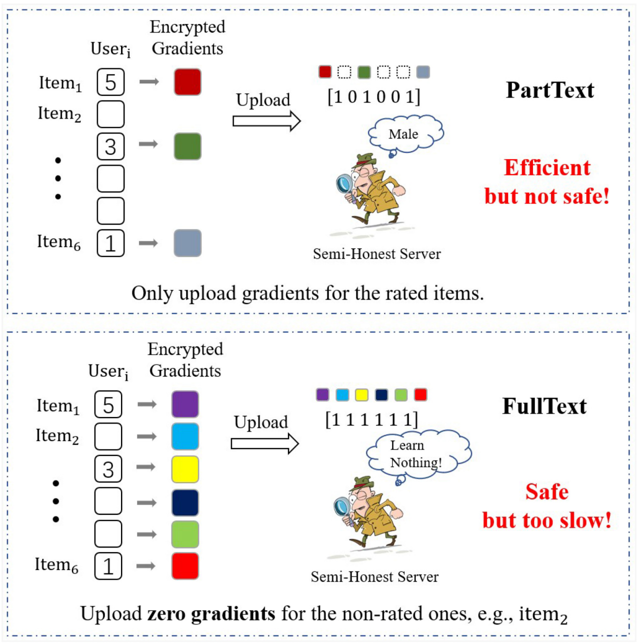


Fig. 1. Neither *FullText* nor *PartText* is a good choice.

although the gradients are encrypted, there is still some information leakage. The server knows which items are rated by a particular user. Such information leakage is not trivial because it could make users vulnerable to a broader set of inference attacks [34].

By contrast, *FullText* setting requires the gradients uploading for all the items. If one item is not rated by the user, then encrypted zero gradients will be used. The zero gradients cannot be discriminated by the server because HE algorithms (e.g., Paillier [27]) exploit random numbers in the encryption, making the ciphertext look very different, even if the raw data are the same. And the random numbers can be removed by the secret key during decryption. Actually, such property exists as long as the HE is secure against **chosen-plaintext attack (CPA)** which is the basic requirement of HE algorithms [14].

Figure 1 illustrates the problem of *FullText* and *PartText*. *FullText* is safe but highly inefficient, and *PartText* is efficient but makes the users vulnerable to inference attacks.

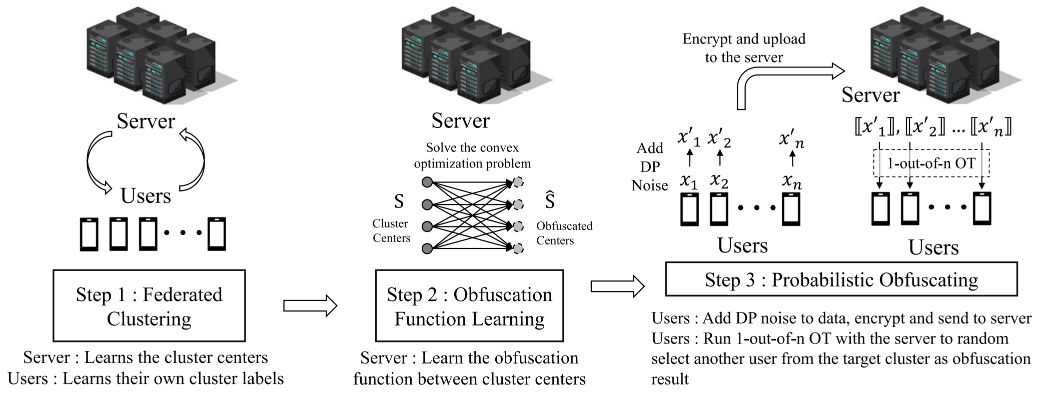
3.3 Problem Formulation

The comparsion in Section 3.2 shows that a new solution is urgently required to balance the efficiency and privacy preservation in federated MF. Assume we have n users, m items, and the rating information $r_{ij} : (i,j) \in \mathcal{M}$. To better model the information leakage, we denote $X \in \{0,1\}^{n\times m}$ as the rating-pair matrix, where the value of x_{ij} is:

$$x_{ij} = \begin{cases} 1, & if\ (i,j) \in \mathcal{M} \\ 0, & otherwise \end{cases} \tag{6}$$

Algorithm 2 shows the users' local update function according to the rating-pair matrix X.

Problem Formulation: Because of HE's nice property, the server cannot discriminate the encrypted zero gradients from the encrypted normal gradients in Algorithm 2. Thus, the only leaked information is the rating-pair matrix X. Given the initial rating-pair matrix X, which contains the



Server
Users
Step 1 : Federated Clustering
Server : Learns the cluster centers
Users : Learns their own cluster labels
Server
Solve the convex optimization problem
S
Cluster Centers
Ŝ
Obfuscated Centers
Step 2 : Obfuscation Function Learning
Server : Learn the obfuscation function between cluster centers
Encrypt and upload to the server
Server
Add DP Noise
x'_1 x'_2 x'_n
x_1 x_2 x_n
Users
$[\![x'_1]\!], [\![x'_2]\!] \dots [\![x'_n]\!]$
1-out-of-n OT
Users
Step 3 : Probabilistic Obfuscating
Users : Add DP noise to data, encrypt and send to server
Users : Run 1-out-of-n OT with the server to random select another user from the target cluster as obfuscation result

Fig. 2. Framework overview.

ALGORITHM 2: User Update According to Rating-pair Matrix X

```
User-i's local update:
Download $\{[\![v_j]\!] | where\ x_{ij} = 1\}$ from server
Decrypt and get $\{v_j | where\ x_{ij} = 1\}$
$LocalGradient \leftarrow \{0\}^d$, where $d$ is the dimensionality of user and item vector
for $j \in \{j | where\ x_{ij} = 1\}$ do
    if $(i, j) \in \mathcal{M}$ then
        $LocalGradient \leftarrow LocalGradient \oplus -2v_j(r_{ij} - \langle u_i, v_j \rangle)$
        $RemoteGradient_{ij} \leftarrow -2\gamma u_i(r_{ij} - \langle u_i, v_j \rangle) + 2\lambda v_j$
    else
        $RemoteGradient_{ij} \leftarrow \{0\}^d$
    end if
end for
Local updates: $u_i^t = u_i^{t-1} - \gamma(LocalGradien + 2\lambda u_i)$
Encrypt and send $\{[\![-\lambda RemoteGradient_{ij}]\!] | where\ x_{ij} = 1\}$ to the server, where $\lambda$ is the learning rate
```

real users' rating pairs, we want to obfuscate X to $\hat{X}$, such that $\hat{X}$ protects users against inference attacks, and the distortion between $\hat{X}$ and X is also bounded to maintain the utility of $\hat{X}$.

4 METHOD

In this section, we introduce EIFedMF to protect users against the inference attacks. The key idea is that we learn an obfuscation function to obfuscate users' rating-pair matrix, such that the mutual information between the distorted result and the private attribute is minimized given the data distortion budget, which guarantees the utility of the obfuscated data. During learning the obfuscation function, we use methods like clustering to reduce the problem complexity. Figure 2 shows the framework of EIFedMF. Briefly, we first perform a federated clustering according to the rating-pair matrix. Then, the server learns the obfuscation function between the clusters by solving a constrained convex optimization problem. Eventually, all the users run a secure probabilistic obfuscating protocol to generate obfuscation results.

4.1 Reduce the Information Leakage

Definition 4.1. The average information leakage of a set of input X regarding a private target Y is given by the $I(X; Y)$, which is the mutual information between X and Y [11, 34].

In order to protect users against inference attacks, we want to reduce the mutual information between the obfuscated rating-pair matrix $\hat{X}$ and users' private attribute Y (e.g., gender):

$$I(\hat{X}|Y) = \sum_{\hat{x} \in \hat{X}, y \in Y} p(\hat{x}, y) log \left(\frac{p(\hat{x}, y)}{p(\hat{x})p(y)} \right) \tag{7}$$

Given the rating-pair matrix X, we use an obfuscation function $p_{\hat{X}|X}$ to generate $\hat{X}$. Then the joint probability $p(\hat{x}, y)$, and the marginal probability $p(\hat{x})$ can be represented as:

$$\begin{aligned} p_{\hat{X},Y}(\hat{x}, y) &= \sum_{x \in X} p_{\hat{X}|X}(\hat{x}|x) p_{X,Y}(x, y) \\ p_{\hat{X}}(\hat{x}) &= \sum_{x \in X, y \in Y} p_{\hat{X}|X}(\hat{x}|x) p_{X,Y}(x, y) \end{aligned} \tag{8}$$

Combined with the above joint and marginal probability, the mutual information between the obfuscated rating-pair matrix $\hat{X}$ and the private attribute Y is:

$$\begin{aligned} I(\hat{X}; Y) = &\sum_{\hat{x} \in \hat{X}, y \in Y} \left[\sum_{x \in X} p_{\hat{X}|X}(\hat{x}|x) p_{X,Y}(x, y) \right] \\ &\cdot log \left(\frac{\sum_{x' \in X} p_{\hat{X}|X}(\hat{x}|x') p_{X,Y}(x', y)}{p(y) \cdot \sum_{x'' \in X, y' \in Y} p_{\hat{X}|X}(\hat{x}|x'') p_{X,Y}(x'', y')} \right) \end{aligned} \tag{9}$$

For a given dataset with privacy attribute Y, we can empirically determine $p_Y(y)$ and $p_{X,Y}(x, y)$. By minimizing the mutual information between the $\hat{X}$ and Y, the information leakage is reduced. Thus, we do not specify the inference attack methods, and any kind of inference attack should be rendered weak on $\hat{X}$.

4.2 Bound the Data Distortion

To guarantee the obfuscation results' utility in providing personalized recommendations and controling the system's efficiency overhead, we bound the data distortion between $\hat{X}$ and X. Since $X \in \{0, 1\}^{n \times m}$ and $\hat{X} \in \{0, 1\}^{n \times m}$, the discrepancy between X and $\hat{X}$ can be categorized into two situations:

- $x_{ij} = 0, \hat{x}_{ij} = 1$: User-i did not give a rating score for item-j because $x_{ij} = 0$, but needs to upload encrypted zero gradients for item-j because $\hat{x}_{ij} = 1$. It brings efficiency overhead because users need to encrypt extra zero gradients and upload them to the server.
- $x_{ij} = 1, \hat{x}_{ij} = 0$: User-i rated item-j because $x_{ij} = 1$, but does not need to upload gradients for item-j because $\hat{x}_{ij} = 0$. Such obfuscation brings recommending loss because the rating information for item-j is discarded.

The recommendation loss can be quantified by the amount of discarded data (i.e., where $x_{ij} = 1$ and $\hat{x}_{ij} = 0$), and it increases when more data are dismissed. The efficiency overhead can be evaluated by measuring the number of increased items for which users need to provide gradients. Given one user's rating-pair vector x_i and the obfuscated $\hat{x_i}$, we use Equation (10) to measure the recommendation loss, and use Equation (11) to quantify the time consumption overhead.

$$dist_{rec}(x_i, \hat{x}_i) = \frac{||x_i|| - ||x_i - \hat{x}_i||_{x_i > \hat{x}_i}}{||x_i||} \tag{10}$$

$$dist_{eff}(x_i, \hat{x}_i) = \frac{||\hat{x}_i|| - ||x_i||}{||x_i||} \tag{11}$$

where $||\cdot||$ is the L1-norm operation, $||\cdot||_{cond}$ is the L1-norm of elements at positions where *cond* is True. For example, if $a = [1, 0, -1, 1]$, then $||a||_{a>0} = 2$.

4.3 Obfuscation Function Learning

Combining the optimization target of reducing the mutual information between the obfuscated rating-pair matrix $\hat{X}$ and the private data Y, and the constraint of bounding the data distortion between $\hat{X}$ and X to maintain the utility, we can learn an obfuscation function $p_{\hat{X}|X}$ through the following constrained optimization:

$$\begin{aligned} min_{p_{\hat{X}|X}} \quad & I(\hat{X}; Y) \\ s.t. \quad & 0 \le p_{\hat{X}|X}(\hat{x}|x) \le 1 \\ & \sum_{\hat{x}} p_{\hat{X}|X}(\hat{x}|x) = 1 \\ & E_{\hat{X},X}(dist_{rec}(x,\hat{x})) \le \alpha \\ & E_{\hat{X},X}(dist_{eff}(x,\hat{x})) \le \beta \end{aligned} \tag{12}$$

Where α and β are two hyperparameters controlling the utility of $\hat{X}$ in recommendation and efficiency.

Theoretically, if we have n users and m items, we could obfuscate each user's rating-pair vector x into 2^m possible $\hat{x}$ according to the permutations and combinations theory, e.g., if we have two items, then $\hat{x} \in \{[0,0],[0,1],[1,0],[1,1]\}$. The complexity of learning the obfuscation function $p_{\hat{X}|X}$ is $o(n \cdot 2^m)$, which is too complicated because a real-world recommender system usually has lots of users and items.

To reduce the problem complexity, we turn to obfuscate one user's rating-pair vector to the one of another user instead of going through all 2^m possible results. Then the complexity of finding an optimal $p_{\hat{X}|X}$ reduces from $o(n \cdot 2^m)$ to n^2.

Although the problem is largely simplified to $O(n^2)$, it's complexity still grows quadratically with the number of users. To further reduce the complexity, we cluster the users into a limited and fixed number of groups. Then we learn the obfuscation function between clusters instead of individual users. Specifically, we first cluster users into k sets according to their rating pairs (i.e., the rating-pair matrix X). Then we learn the obfuscation function between different groups of users using the cluster centroids $S = \{s_0, s_1, \ldots, s_k\}$. The learning problem is represented in Equation (13).

$$\begin{aligned} min_{p_{\hat{S}|S}} \quad & I(\hat{S}; Y) \\ s.t. \quad & 0 \le p_{\hat{S}|S}(\hat{s}|s) \le 1 \\ & \sum_{\hat{s}} p_{\hat{S}|S}(\hat{s}|s) = 1 \\ & E_{\hat{S},S}(dist_{rec}(s,\hat{s})) \le \alpha \\ & E_{\hat{S},S}(dist_{eff}(s,\hat{s})) \le \beta \end{aligned} \tag{13}$$

It has been proved by Calmon et al. [11] that the constrained optimization in Equation (12) is a convex optimization problem, so is the problem in Equation (13). We solved the optimization using a dual minimization procedure, which is similar to the Arimoto-Blahut algorithm [9], by starting at a fixed marginal probability $p_{\hat{X}}$, solving the convex optimization, and updating the marginal probability at each step.

4.3.1 Federated Clustering. User clustering is indispensable in EIFedMF to reduce the optimization complexity. However, existing clustering methods require the data to be collected in one place, which cannot satisfy the requirements of privacy preservation. Thus, we propose a federated clustering algorithm to meet our requirement. In particular, EIFedMF is based on the k-means [33] algorithm.

ALGORITHM 3: Federated k-means Algorithm

```
Initialize: Server holds the public key, initializes cluster centers $C \in \{0,1\}^{K \times M}$, where $K$ is the number of clusters and $M$ is the number of items. Users hold the secret key.
Output: Converged cluster centers $C$.
Server send encrypted centers $[\![C]\!]$ to all users
for $User_i \leftarrow User_1$ to $User_N$ do
    Download $[\![C]\!]$ and decrypt
    Decide local cluster label $g_i = argmin_j ||x_i - C_j||_2^2$
    Prepare a one-hot-vector for counting the number of users in each cluster: $D_i = \{0\}^K, D_i[g_i] \leftarrow 1$
    Encrypt and upload $[\![D_i]\!]$
end for
Server: Compute $[\![D]\!] = \sum_i [\![D_i]\!]$, and return $[\![D]\!]$ to all users
for $User_i \leftarrow User_1$ to $User_N$ do
    Download $[\![D]\!]$ and decrypt
    Compute local contribution on cluster center: $C_i \leftarrow \{0\}^{k \times M}, C_i[g_i] \leftarrow \frac{x_i}{D[g_i]}$
    Encrypt and upload $[\![C_i]\!]$
end for
Server: Compute the new centers $[\![C]\!] \leftarrow \sum_{i=1}^{N} [\![C_i]\!]$
Repeat the algorithm until the cluster centers are converged, which can be monitored by a selected user, who will also send the final cluster centers to the server.
```

Algorithm 3 shows the federated k-means algorithm, which guarantees that the server only learns the final converged cluster centers. The users' cluster labels are only known to themselves.

After obtaining the cluster centers, the server will solve the convex optimization in Equation (13) and send the obfuscation function $p_{\hat{S}|S}$ to each user.

4.4 Probabilistic Obfuscation

Since the obfuscation function is learned based on clusters, we still need to fill in the gap between clusters and users to perform obfuscation on individual users. The key idea is that we firstly obfuscate one user from his own cluster s to another cluster s', then randomly select one user's data from cluster s' as the obfuscation result.

However, the above solution brings two kinds of information leakage: (1) Users' data are directly sent to the others, which brings serious privacy data leakages. (2) The server could inspect the mapping of obfuscation results (e.g., user-i finally picked user-j's data), because there is usually no direct connection between users in the recommender system and their inner communication needs the server as an assistant, who could see the result of random selection and trace it back to the real user of the obfuscated data.

In order to solve the above privacy issues, we use **differential privacy (DP)** and **oblivious transfer (OT)** to design a secure model for each user to obtain the obfuscation result safely. And it contains the following three steps:

- Step 1: All the users add DP noise to local data x_i getting x'_i, then encrypt and send $[\![x'_i]\!]$ to the server.
- Step 2: Get the local cluster s_i, obfuscate to another cluster $\hat{s}$ according to the obfuscation funtion $p_{\hat{S}|S}$.

- Step 3: Perform a 1-out-of-n OT with the server to randomly get a user-r from cluster $\hat{s}$, then use x'_r as the obfuscation result.

Next, we will introduce how the DP and OT are fitted into the system to reduce the information leakage.

4.4.1 Hide Users Through Differential Privacy. We model the random selection as a funcrework for better clarity.tion f, then add noise to the output of f inside each cluster. Such that the randomized output on each cluster satisfies the definition of ϵ-differential privacy (ϵ-DP). And the probability that we can specify whether the cluster containers one particular user is bounded.

Theorem 4.2. *Denote dateset D as a subset of users' data from one cluster. Based on D, we build dataset D' by randomly drop or add one user from the same cluster. Denote function $f : D \rightarrow D_i$ as the random pick function. Define function $A(D) = (f(D) + N) \ mod \ 2$, where $N \in \{0,1\}^m$ is discreate noise. And we generate N using Equation (14), where $1 \le i \le m$. We set $b = \frac{\Delta f}{\epsilon}, \Delta f = max_{x \in D, y \in D'} \sum_{i=1}^{m} |x_i - y_i|$, then function A satisfies the definition of ϵ-DP.*

$$Pr(N_i = 0) = \frac{1}{1 + e^{-\frac{1}{b}}}, Pr(N_i = 1) = \frac{e^{-\frac{1}{b}}}{1 + e^{-\frac{1}{b}}} \tag{14}$$

Proof. Assume $f(D) = (x_1, \ldots, x_m)^T$ and $f(D') = (y_1, \ldots, y_m)^T$. Denote $O = (z_1, \ldots, z_m)$ as a possible output of function A. Then we have:

$$\begin{aligned} Pr[A(D) = O] &= \prod_{i=1}^{m} \frac{1}{1 + e^{-\frac{\epsilon}{\Delta f}}} e^{-\frac{\epsilon}{\Delta f}|x_i - z_i|} \\ Pr[A(D') = O] &= \prod_{i=1}^{m} \frac{1}{1 + e^{-\frac{\epsilon}{\Delta f}}} e^{-\frac{\epsilon}{\Delta f}|y_i - z_i|} \\ \frac{Pr[A(D) = O]}{Pr[A(D') = O]} &= \frac{\prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\epsilon}{\Delta f}}} e^{-\frac{\epsilon}{\Delta f}|x_i - z_i|}}{\prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\epsilon}{\Delta f}}} e^{-\frac{\epsilon}{\Delta f}|y_i - z_i|}} \\ &= \prod_{i=1}^{m} e^{-\frac{\epsilon}{\Delta f}(|x_i - z_i| - |y_i - z_i|)} \\ &= e^{\epsilon \cdot \frac{\sum_{i=1}^{m}(|y_i - z_i| - |x_i - z_i|)}{\Delta f}} \end{aligned} \tag{15}$$

According to the triangle inequality $|a + b| - |a| \le |b|$, we have:

$$\begin{aligned} \frac{Pr[A(D) = O]}{Pr[A(D') = O]} &= e^{\epsilon \cdot \frac{\sum_{i=1}^{m}(|y_i - z_i| - |x_i - z_i|)}{\Delta f}} \\ &\le e^{\epsilon \cdot \frac{\sum_{i=1}^{m}(|y_i - x_i|)}{\Delta f}} \le e^{\epsilon} \end{aligned} \tag{16}$$

Thus, A satisfies the definition of ϵ-DP.

Algorithm 4 shows the method of jointly compute Δf among different users, while it keeps the data confidentiality. The challange is that $|\cdot|$ operation is not supported in existing HE algorithms, thus, we cannot just encrypt all the data and let the server do the computation. Given data x, y from two users, we want to compute $\sum_{i=1}^{m} |x_i - y_i|$. Our idea is splitting the computation into two steps (i.e., one step at the server, one step at the users) and using random shuffle to avoid information leakage:

ALGORITHM 4: Apply DP Noise (Compute Δf)

```
Initialize: $\epsilon$
Output: $\Delta f$ for each user
for $i \leftarrow 1$ to $N$ do
    $User_i$ encrypts $[\![x_i]\!]$ and cluster $[\![g_i]\!]$, send them to server
end for
for $i \leftarrow 1$ to $N$ do
    for $j \leftarrow 1$ to $N$ do
        Server computes $[\![D_{ij}]\!] = [\![x_i]\!] - [\![x_j]\!]$
        Server random shuffles $[\![D_{ij}]\!]$                    ▷ $[\![D_{ij}]\!]$ is a vector
    end for
    Server random shuffles $[\![D_i]\!]$ together with $[\![g]\!]$
    Server sends $[\![D_i]\!]$ and $[\![g]\!]$ to $user_i$
end for
for $i \leftarrow 1$ to $N$ do
    $User_i$ decrypts $[\![D_i]\!]$ and $[\![g]\!]$, $\Delta f \leftarrow 0$
    for $j \leftarrow 1$ to $N$ do
        for $k \leftarrow j + 1$ to $N$ do
            if $g_j = g_k = g_i$ then                    ▷ $g_i$ is the cluster of $user_i$
                $dist = \sum_{d=1}^{m} |D_{jk}^{d}|$
                if $\Delta f < dist$ then $\Delta f = dist$
                end if
            end if
        end for
    end for
    $User_i$ gets $\Delta f$, and applies DP noise locally
end for
```

ALGORITHM 5: Secure Random Select Using OT

```
for $i \leftarrow 1$ to $N$ do
    $User_i$ encrypt $[\![x_i]\!]$ and cluster $[\![g_i]\!]$, send them to server
end for
Server hold $[\![x_i]\!]$ as the messages and send $[\![g]\!]$ to all the users
for $i \leftarrow 1$ to $N$ do
    $User_i$ decrypt $[\![g]\!]$ and random select one index $b_i$ such that $g_{b_i} = g_i$.
    Using $b_i$, $user_i$ performs an 1-out-of-n OT with the server, and get $x_{b_i}$.
end for
```

- Step 1: Server compute $[\![d]\!] = [\![x]\!] - [\![y]\!]$, random shuffles $[\![d]\!]$ and sends it to the corresponding user. Here $[\![d]\!] \in \{0, 1\}^m$, and the random shuffle prevents user x from recovering user y's data through $y = x - d$.
- Step 2: User decrypts $[\![d]\!]$, and computes $\sum_{i=1}^{m} |d_i|$. Assuming the current user holds x, it cannot recover y because $[\![d]\!]$ is already shuffled by the server.

4.4.2 1-out-of-n OT with the Server. In order to blind the server with which user is finally chosen as the obfuscation result, we use 1-out-of-n oblivious transfer technology which guarantees that the server learns nothing about the users' choices and the users only learn the data he chooses. Algorithm 5 shows the detail of the method.

4.5 Privacy Analysis

Privacy-preservation is an essential requirement for federated learning frameworks. Thus, the framework should be carefully designed to minimize private data leakage. The federated matrix factorization is a typical horizontal federated learning scenario in which the adversary is the semi-honest server. Thus, the framework needs to protect the users from the server. Here we analyze the privacy-preservation of EIFedMF by summarizing what information the server receives and how much knowledge the server can learn.

As shown in Figure 2, EIFedMF has a workflow of three parts to select a set of uploading items such that the efficiency and privacy protection regarding the inference attacks are jointly optimized. Next, we list the valid information (i.e., the information in plain-text and the encrypted messages are excluded) that the server receives in the EIFedMF's workflow:

- Part 1 Federated KMean Clustering: The only valid information that the server receives is the cluster centers. The cluster centers leak no private information because it represents the character that belongs to a group of users, which is not private because privacy is defined as information that could be linked to a natural person.
- Part 2 Obfuscation Function Learning: In this step, the server learns the obfuscation function based on the clusters' centers. Thus, the valid information that the server learns is the obfuscation function, which contains no more knowledge than the cluster centers because the functions are derived from them. Since the cluster centers leak no private information, the obfuscation function also leaks nothing private.
- Part 3 Probabilistic Obfuscating: During the probabilistic obfuscation, end-to-end privacy-preserving techniques, e.g., homomorphic encryption and oblivious transfer, are adopted. Thus, the server gets no valid message and learns zero knowledge about users' private data. It is worth noting that each user randomly picks another user's data as obfuscation in this step, and we have used differential privacy to guarantee that the privacy leakage between users is bounded.

To summarize, EIFedMF can protect the users from inference attacks, and leaks no private information during the obfuscating items selection process. Meanwhile, EIFedMF can protect the users from inference attacks.

5 EVALUATION

5.1 Experiment Settings

We evaluate EIFedMF on two real-world datasets. The first dataset is collected from MovieLens that contains ratings of 1,682 movies made by 943 users,[1] and we focus on protecting users' gender, age, and occupation in this dataset. Another dataset is the users' check-in records at different **POIs (point of interest)** in NYC [35–37]. The POIs are treated as items, and users' gender needs to be protected in this dataset. The raw dataset contains check-in records made by 18,201 users at more than 500K POIs. We choose the top 10,000 POIs with the highest number of visiting records and users with more than 50 visiting records. After the preprocessing, the CheckIn dataset contains records of 10,000 POIs made by 9,233 users.

Table 1 shows the processing of attribute labels. Some attributes are naturally discrete, such as gender, and we directly use them as attribute labels. Some attributes are continuous, such as age, and we manually transfer them into discrete values.

In the federated user clustering, we set the $k = 10$ (i.e., the number of clusters) for both MovieLens and CheckIn datasets. The α and β in Equation (13) are set to 0.3 and 1.0, respectively. We

[1] https://www.kaggle.com/prajitdatta/movielens-100k-dataset.

Table 1. Processing of Attributes

Attribute	#Classes	Values
Gender	2	{Male, Female}
Age	5	$\{\leq 18, (18, 30], (30, 40], (40, 50], 50<\}$
Occuption	21	administrator, artist, doctor, etc.

set $\epsilon = 0.01$ when applying the differential privacy. We use SEAL-CKKS encryption [8, 28] as the homomorphic encryption method, and the poly modulus degree is set to 8192. The bandwidth of the communication is 1 *Gb/s*. All the experiments are performed on a Linux server with 3.6GHz 8-core CPU and 32GB RAM. The programming language is Python, and we use SEAL-Python[2] in the implementation, which is a python binding for the Microsoft SEAL library.

It is worth noting that EIFedMF protects users from inference attacks through an optimization that reduces the mutual information between the uploaded items and a certain private target (i.e., one attribute). In reality, users usually have more than one private attribute. EIFedMF can be extended to a multi-attribute version by incorporating all the attributes in the optimization, in which different targets can be equally treated or assigned different weights. Here we consider two settings of optimization targets in the evaluation.

- (EIFedMF-Single) Protect a single attribute, e.g., gender.
- (EIFedMF-All) Protect all the attributes in one dataset, e.g., gender, age, and occupation.

To demonstrate the effectiveness of EIFedMF, we compare with the following baselines in the evaluation:

- **PartText Solution (PT) [7]:** Users use real rating-pair without any protection.
- **FullText Solution (FT) [7]:** Users use full rating-pair matrix, i.e., uploading gradients for all items.
- **Random Increase (RI-α) [34]:** Users randomly pick α percent of non-rated items as obfuscation to resist the inference attacks.
- **Random Flip (RF-β) [34]:** Users randomly flip the values of the expose matrix X with probability β.

5.2 Evaluation on Privacy-preservation

To evaluate the privacy-preservation of EIFedMF, we have performed inference attack experiments using three types of machine learning models: **Naive Bayesian (NB)**, **support vector machine (SVM)**, and **gradient boosting decision tree (GBDT)**. A 10-fold cross-validation is used, and the average results are reported in Table 2. A solution yields better privacy preservation if it has lower inference attack accuracy.

Compared with *PartText*, which has no protection to inference attacks, EIFedMF significantly reduces the inference attack accuracy. In particular, EIFedMF reduces the inference attack accuracy by 16.7% compared with *PartText* regarding the MovieLen's age attribute when the adversaries use the GBDT attack model. Furthermore, compared with the other three baselines, i.e., RI, RF, and NB, EIFedMF consistently yields lower attack precision, which shows the effectiveness of EIFedMF. We also observe the NB is a relatively weak attack model compared with SVM and GBDT, thus easier to defense, e.g., the RI and RF methods do a good job in resisting the NB attack but not that effective when dealing with SVM and GBDT models. In fact, none of the baseline methods is effective when attack using the SVM model, but EIFedMF can reduce SVM's attack accuracy up to 10%.

[2] https://github.com/Huelse/SEAL-Python.

Table 2. Inference Attack Accuracy

Attribute	Model	FT	PT	RI-0.1	RI-0.3	RI-0.5	RI-0.7	RI-0.9	RF-0.1	RF-0.3	RF-0.5	RF-0.7	RF-0.9	EIFedMF Single	EIFedMF All
MovieLens Gender	NB		0.696	0.671	0.687	0.683	0.693	0.711	0.656	0.666	0.664	0.685	0.647	**0.575**	0.614
	SVM	**0.588**	0.722	0.711	0.711	0.711	0.711	0.711	0.711	0.711	0.711	0.711	0.711	0.662	**0.654**
	GBDT		0.733	0.711	0.700	0.699	0.699	0.689	0.705	0.704	0.690	0.701	0.710	0.640	**0.640**
MovieLens Age	NB		0.421	0.408	0.418	0.384	0.405	0.418	0.397	0.380	0.361	0.389	0.393	**0.289**	0.300
	SVM	**0.277**	0.449	0.424	0.419	0.418	0.418	0.418	0.423	0.418	0.418	0.418	0.421	0.380	**0.349**
	GBDT		0.476	0.441	0.407	0.403	0.390	0.396	0.431	0.401	0.355	0.404	0.420	0.335	**0.309**
MovieLens Occupation	NB		0.209	0.192	0.193	0.186	0.190	0.207	0.189	0.172	0.160	0.211	0.197	**0.104**	0.111
	SVM	**0.094**	0.231	0.209	0.208	0.208	0.208	0.208	0.209	0.208	0.208	0.208	0.209	0.186	**0.173**
	GBDT		0.192	0.212	0.183	0.175	0.179	0.164	0.193	0.193	0.164	0.191	0.183	0.132	**0.117**
NYC CheckIn Gender	NB		0.688	0.664	0.666	0.670	0.673	0.704	0.649	0.661	0.662	0.673	0.702	**0.608**	–
	SVM	**0.582**	0.710	0.703	0.703	0.703	0.703	0.703	0.703	0.703	0.703	0.703	0.703	**0.661**	–
	GBDT		0.721	0.675	0.700	0.698	0.698	0.678	0.681	0.699	0.699	0.700	0.682	**0.653**	–

The lower the attack accuracy, the better privacy preservation. The lowest two attack results are bolded.

The *FullText* solution leaks no extra private data to the semi-honest server, thus is naturally free from inference attacks. However, the server still can make random guesses based on some prior knowledge, e.g., 70% of the users are male. Table 2 shows the attack accuracy using random guess in *FullText*, which could be treated as the lower bound of the inference attack accuracy. Compared with *FullText*, EIFedMF achieves comparable privacy preservation, and the inference attack accuracy is only 4.2% higher on average. Although *FullText* achieves the best protection against inference attacks, it enforces users to upload gradient for all items, which is very low efficient and not practical. We compare EIFedMF with *FullText* regarding efficiency in Section 5.4, and the results show that EIFedMF is much more efficient than *FullText*.

5.3 Evaluation on Data Utility

EIFedMF guarantees the data utility by setting a data distortion budget between the real data and the obfuscation results during the optimization. To evaluate the data utility, we have built a matrix-factorization-based recommender system and use the recommendation error to evaluate the data utility. **Root mean square error (RMSE)** is used as the metric. Briefly, 10-fold cross-validation is used in the matrix factorization, 90% of the data are used for training, and 10% for testing. The average results are presented in Table 3.

The *FullText* and *PartText* have no data utility loss, since all the rated items are uploaded. Compared with *FullText* and *PartText*, the recommendation error of EIFedMF is about 2.4% larger, i.e., 2.4% of data utility loss. The baseline method RI has no utility loss because it only uploads more zero gradients for the unrated items and leaves the rated items untouched. The baseline method RF also brings utility loss, which increases with the flipping probability. It is worth noting that although RI and RF with flipping probability smaller than 0.5 have better data utility than EIFedMF, they cannot protect the users against inference attacks very well, especially when using strong attack models, such as GBDT and SVM. In contrast, EIFedMF reduces up to 16.7% attack accuracy compared with *PartText* and only brings 2.4% utility loss.

Figure 3 shows the recommendation error of EIFedMF under different privacy budgets, i.e., when there is no privacy protection and $\epsilon = 10, 1, 0.1, 0.01$. Compared with no privacy protection, EIFedMF does bring some utility loss but significantly enhances privacy. We can observe that EIFedMF's recommendation error firstly shows a small increase, then slightly fluctuates with the decrease of ϵ. Thus, the value of ϵ, changing from 10 to 0.01, does not significantly influence the EIFedMF's data utility. One reason is that the DP noise added on the data is only semi-decided by ϵ, and another factor Δf is usually very large in recommender application due to the users' heterogeneity making the data utility less sensitive to ϵ.

Table 3. Recommendation Error (RMSE) on MovieLens

Method	FT	PT	RI-(0.1 ∼ 0.9)	RF-0.1	RF-0.3	RF-0.5	RF-0.7	RF-0.9	EIFedMF Gender	EIFedMF Age	EIFedMF Occupation	EIFedMF All
RMSE	0.950	0.950	0.950	0.953	0.960	0.974	1.014	1.223	0.975	0.974	0.975	0.973

The lower the error, the better the data utility.

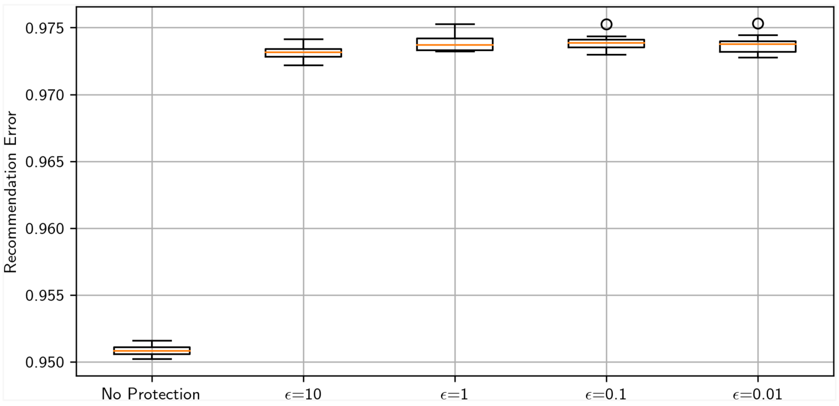


Fig. 3. Data utility under different privacy budgets on MovieLens data. The lower the recommendation error (RMSE), the higher the utility.

5.4 Evaluation on Efficiency

The federated matrix factorization basically has two procedures: (1) determining which items' gradients should be uploaded, i.e., uploading real encrypted gradients for rated items and encrypted zero gradients for non-rated items, (2) performing a standardized secure federated matrix factorization training [7], in which multiple clients jointly update a group of global item vectors.

The first step (i.e., selecting the items) in *FullText* and *PartText* are relatively simple, which consequently brings disadvantages to the second step. *FullText* requires the users to upload gradients for all items, which brings large efficiency overhead to the second step. *PartText* only lets the users upload the rated items, which leaks too much information and makes the users vulnerable to a broader set of inference attacks in the second step. EIFedMF mainly focuses on optimizing the item selection process in step 1, such that the efficiency is improved compared with *FullText* and the leaked information is less than *PartText*.

The efficiency evaluation of EIFedMF should include both step 1 and step 2, however, we will show that the time consumption in step 1 of EIFedMF is much less than step 2. Thus, we only compare the efficiency on step 2. According to the framework presented in Figure 2, EIFedMF has a workflow of three parts in step 1, in which the federated clustering costs the most time because it requires multi-rounds training to find the cluster centers. However, in the federated KMeans clustering (i.e., Algorithm 3), each user only uploads a single vector in each round of training. In step 2, each user will need to upload a group of item vectors (i.e., an encrypted matrix) which costs much more time compared with step 1. To simplify the comparison, we only consider step 2 (i.e., the most time-consuming step in federated matrix factorization) in the efficiency evaluation.

In the second step of federated matrix factorization, the time consumption is linearly correlated with the number of uploading items. Thus, we compare the averaged number of uploading items between EIFedMF, *FullText*, and *PartText* solutions. Figure 4 shows the results. EIFedMF can reduce 49.8% and 39.7% numbers of uploading items compared with *FullText* solution in MovieLens and CheckIn datasets, respectively. Thus, the system efficiency is significantly improved.

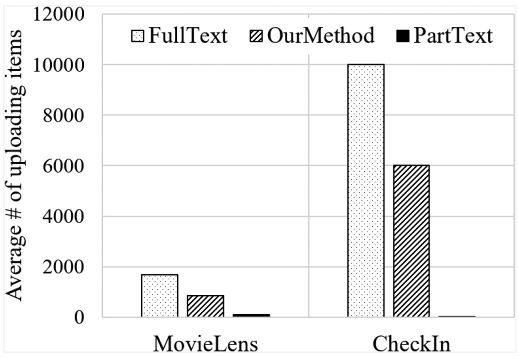


Fig. 4. Efficiency evalution.

6 DISCUSSION

6.1 Generality of Our Method

In the recommender system, the users' rating matrix is usually very sparse. The recommender system needs to train the machine learning model using the non-empty values in the user rating matrix, which are formed as a list of user-item rating pairs. Most of the existing recommender system studies [10, 12, 18, 21, 32, 38] adopt the user and item profiles in the modeling or explore higher-order features [16] to improve the model performance. During training, these models are usually partially updated using each data record. The user and item latent vectors are updated when there exist rating records. The weights of higher-order features are updated when the corresponding rating values are not empty. Consequently, if we put these models in the federated learning scenario, all users jointly update one global model. They will face the same issue with FedMF [7], in which the users are not required to update the whole model to improve efficiency, but the server could perform an inference attack on users according to which parts of the model are updated.

As we mentioned in Section 5.4, EIFedMF has two steps: (1) determining which items' gradients should be uploaded, and (2) performing a standardized secure federated matrix factorization training. EIFedMF mainly enhances the system in the first step by carefully selecting a group of items such that the efficiency and privacy protection are jointly optimized. The second step (i.e., the federated matrix factorization training) of EIFedMF is untouched and inherited from [7]. Thus, the first step of EIFedMF is not strictly coupled with FedMF [7], and could be easily combined with other models [10, 12, 16, 18, 21, 32, 38] to protect users from inference attacks in the federated learning scenario.

6.2 EIFedMF's Technical Selection

To avoid private data leakage, we have adopted different privacy-preserving techniques in EIFedMF: homomorphic encryption, differential privacy, and oblivious transfer. Next, we will elaborate on why these three techniques are needed and whether they are substitutable.

- Homomorphic Encryption: In the federated KMeans clustering of EIFedMF, the users jointly update the cluster centers, during which many pieces of private information are submitted to the server, by which the information is aggregated to produce the new cluster centers. Thus, HE is adopted in EIFedMF to conceal the private data. HE is not the only choice here and could be substituted by other privacy-preserving aggregation methods, e.g., SecureAggregation [6]. Compared with HE, the SecureAggregation has lower computation complexity but brings large communication overhead. In practice, these two techniques could be chosen according to the system resources, i.e., choose HE if the system has rich computing resources and choose SecureAggregation if the system has rich networking resources.

- Differential Privacy: During the probabilistic obfuscation, all users randomly select data from another cluster as the obfuscation results to protect themselves from inference attacks, which means that one user's item visiting records could be leaked to another user. Although the obfuscation results are randomly selected from clusters (i.e., a group of users), the users that receive the results still have a chance to link that information to a natural person. Thus, we have proposed to use differential privacy to bound the probability of specifying whether one user is inside one cluster or not from the obfuscation results. To the best of our knowledge, differential privacy cannot be substituted because it is the only technique to theoretically bound the privacy data leakage regardless of adversaries' prior knowledge when revealing the raw data is necessary.
- Oblivious Transfer: The oblivious transfer is adopted in EIFedMF to blind the server with users' obfuscation process, i.e., the mapping between users and the obfuscation results. If the server knows the mapping correlation, e.g., user-i finally uses user-j's visiting records as obfuscation result, then the server can directly use user-i's information to attack user-j. Thus, according to the scenario, the oblivious transfer is the ideal technique to keep the obfuscating process oblivious to the server.

7 RELATED WORKS

The privacy-preserving recommender system is currently a hot research topic, a lot of works have demonstrated the importance of protecting users' privacy in the RecSys [3, 15, 19, 31]. Koren et al. [23] first applied matrix factorization into the recommender system, and it helped them win the Netflix competition. Afterward, lots of variational models were proposed to improve the precision and robustness, such as SVD++ [22], CMN [13], etc. Thus, many works focus on privacy issues in MFRecSys because MF is one of the most fundamental and essential methods in the recommender system. Existing works in secure matrix factorization can be categorized into three types: encryption methods, differential privacy methods, and federated methods.

Encryption methods: Data encryption is a traditional and intuitive method to protect users' privacy. Sungwook et al. [20] proposed a **fully homomorphic encryption (FHE)** secure matrix factorization framework. To reduce the large time consumption overhead, they merged FHE with **2-party computation (2PC)**. Thus, their framework requires a **crypto-service provider (CSP)** to work with the recommender server. Although the time efficiency is significantly improved, a trusted CSP is hard to find in reality, and the recommender server may collude with CSP which brings new privacy concerns. Shahriar et al. [4] proposed a secure framework for collaborative filtering using a combination of AHE and 2PC. In their framework, AHE works for the addition operation, and 2PC handles the multiply computation. They also require a trusted third party because of 2PC, thus has similar implementation problem and the new privacy concerns. Valeria et al. [26] proposed a secure matrix factorization framework using homomorphic encryption and garbled circuits, and they also require a trusted CSP in the system.

Compared with existing encryption methods [4, 20, 26], our framework does not require a third party, only a central recommender server is needed.

Differential privacy: Apart from encryption methods, differential privacy (DP) methods also have broad applications in the recommender system [5, 24, 29, 30]. DP protects users' privacy by adding noise into the rating data, which shows undesirable reduction in recommendation accuracy. Moreover, DP also can not ensure users' data confidentiality. Our solution also utilizes differential privacy to hide the users during the obfuscation.

Federated methods: Federated machine learning (FML) is a new technique for solving the data isolation problems. It enables different parties to do data mining tasks on their joint data without revealing any party's data to the others. FML is tailored for solving the privacy issues in the recommender system. Users are different parties and they try to build a recommender system without leaking personal data. Some pioneering works already show the feasibility of federated matrix factorization [2, 7]. Researchers in [2] built a **distributed matrix factorization (DMF)** system, where users directly send the intermediate results to the server. Afterwards, [7] proved that the intermediate results in DMF leaks users' privacy, and they applied HE to avoid the information leakage. However, neither of them considered the attribute inference attack issue. The server can still attack the users' attributes even after the HE is applied. To fill in this research gap, we propose a noval attribute-preserving federated matrix factorization framework that can protect users against attribute inference attacks.

8 CONCLUSION AND FUTURE WORK

Privacy protection is a fundamental problem in the recommender system. In this paper, we focus on federated matrix factorization, which is one of the most widely used technologies in recommender systems. We propose an obfuscation based method to protect users against inference attacks in the federated matrix factorization. Briefly, we learn an obfuscation function to obfuscate users' rating-pair matrix such that the mutual information between the obfuscated result and the private attribute is minimized under a certain distortion budget, which guarantees the utility of the obfuscation result. Our method is more practical compared with existing federated matrix factorization works [7]. In particular, it is more efficient than the $FullText$ solution and safer than the $PartText$ solution. We implement and test our method on real-world datasets. The results show that our method can reduce the inference attack accuracy by up to 16.7%, while maintaining the recommending performance and only brings 2.4% recommending loss.

In the future, we will continue working on the following problems to improve the practicality of our solution:

Extend to other models: We will verify the generality of the proposed privacy-preserving method, e.g., when using the deep learning models (e.g., DeepFM [17]). Intuitively, most federated learning scenarios that adopt matrix factorization technology will meet the inference attack issues, and our proposed privacy-preserving method could be applied to solve such problems.

Against more power attack models: When the server knows more data, e.g., users' social network, more powerful attack models could be applied. Thus, protecting users' privacy against these powerful models will be one of our essential research problems.

Stragglers and Dropouts: Recommender systems usually have a lot of users. The training process of federated matrix factorization is distributed and asynchronous, and a large number of users will bring notable system uncertainty. For example, some users may have a poor internet connection or insufficient computing power such that their update frequency is lower than others, and we call these users stragglers. The situation could be worse when some users are offline due to some reasons (e.g., out of battery), and we call the offline users dropouts. Stragglers and dropouts will affect the model's training convergence. Thus, we want to improve our method to minimize such influences.

REFERENCES

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 79.

[2] Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888.*

[3] Shahriar Badsha, Xun Yi, and Ibrahim Khalil. 2016. A practical privacy-preserving recommender system. *Data Science and Engineering* 1, 3 (2016), 161–177.

[4] Shahriar Badsha, Xun Yi, Ibrahim Khalil, and Elisa Bertino. 2017. Privacy preserving user-based recommender system. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1074–1083.

[5] Arnaud Berlioz, Arik Friedman, Mohamed Ali Kaafar, Roksana Boreli, and Shlomo Berkovsky. 2015. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 107–114.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[7] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Secure federated matrix factorization. *IEEE Intelligent Systems* (2020), 1–1. https://doi.org/10.1109/MIS.2020.3014880

[8] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 409–437.

[9] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (2. ed.)*. Wiley. http://www.elementsofinformationtheory.com/.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells (Eds.). ACM, 191–198. https://doi.org/10.1145/2959100.2959190

[11] Flávio du Pin Calmon and Nadia Fawaz. 2012. Privacy against statistical inference. In *50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012, Allerton Park & Retreat Center, Monticello, IL, USA, October 1-5, 2012*. IEEE, 1401–1408. https://doi.org/10.1109/Allerton.2012.6483382

[12] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 515–524. https://doi.org/10.1145/3209978.3209991

[13] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 515–524.

[14] Oded Goldreich. 2004. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press. https://doi.org/10.1017/CBO9780511721656

[15] Haiyan Guan, Hongyan Qian, and Yanchao Zhao. 2016. Location privacy protected recommendation system in mobile cloud. In *International Conference on Cloud Computing and Security*. Springer, 409–420.

[16] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 1725–1731. https://doi.org/10.24963/ijcai.2017/239

[17] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 173–182. https://doi.org/10.1145/3038912.3052569

[19] Weiming Huang, Baisong Liu, and Hao Tang. 2019. Privacy protection for recommendation system: A survey. In *Journal of Physics: Conference Series*, Vol. 1325. IOP Publishing, 012087.

[20] Sungwook Kim, Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, and Junbum Shin. 2016. Efficient privacy-preserving matrix factorization via fully homomorphic encryption. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 617–628.

[21] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 426–434. https://doi.org/10.1145/1401890.1401944

[22] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 426–434.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[24] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the Netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 627–636.

[25] Moni Naor and Benny Pinkas. 1999. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton (Eds.). ACM, 245–254. https://doi.org/10.1145/301250.301312

[26] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. 2013. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 801–812.

[27] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

[28] SEAL 2019. Microsoft SEAL (Release 3.3). Microsoft Research, Redmond, WA. https://github.com/Microsoft/SEAL.

[29] Yilin Shen and Hongxia Jin. 2014. Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *2014 IEEE International Conference on Data Mining*. IEEE, 540–549.

[30] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1770–1782.

[31] Cong Wang, Yifeng Zheng, Jinghua Jiang, and Kui Ren. 2018. Toward privacy-preserving personalized recommendation services. *Engineering* 4, 1 (2018), 21–28.

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 1235–1244. https://doi.org/10.1145/2783258.2783273

[33] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael S. Steinbach, David J. Hand, and Dan Steinberg. 2008. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* 14, 1 (2008), 1–37. https://doi.org/10.1007/s10115-007-0114-2

[34] Dingqi Yang, Bingqing Qu, and Philippe Cudré-Mauroux. 2018. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 3 (2018), 507–520.

[35] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[36] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2015. Participatory cultural mapping based on collective behavior in location based social networks. *ACM Transactions on Intelligent Systems and Technology* (2015). In press.

[37] Dingqi Yang, Daqing Zhang, Bingqing Qu, and Philippe Cudre-Mauroux. 2016. PrivCheck: Privacy-preserving check-in data publishing for personalized location based services. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 545–556.

[38] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan (Eds.). ACM, 311–319. https://doi.org/10.1145/3240323.3240343

Received March 2021; revised August 2021; accepted November 2021