Privacy-Preserving Multiview Matrix Factorization for Recommender Systems

Peihua Mai, Member, IEEE, and Yan Pang, Member, IEEE

Abstract—With an increasing focus on data privacy, there have been pilot studies on recommender systems in a federated learning (FL) framework, where multiple parties collaboratively train a model without sharing their data. Most of these studies assume that the conventional FL framework can fully protect user privacy. However, there are serious privacy risks in matrix factorization in federated recommender systems based on our study. This article first provides a rigorous theoretical analysis of the server reconstruction attack in four scenarios in federated recommender systems, followed by comprehensive experiments. The empirical results demonstrate that the FL server could infer users’ information with accuracy ≥ 80% based on the uploaded gradients from FL nodes. The robustness analysis suggests that our reconstruction attack analysis outperforms the random guess by > 30% under Laplace noises with b ≤ 0.5 for all scenarios. Then, the article proposes a new privacy-preserving framework based on a threshold variant of homomorphic encryption, privacy-preserving multiview matrix factorization (PrivMVMF), to enhance user data privacy protection in federated recommender systems. The proposed PrivMVMF is successfully implemented and tested thoroughly with the MovieLens dataset.

Impact Statement—The recommender system is one of the most successful and popular AI applications in the industry. Multi-view matrix factorization (MVMF) has been proposed as an effective framework for the cold-start recommendation, where side information is incorporated into matrix factorization. MVMF requires users to upload their personal data to a centralized recommender, which raises serious privacy concerns. A solution to address the privacy problem is federated learning. However, the conventional federated MVMF allows the exchange of plaintext gradients, which is susceptible to information leakage. There is a lack of thorough study of the privacy risk in conventional federated MVMF in previous research works. To fill this gap, this paper provides rigorous theoretical analysis and comprehensive experiments on the privacy threat in the traditional federated MVMF. The analysis and experiments demonstrate that the server could reconstruct users’ original information even when small amounts of noise are added to the gradients. Then we propose a PrivMVMF framework to address the information leakage problem successfully.

Index Terms—Data privacy, federated learning (FL), homomorphic encryption, recommender system.

I. INTRODUCTION

The recommendation system relies on collecting users’ personal information, such as purchase history, explicit feedback, social relationship, and so on. Recently, some laws and regulations have been enacted to protect user privacy, which places constraints on the collection and exchange of users’ personal data.

To protect user privacy, one way is to develop a recommendation system in federated learning (FL) framework that enables the clients to jointly train a model without sharing their data. In the FL setting, each client computes the updated gradient locally and sends the model update instead of the original data to a central server. The server then aggregates the gradients and updates the global model [1].

Collaborative filtering (CF) is one of the most effective approaches in recommendation systems [9], and matrix factorization (MF) is a popular technique in CF algorithms. MF decomposes a user-item interaction matrix into two low-rank matrices: 1) user latent factors and 2) item latent factors, which are used to generate the preference prediction [7]. The dense low-rank embeddings are better representations than the sparse vectors [8]. One disadvantage of MF-based recommendation is the cold-start problem: if an item or user has no rating information, the model cannot generate a latent factor representation for it. A solution to the cold-start issue is to incorporate side information, i.e., user and item attributes, into matrix factorization.

Various approaches have been proposed for centralized recommender systems [2], [3], [4], [5]. However, few studies have researched the topic in the federated setting. To the best of our knowledge, Flanagan et al. [6] is the first to propose a federated multiview matrix factorization (MVMF) to address this problem. However, this method assumed that the conventional FL framework could fully protect user privacy. Indeed, severe privacy risks exist in the federated MVMF recommender system, which is susceptible to server reconstruction attacks, i.e., the attack to recover users’ sensitive information.

To fill this gap, this article first provides a theoretical analysis of the privacy threat of the federated MVMF method. In theoretical analysis, we develop server reconstruction attacks in four scenarios based on different treatments of unobserved ratings and methods to update user latent factors. The empirical study results indicate that the original federated MVMF method could leak users’ personal information. Then, we design a privacy-preserving federated MVMF framework using threshold homomorphic encryption (HE) to enhance user data privacy protection in federated recommender systems.

The main contributions of this article are twofold as follows. 1) To the best of our knowledge, we are the first to provide a rigorous theoretical analysis of server reconstruction attacks with accuracy ≥ 80% under Laplace noises with b ≤ 0.5 for all scenarios. Then, the article proposes a new privacy-preserving framework based on threshold homomorphic encryption max, Member, IEEE (Cor—}
attacks in the federated MVMF recommender system. We also conducted comprehensive experiments, which show that the server could infer users’ sensitive information with accuracy > 80% using such attacks, and the attack is effective under a small amount of noise.

2) To overcome the information leakage problem, we propose PrivMVMF, a privacy-preserving federated MVMF framework enhanced with HE. The proposed framework has three advantages: 1) To balance the tradeoff between efficiency and privacy protection, it adopts a strategy in which some unrated items are randomly sampled and assigned a weight on their gradients. 2) To reduce complexity, it allocates some decrypting clients to decrypt and transmit the aggregated gradients to the server. 3) To ensure the security under collaboration between server and decrypting clients, we adopt a threshold variant of the Paillier encryption scheme, where a minimum number of clients is required to decrypt the ciphertext. A prototype of PrivMVMF is implemented and tested on the movielens dataset.

II. LITERATURE REVIEW

Federated Matrix Factorization: Federated recommender systems enable parties to collaboratively train the model without putting all data on a centralized server. Several federation methods for recommender systems have been introduced in recent works. Ammad-ud-din et al. [12] proposed a federated matrix factorization method for implicit feedback. Each client updates the user latent factor locally and sends back the item latent factor gradient to the server for aggregation and update. Duriakova et al. [10] presented a decentralized approach to matrix factorization without a central server, where each user exchanges the gradients with their neighbors. Lin et al. [11] provided a matrix factorization framework based on federated meta-learning by generating private item embedding and rating prediction model. The above works have not considered the cold-start recommendation. To address the problem, Flanagan et al. [6] devised a federated multiview matrix factorization based on the implicit feedback (e.g., clicks), where three matrices are factorized simultaneously with sharing latent factors.

Cryptographic Techniques in Federated Recommender Systems: Some studies used encryption schemes to develop privacy-preserving recommendation systems. Chai et al. [16] introduced FedMF, a secure federated matrix factorization framework. To increase security, each client can encrypt the gradient uploaded to the server with HE. Shmueli et al. [17] proposed multiparty protocols for item-based CF of vertical distribution settings. In the online phase, the parties communicate only with a mediator that performs computation on encrypted data, which reduces communication costs and allows each party to make recommendations independent of other parties. Meng et al. [18] employs local sensitive hash (LSH) to map users’ personal information into low-dimensional hash values. Although both [16] and our article adopt HE to enhance security, our work extends the method by adopting a threshold variant of the scheme, introducing decrypting clients and sampling of unrated items. The decrypting clients improve the efficiency of performing parameter updates, and the threshold homomorphic encryption requires the decryptors to jointly perform decryption to enhance security. The unrated items sampling strikes a balance between efficiency and privacy protection.

To the best of our knowledge, Flanagan et al. [6] is the first to devise a federated multiview matrix factorization to address the cold-start problem, where the users directly upload the plaintext gradients to the server, and no work has considered the information leakage from the gradients. This article first demonstrates the feasibility of the server reconstruction attack, and then proposes a framework to enhance privacy protection.

III. FEDERATED MVMF

The federated MVMF proposed by Flanagan et al. [6] is based on implicit feedback. In this section, we extend the framework to explicit feedback.

A. Notations

Table I lists the notations and their descriptions used throughout this article.

| Notation | Description |
|----------|-------------|
| n        | The number of users. |
| m        | The number of items. |
| lσ       | Dimension of user attributes. |
| lρ        | Dimension of item attributes. |
| R, rσ,j, rρ,i,j | Rating matrix. |
| X, xσ,a   | User feature. |
| U, uσ,a   | User feature latent factor. |
| P, pσ     | User latent factor. |
| Q, qj     | Item latent factor. |
| V, vσ,p   | Item feature latent factor. |
| c         | Uncertainty coefficient. |
| K         | Dimension of latent factor. |
| λ         | Regularization coefficient. |
| O        | Set of rated items for user i. |
| Y, yσ,a   | Item feature. |
| γ         | Learning rate. |
| β1, β2   | Exponential decay rate. |
| ε         | Small number. |

Multiview Matrix Factorization

Multiview matrix factorization is performed on the three data sources: 1) the rating matrix \( R_{n \times m} \), 2) the user attribute matrix \( X_{n \times lσ} \), and 3) the item content matrix \( Y_{m \times lρ} \), for \( n \) users with \( lσ \) features, and \( m \) items with \( lρ \) features. The decomposition of the three matrices is given as

\[
R \approx PQ^T, X \approx PU^T, Y \approx QV^T
\]

where \( P = P_{n \times K}, Q = Q_{m \times K}, U = U_{lσ \times K}, V = V_{lρ \times K} \) with \( K \) representing the number of latent factors. For \( P \) and \( Q \), each row represents the latent factors for each user and item, respectively. For \( U \) and \( V \), each row represents the latent factors for each feature of user and item, respectively. The predicted rating of user \( u \) on item \( i \) is given as

\[
\hat{r}_{u,i} = p_{u}^T q_{i}.
\]
The latent factor representation is learned by minimizing the following cost function:

\[
J = \sum_i \sum_j c_{i,j} \left( r_{i,j} - p_i q_j^T \right)^2 + \lambda_1 \left( \sum_i \sum_{d_u} (x_{i,d_u} - p_i u_{d_u}^T)^2 \right) + \sum_{d_u} \left( \sum_j (y_{j,d_u} - q_j v_{d_u}^T)^2 \right) + \lambda_2 \left( \sum_i \|p_i\|^2 + \sum_j \|q_j\|^2 \right) + \sum_{d_u} \|u_{d_u}\|^2 + \sum_{d_u} \|v_{d_u}\|^2 \]

(3)

where \(\lambda_1\) adjusts how much information the model should learn from side data, and \(\lambda_2\) is a regularization term to prevent overfitting. \(r_{i,j} = 0\) if the rating is unobserved, and \(r_{i,j} > 0\) otherwise. \(c_{i,j}\) is a weight on the error term for each rating record. This article considers two definitions of \(c_{i,j}\) as follows.

1) **ObsOnly**: \(c_{i,j} = 1\) if \(r_{i,j} > 0\), and \(c_{i,j} = 0\) if \(r_{i,j} = 0\). The loss function only minimizes the square error on the observed ratings.

2) **InclUnc**: \(c_{i,j} = 1\) if \(r_{i,j} > 0\), and \(c_{i,j} = \alpha\) if \(r_{i,j} = 0\), where \(0 < \alpha < 1\) is an uncertainty coefficient on the unobserved ratings. This case assigns a lower weight to the loss for unobserved ratings.

The matrix factorization for explicit feedback typically employs the first definition to reduce the bias of unobserved interaction and improve efficiency. However, employing the second definition reveals less information to the FL server. As is shown in Sections IV and VI, adopting the second definition would present a challenge for the server attack. Therefore, we will consider both cases when designing the server attack.

### C. Federated Implementation

The federated setting consists of three parties: 1) clients, 2) FL server, and 3) item server. Each client holds their ratings and attributes locally and performs local update of \(P\). FL server receives the gradients from clients and item server, and updates \(U\) and \(Q\). Item server is introduced to facilitate the training process. It stores the item features and conducts update of \(V\). The following explains the details of the updates of each latent factor matrix.

**User feature latent factor** \(U\) is updated on the FL server with the formula

\[
u_{d_u}^t = u_{d_u}^{t-1} - \gamma \frac{\partial J}{\partial u_{d_u}} \]

(4)

where

\[
\frac{\partial J}{\partial u_{d_u}} = -2 \sum_i f(i, d_u) + 2\lambda_2 u_{d_u} \]

(5)

where \(f(i, d_u) = (x_{i,d_u} - p_i u_{d_u}^T) p_i\) is computed on each user locally.

**Item latent factor** \(Q\) is updated on the FL server with the formula

\[
q_j^t = q_j^{t-1} - \gamma \frac{\partial J}{\partial q_j} \]

(6)

where

\[
\frac{\partial J}{\partial q_j} = -2 \sum_i f(i, j) - 2\lambda_1 \sum_{d_y} f(j, d_y) + 2\lambda_2 q_j \]

(7)

where \(f(j, d_y) = (y_{j,d_y} - v_j v_{d_y}^T) v_{d_y}\) is computed on the item server, and \(f(i, j) = c_{i,j} (r_{i,j} - p_i q_j^T) p_i\) is computed on each user locally. Noted that for **ObsOnly**, the user only computes and sends the gradients of items with \(c_{i,j} > 0\), i.e., the rated items. For **InclUnc**, the gradients for all items will be sent to the server.

Both user latent factor \(P\) and item feature latent factor \(V\) adopt two updating methods as follows.

1) **SemiAlternating Least Squares (SemiALS)**: Optimal \(P\) and \(V\) are computed using closed form formula under fixed \(U\) and \(Q\). Other parameters are updated using gradient descent method.

2) **Stochastic Gradient Descent (SGD)**: All of the parameters are updated using gradient descent method.

The time complexity for **SemiALS** is \(O(mK^2 + K^3)\) per iteration, higher than that of **SGD**. However, **SGD** requires more iterations to achieve the optimum [29].

User latent factor \(P\) is updated on each client locally. For **SemiALS**, it is updated with the formula

\[
p_i^t = \left( r_i C(i)^T + \lambda_1 x_i U \right) \left( Q^T C(i)^T Q + \lambda_1 U^T U + \lambda_2 I \right)^{-1} \]

(8)

where \(C(i)^T\) is an \(m \times m\) diagonal matrix with \(C(i)_{j,j} = c_{i,j}\).

For **SGD**, it is updated with the formula

\[
p_i^t = p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i} \]

(9)

where

\[
\frac{\partial J}{\partial p_i} = -2 \sum_i c_{i,j} (r_{i,j} - p_i q_j^T) q_j - 2\lambda_1 \sum_{d_u} (x_{i,d_u} - p_i u_{d_u}^T) u_{d_u} + 2\lambda_2 p_i. \]

(10)

**Item feature latent factor** \(V\) is updated on the item server. For **SemiALS**, it is updated with the formula

\[
v_{d_y}^t = \left( y_{d_y} Q + \frac{\lambda_2}{\lambda_1} I \right)^{-1}. \]

(11)

For **SGD**, it is updated with the formula

\[
v_{d_y}^t = v_{d_y}^{t-1} - \gamma \frac{\partial J}{\partial v_{d_y}} \]

(12)

where

\[
\frac{\partial J}{\partial v_{d_y}} = -2 \sum_i (y_{i,d_y} - q_j v_{d_y}^T) q_j + 2\lambda_2 v_{d_y}. \]

(13)
Algorithm 1: FedMVMF.

**FL Server:**

1. Initialize $U$ and $Q$.
2. for $t = 1$ to $T$ do
   1. Receive and aggregate $f(i, j)$ and $f(i, d_u)$ from user $i$ for $i \in [1, n]$.
   2. Receive $f(j, d_v)$ from item server.
   3. Update $U$ using (4).
   4. Update $Q$ using (6).
3. end for

**Item Server:**

1. while True do
   1. Receive $Q$ from FL server.
   2. Compute local $V$ using (11).
   3. Compute item latent factor gradients $f(j, d_v)$.
   4. Transmit gradients to server.
2. end while

**Client:**

1. while True do
   1. Receive $U$ and $Q$ from server.
   2. Compute local $p_i$ using (8).
   3. Compute $U$ gradients $f(i, d_u)$ for $d_u \in [1, l_x]$.
   4. Compute $Q$ gradients $f(i, j)$ for $j \in [1, m]$.
   5. Transmit gradients to server.
2. end while

Algorithm 1 outlines the federated implementation of MVMF (FedMVMF). The gradient descents of $U$ and $Q$ are performed using the adaptive moment estimation (Adam) method to stabilize the convergence.

**D. Cold-Start Recommendation**

**Cold-Start User Recommendation:** For any new user $i$, the system first generates the user latent factor $p_i$ based on the user’s attribute $x_i$ and the user feature latent factor matrix $U$. Then the predicted rating of user $i$ on item $j$ is given by the inner product of $p_i$ and $q_j$. $p_i$ is calculated by minimizing the loss function

$$J = \lambda_1 \sum_{d_u} (x_i, d_u - p_i u_{d_u}^T)^2 + \lambda_2 \sum_{d_u} \|u_{d_u}\|^2. \quad (14)$$

The optimal solution of $p_i$ is defined as

$$p_i^* = x_i U \left( U^T U + \frac{\lambda_2}{\lambda_1} I \right)^{-1}. \quad (15)$$

**Cold-Start Item Recommendation:** Given a new item $j$, the system first generates the item latent factor $q_j$ based on the item’s feature $y_j$ and the item feature latent factor matrix $V$. The estimated $q_j$ is then used to compute the predicted rating. $q_j$ is calculated by minimizing the loss function

$$J = \lambda_1 \sum_{d_u} (y_j, d_u - q_j v_{d_u}^T)^2 + \lambda_2 \sum_{d_u} \|v_{d_u}\|^2. \quad (16)$$

The optimal solution of $q_j$ is defined as

$$q_j^* = y_j V \left( V^T V + \frac{\lambda_2}{\lambda_1} I \right)^{-1}. \quad (17)$$

**IV. SERVER RECONSTRUCTION ATTACK ANALYSIS**

In FedMVMF, the FL server could reconstruct the user information and attributes based on the gradients they received. In this section, we consider the attacks for both SemiALS and SGD updates on user latent factor. Within each case, the attacks are slightly different between ObsOnly and InclUnc. The analysis is based on the assumption of honest clients and an honest-but-curious server [19].

**A. Reconstruction Attack for SemiALS Update**

For SemiALS the FL server can recover the user information within only one epoch given that the server has access to $U$ and $Q$.

**Attack for ObsOnly:** In this case, the clients only upload the gradients for items with observed ratings. Therefore, for any user $i$, the gradients which the FL server receives are given by

$$f(i, j) = (r_{i,j} - p_i q_{j}^T) p_i, \quad j \in O_i$$

$$f(i, d_u) = (x_i, d_u - p_i u_{d_u}^T) p_i, \quad d_u \in [1, l_x] \quad (18)$$

where $f(i, j)$ and $f(i, d_u)$ denote the vector of gradient with length $K, O_i$ denotes the collection of items rated by user $i$, and $l_x$ denote the number of user attributes.

In SemiALS, $p_i$ is updated by (8). Given that $c_{i,j} = 0$ when $r_{i,j} = 0$, the formula could be reduced to

$$p_i = (r_i Q_i + \lambda_1 A_i) (Q_i^T Q_i + \lambda_1 U^T U + \lambda_2 I)^{-1} \quad (19)$$

where $r_i$ is the vector of observed ratings, and $Q_i = Q_{|O| \times K}$ is the latent factors for items rated by user $i$.

Let $A_i = Q_i (Q_i^T Q_i + \lambda_1 U^T U + \lambda_2 I)^{-1}$, and $B_i = \lambda_1 U (Q_i^T Q_i + \lambda_1 U^T U + \lambda_2 I)^{-1}$, both of which could be computed on the FL server. Then $p_i$ could be written as $p_i = r_i A_i + x_i B_i$. Plug into (18)

$$f_i^Q = r_i^T r_i A_i + r_i^T x_i B_i - Q_i (A_i^T R_r A_i + A_i^T R_r B_i + B_i^T R_x A_i + B_i^T R_x B_i)$$

$$f_i^U = x_i^T r_i A_i + x_i^T x_i B_i - U (A_i^T R_r A_i + A_i^T R_r B_i + B_i^T R_x A_i + B_i^T R_x B_i) \quad (20)$$

where $f_i^Q = f_i^Q |_{O_i \times K}$ with $i$th row being $f(i, j)$, $f_i^U = f_i^U |_{l_x \times K}$ with $d_u$th row being $f(i, d_u)$, and

$$R_r = r_i^T r_i, R_x = r_i^T x_i$$

$$X_r = x_i^T r_i, X_x = x_i^T x_i. \quad (21)$$

Then the FL server obtains a second order nonlinear system with $(l_x + |O_i|) \times K$ equations, consisting of $(l_x + |O_i|)$ variables, $r_i$ and $x_i$. Therefore, it is plausible to find the solution of user ratings $r_i$ and user attributes $x_i$ using methods, such as the Newton–Raphson algorithm. To reconcile the number of equations and variables, we choose a random factor $n \in [1, K]$, and solve the equation systems under the fixed $n$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
**Attack for InclUnc:** In this case, the client sends gradients of all items to the FL server, multiplied by an uncertainty coefficient \( c_{i,j} \). For any user \( i \), the gradients of the FL server are given by

\[
f(i, j) = c_{i,j} \left( r_{i,j} - p_i q_j^T \right) p_i, j \in [1, m]
\]

\[
f(i, d_u) = \left( x_{i,d_u} - p_i u_{d_u}^T \right) p_i, d_u \in [1, l_x].
\]

(22)

Let \( A'_i = C^{(i)} Q (Q^T C^{(i)} Q + \lambda_1 U^T U + \lambda_2 I)^{-1} \), and \( B'_i = \lambda_1 (Q^T C^{(i)} Q + \lambda_1 U^T U + \lambda_2 I)^{-1} \). Then \( p_i \) can be written as

\[
p_i = r_i A'_i + x_i B'_i.
\]

(23)

Plugging into (22), we can obtain the final equation system

\[
\begin{align*}
f^Q_i = C^{(i)} \left( r_i A'_i + x_i B'_i - Q ((A'_i)^T R_i A'_i + (A'_i)^T R_x B'_i + (B'_i)^T X_i A'_i + (B'_i)^T X_i B'_i) \right) \\
f^U_i = x_i^T r_i A'_i + x_i^T x_i B'_i - U ((A'_i)^T R_i A'_i + (A'_i)^T R_x B'_i + (B'_i)^T X_i D_i + (B'_i)^T X_i B'_i)
\end{align*}
\]

where \( r_i \) is user \( i \)'s ratings for all items, and \( A'_i = C^{(i)} Q (Q^T C^{(i)} Q + \lambda_1 U^T U + \lambda_2 I)^{-1} \) and \( B'_i = \lambda_1 (Q^T C^{(i)} Q + \lambda_1 U^T U + \lambda_2 I)^{-1} \) are dependent on \( r_i \).

Since \( C^{(i)} \) is a function of \( r_i \), the system consists of \( l_x + m \) variables and \( (l_x + m) \times K \) equations. Therefore, it is possible to recover the user information by solving the equation system. Similarly, a random factor \( n \in [1, K] \) is fixed to align the number of equations and variables.

**B. Reconstruction Attack for SGD Update**

For SGD, the FL server can recover the user information within only two episodes given that the server has access to \( U \) and \( Q \).

**Attack for ObsOnly:** After two epochs, the gradients of the FL server receive from users \( i \) is given by

\[
\begin{align*}
f^t(i, j) &= \left( r_{i,j} - \frac{p_i q_j^T}{p_i q_j^T} \right) p_i, j \in O_i \\
f^{t-1}(i, j) &= \left( r_{i,j} - \frac{p_i^{-1}(q_j^{-1})^T}{p_i^{-1}(q_j^{-1})^T} \right) p_i^{-1}, j \in O_i \\
f^t(i, d_u) &= \left( x_{i,d_u} - \frac{p_i u_{d_u}^T}{p_i u_{d_u}^T} \right) p_i, d_u \in [1, l_x] \\
f^{t-1}(i, d_u) &= \left( x_{i,d_u} - \frac{p_i^{-1}(u_{d_u}^{-1})}{p_i^{-1}(u_{d_u}^{-1})} \right) p_i^{-1}, d_u \in [1, l_x].
\end{align*}
\]

(25)

In pure SGD, the user latent factor is updated using (9) and (10). Plugging into the first gradient of (25), we have

\[
c^{f^t}_n(i, j) = \left( r_{i,j} - \frac{p_i^{-1}(q_j^{-1})^T}{p_i^{-1}(q_j^{-1})^T} \right) \left( q_j^{-1} + \Delta q_j^{-1} \right)^T
\]

\[
\times \left( \frac{p_i^{-1} - \frac{\partial J}{\partial p_i^{t-1}}}{p_i^{-1}} \right)
\]

(26)

where \( \Delta q_j^{-1} = q_j^{-1} - q_j^{-1} \), \( f^t_n(i, j) \) denote the \( n \)th element of \( f^t(i, j) \), and \( p_i^{t-1} \) denote the \( n \)th element of \( p_i^{t-1} \).

Equation (26) is a multiplication of two terms. By looking at the first term, we have

\[
\begin{align*}
r_{i,j} &- \left( \frac{p_i^{-1} - \gamma \frac{\partial J}{\partial p_i^{t-1}}}{p_i^{t-1}} \right) \left( q_j^{-1} + \Delta q_j^{-1} \right)^T = r_{i,j} \\
- p_i^{-1}(q_j^{-1})^T &- p_i^{-1}(\Delta q_j^{-1})^T + \gamma \frac{\partial J}{\partial p_i^{t-1}} (q_j^{-1})^T
\end{align*}
\]

\[
= \frac{G_n(j)}{p_i^{t-1}} + p_i^{-1} g
\]

(27)

where

\[
G_n(j) = f^{t-1}(i, j) - 2\gamma \left( \sum_{k \in O_i} f^{t-1}(i, k) q_k^{-1} \right)
\]

\[
+ \lambda_1 \sum_{d_u} f^{t-1}(i, d_u) u_{d_u}^{-1} \left( q_j^{-1} \right)^T
\]

\[
g = 2\gamma \lambda_2 \left( q_j^{-1} \right)^T - (\Delta q_j^{-1})^T.
\]

(28)

Then we look at the second term of (26), which is given by

\[
\begin{align*}
p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i^{t-1}} &= p_i^{t-1}(1 - 2\gamma \lambda_2) + \frac{2\gamma}{p_i^{t-1}} \\
\times \left[ \sum_{k \in O_i} f^{t-1}(i, k) q_k^{-1} + \lambda_1 \sum_{d_u} f^{t-1}(i, d_u) u_{d_u}^{-1} \right]
\end{align*}
\]

\[
= p_i^{t-1}(1 - 2\gamma \lambda_2) + \frac{F_n}{p_i^{t-1}}
\]

(29)

where

\[
F_n = 2\gamma \left[ \sum_{k \in O_i} f^{t-1}(i, k) q_k^{-1} \right.
\]

\[
+ \lambda_1 \sum_{d_u} f^{t-1}(i, d_u) u_{d_u}^{-1} \bigg].
\]

(30)

Then (26) can be written as

\[
\begin{align*}
f^t_n(i, j) = \left( \frac{G_n(j)}{p_i^{t-1}} + p_i^{-1} g \right) \left( p_i^{t-1}(1 - 2\gamma \lambda_2) + \frac{F_n}{p_i^{t-1}} \right)
\end{align*}
\]

(31)

For \( n \in [1, K], j \in O_i \), where \( p_i^{t-1} \) is the variable to solve. Noted that \( G_n(j) \), \( g \), and \( F_n \) could be computed on the FL server.

Since there are \( K \) variables and \( K \times |O_i| \) equations, there should exist a solution \( p_i^{t-1} \) to satisfy the system (31). To reconcile the number of equations and variables, we choose a random item \( j \in O_i \), and solve the equation systems under the fixed \( j \).

After obtaining \( p_i^{t-1} \), the server could compute \( r_{i,j} \) and \( x_{i,d_u} \) as follows:

\[
r_{i,j} = \frac{f_n^{t-1}(i, j)}{p_i^{t-1}} + p_i^{-1}(q_j^{-1})^T
\]

\[
x_{i,d_u} = \frac{f_n^{t-1}(i, d_u)}{p_i^{t-1}} + p_i^{-1}(u_{d_u}^{-1})^T.
\]

(32)
Attack for InclUnc: Similarly, the FL server first obtains the equation system for $p_i^{-1}$ given by

$$f_n(i,j) = \frac{f_n^{-1}(i,j)}{p_{in}^{-1}} + c_{i,j}G_n(j) + c_{i,j}p_i^{-1}g$$

$$= \left(\frac{F_n}{p_{in}^{-1}} + p_{in}^{-1}(1 - 2\gamma_l^2)\right), n \in [1, K], j \in [1, m]$$

where

$$G_n(j) = -2\gamma \left(\sum_k f_n^{-1}(i,k)q_{kn}^{-1}\right) + \lambda \sum_{d_u} f_n^{-1}(i,d_u)u_{d_u}^{-1}$$

$$F_n = 2\gamma \left(\sum_k f_n^{-1}(i,k)q_{kn}^{-1} + \lambda \sum_{d_u} f_n^{-1}(i,d_u)u_{d_u}^{-1}\right)$$

$$g' = 2\lambda^2\gamma p_i^{-1} - \Delta p_i^{-1}.$$  

(33)

For detail derivation of (33) see Appendix A. Noted that $c_{i,j}$ is a function of $r_{i,j}$, which is dependent on $p_i^{-1}$ based on (35). Therefore, $c_{i,j}$ is linked with $p_i^{-1}$.

Given $K$ variables and $K \times m$ equations, the server should be able to find a solution $p_i^{-1}$ for the system. Similarly, a random item $j \in [1, m]$ is fixed when solving the equation system.

Then the rating and user attributes could be computed as

$$r_{i,j} = \frac{f_n^{-1}(i,j)q_{kn}^{-1}p_i^{-1} - f_n^{-1}(i,j)p_i^{-1}(q_{kn}^{-1})'}{f_n^{-1}(i,j)p_i^{-1} - f_n^{-1}(i,j)p_i^{-1}(q_{kn}^{-1})'}$$

$$x_{i,d_u} = \frac{f_n^{-1}(i,d_u)p_i^{-1}}{p_{in}^{-1}} + p_{in}^{-1}(u_{d_u}^{-1})'$$

(35)

where $p_{in}^{-1}$ and $p_i^{-1}$ can be obtained from formula (9) and (10).

V. PRIVACY-PRESERVING MVMF (PriMVMF)

To prevent information leakage, we develop PrivMVMF, a privacy-preserving federated MVMF framework enhanced with homomorphic encryption (HE). In this framework, the client encrypts the gradients before sending them to the server, and the server can perform computation on the encoded gradients. The above attacks are based on access to individual gradients, while in HE, these gradients are sent to the server in encrypted form, rendering the reconstruction attacks infeasible.

A. Threshold Paillier Cryptosystem

This study utilizes a threshold variant of partially HE scheme: the Paillier cryptosystem [20], [21], where the secret key is distributed among a set of clients, such that a minimum number of parties is required to do the decryption. The protocol consists of four parts: 1) key generation, 2) encryption, 3) share decryption, and 4) share combining.

1) Key Generation: Based on the $\text{keysize}$, $(sk, pk) = \text{Gen}(\text{keysize})$ returns the public key $pk$ shared among all participants, and secret key $sk = (sk_1, sk_2, \ldots, sk_i)$ distributed only among $t$ decrypters. Before the training process, the decrypters jointly generate the key pair following Veugen et al.’s method [22].

2) Encryption: $c = \text{Enc}(m, pk)$ encrypts message $m$ to ciphertext $c$ using public key $pk$.

3) Share Decryption: $c_i = \text{PartialDec}(c, sk_i)$ partially decrypts the ciphertext $c$ by decrypter $i$ using local secret key $sk_i$.

4) Share Combining: $m = \text{Combine}(c_1, c_2, \ldots, c_t)$ combines the partially decrypted shares to reconstruct the message.

Given two plaintexts $m_1$ and $m_2$, the Paillier cryptosystem $E$ has the following properties.

1) Addition: $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$.

2) Multiplication: $E(m_1)^{m_2} = E(m_1 \cdot m_2)$.

Number Encoding Scheme: The Paillier encryption is only defined for nonnegative integers, but the recommendation system contains float and negative numbers. The study follows Chai et al.’s [16] method to convert floating points and negative numbers into unsigned integers.

Sampling of Unrated Item: For the treatment of unrated items, this framework strikes a balance between efficiency and privacy protection. The ObsOnly method is efficient but it reveals what items have been rated by the user. The InclUnc method leaks no information but is computation intensive. To reconcile the two objectives, we design a strategy to randomly sample a portion of unrated items. Then the $c_{i,j}$ is given as follows:

$$c_{i,j} = \begin{cases} 1, & r_{i,j} > 0 \\ \alpha, & r_{i,j} = 0 \text{ and } \text{samp}_{i,j} = 1 \\ 0, & r_{i,j} = 0 \text{ and } \text{samp}_{i,j} = 0 \end{cases}$$

(36)

where $0 < \alpha < 1$, $\text{samp}_{i,j} = 1$ if item $j$ appears in the sampled unrated items for user $i$, and $\text{samp}_{i,j} = 0$ otherwise. Users only send the gradients with $c_{i,j} > 0$.

For each user, we determine the number of sampled unrated items as a multiple of his rated items, denoted by $\rho$. Then the upper-bound probability that the FL server could correctly infer whether a given item is rated by the user is given by $\frac{1}{\rho + 1}$.

Decrypting Clients: It is time-consuming to perform the update using the encrypted gradients. To reduce complexity, the server sends the aggregated gradient to a set of decrypting users for decryption. Before training, the decrypting clients perform distributed key generation to jointly compute the public key and securely generate the private key. On receiving the encrypted gradients, each decrypting user computes their decryption share and sends it to the server. The shares are merged by the server to recover the plaintext aggregated gradients for parameter update.

Algorithms: The detailed steps of PrivMVMF are shown in Algorithm 2. Noted that for the update of user latent factor $P$ and item feature latent factor $V$, we adopt the SemiALS strategy for the following reason: Although SemiALS has higher time complexity per iteration, it requires fewer iterations to achieve the optimum, and thus, fewer encryption and decryption operations, the bottleneck of the HE scheme.
B. Privacy Analysis

The privacy of the algorithm is analyzed in terms of information leakage, which is characterized into two forms: 1) original information, the observed user data, and 2) latent information, properties of user data [30]. This section analyzes the privacy of original information under two threat models.

Security Against Honest-but-Curious Server: An honest-but-curious server will not deviate from the defined protocol but attempt to learn information from legitimately received messages. During the training of PrivMVMF, the individual gradients are sent to the server in the encrypted form, and only the plaintext aggregated gradients are available to the server. The following shows that given the aggregated gradients, it leaks trivial original information about user data to the server.

Let \( f(j) \) and \( f(d_u) \) be the aggregated gradients for item \( j \) and user feature \( d_u \), given by

\[
\begin{align*}
    f(j) &= \sum_{i} f(i, j) = \sum_{i:j \in O^i} c_{i,j}(r_{ij} - p_i^T d_j^i) p_i, j \in [1, m] \\
    f(d_u) &= \sum_{i} f(i, d_u) = \sum_{i}(x_{i,d_u} - p_i u_{d_u}^T) p_i, d_u \in [1, l_x] 
\end{align*}
\]

where \( O^i \) denotes the set of items rated by or appeared in the sampled unrated items for user \( i \).

In PrivMVMF, \( p_i \) is updated by

\[
    p_i = (r_i C^{(i)} Q^i_1 + \lambda_1 x_i U)((Q^i_1)^T C^{(i)} Q^i_1 + \lambda_1 U^T U + \lambda_2 I)^{-1} 
\]

where \( Q^i_1 = Q_{i | O^i \times K} \) is the latent factors for items in \( O^i \).

Let \( A'_i = C^{(i)} Q^i_1((Q^i_1)^T C^{(i)} Q^i_1 + \lambda_1 U^T U + \lambda_2 I)^{-1} \), and \( B'_i = \lambda_1 U((Q^i_1)^T C^{(i)} Q^i_1 + \lambda_1 U^T U + \lambda_2 I)^{-1} \). Then \( p_i \) can be written as

\[
    p_i = r_i A'_i + x_i B'_i. 
\]

Plugging into (38), we can obtain the equation system as follows:

\[
\begin{align*}
    f(j) &= \sum_{i} c_{i,j}(r_{ij} r_i A'_i + r_{ij} x_i B'_i - q_j ((A'_i)^T R_r A'_i \\
    &+ (A'_i)^T R_x B'_i + (B'_i)^T X_r A'_i + (B'_i)^T X_x B'_i)) j \in [1, m] \\
    f(d_u) &= \sum_{i}(x_{i,d_u} r_i A'_i + x_{i,d_u} x_i B'_i - u_{d_u}((A'_i)^T R_r A'_i \\
    &+ (A'_i)^T R_x B'_i + (B'_i)^T X_r A'_i + (B'_i)^T X_x B'_i)) d_u \in [1, l_x] 
\end{align*}
\]

where

\[
    R_r = r_i^T r_i, R_x = r_i^T x_i, X_r = x_i^T r_i, X_x = x_i^T x_i. 
\]

Algorithm 2: PrivMVMF.

Randomly select \( t \) honest clients as decryptors.
Decrypytors collaboratively generate public and private keys, and distribute the public key.

**FL Server:**

Initialize \( U \) and \( Q \).

**for** \( t = 1 \) **to** \( T \)

- Receive and aggregate encrypted \( f(i, j) \) and \( f(i, d_u) \) from user \( i \) for \( i \in [1, n] \).
- Send encrypted \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) to decryptors.
- Receive partially decrypted shares of \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) from decryptors, and reconstruct the plaintext gradients.
- Receive \( f(j, d_u) \) from item server.
- Update \( U \) using (4).
- Update \( Q \) using (6).

end for

**Item Server:**

**while** True do

- Receive \( Q \) from FL server.
- Compute local \( V \) using (11).
- Compute item latent factor gradients \( f(j, d_u) \).
- Transmit gradients to server.

end while

**Client:**

**while** True do

- Receive \( U \) and \( Q \) from server.
- Compute local \( p_i \) using (8).
- Compute encrypted \( U \) gradients \( f(i, d_u) \) for \( d_u \in [1, l_x] \).
- Compute encrypted \( Q \) gradients \( f(i, j) \) for \( j \in [1, m] \).
- Transmit gradients to server.

end while

**Decrypter:**

**while** True do

- Receive encoded \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) from FL server.
- Partially decrypt and transmit \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) to FL server.

end while

| Algorithm 2: PrivMVMF. |
|------------------------|
| Randomly select \( t \) honest clients as decryptors |
| Decrypytors collaboratively generate public and private keys, and distribute the public key |
| **FL Server:** |
| Initialize \( U \) and \( Q \). |
| **for** \( t = 1 \) **to** \( T \) do |
| Receive and aggregate encrypted \( f(i, j) \) and \( f(i, d_u) \) from user \( i \) for \( i \in [1, n] \). |
| Send encrypted \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) to decryptors. |
| Receive partially decrypted shares of \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) from decryptors, and reconstruct the plaintext gradients. |
| Receive \( f(j, d_u) \) from item server. |
| Update \( U \) using (4). |
| Update \( Q \) using (6). |
| end for |
| **Item Server:** |
| **while** True do |
| Receive \( Q \) from FL server. |
| Compute local \( V \) using (11). |
| Compute item latent factor gradients \( f(j, d_u) \). |
| Transmit gradients to server. |
| end while |
| **Client:** |
| **while** True do |
| Receive \( U \) and \( Q \) from server. |
| Compute local \( p_i \) using (8). |
| Compute encrypted \( U \) gradients \( f(i, d_u) \) for \( d_u \in [1, l_x] \). |
| Compute encrypted \( Q \) gradients \( f(i, j) \) for \( j \in [1, m] \). |
| Transmit gradients to server. |
| end while |
| **Decrypter:** |
| **while** True do |
| Receive encoded \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) from FL server. |
| Partially decrypt and transmit \( \sum_i f(i, j) \) and \( \sum_i f(i, d_u) \) to FL server. |
| end while |

A. Dataset and Experimental Setup

The experiment is performed on MovieLens-1M dataset.

The dataset contains 914,676 ratings from 6040 users on 3952 items.

VI. EXPERIMENTS

[1] Online. Available: https://grouplens.org/datasets/movielens/1m/
TABLE II
HYPERPARAMETER FOR MVMF ON MOVIELENS DATASET

| Hyperparameter          | Value | \( \beta_1 \) | \( \beta_2 \) | \( \varepsilon \) | \( \gamma \) | \( \alpha \) |
|------------------------|-------|---------------|---------------|---------------|---------------|---------------|
| Occupation, Gender     | 6     | 0.5           | 0.99          | 1e-8          | 0.05          | 0.1           |
| Value                  | 1     | 10            | 1             | 10            | 20            | 1024          |

([f|e] denotes the number of iterations and epochs. \( p \) is the proportion of sampled items based on rated items. \( l_{0} \) denotes the length of public key.)

TABLE III
ACCURACY OF SERVER RECONSTRUCTION ATTACK

| SemiALS                      | SGD                      |
|------------------------------|--------------------------|
|                             | OvByOnly, InclUnc | OvByOnly, InclUnc |
| Rating                       | 0.9911                   | 0.8785           | 0.8223 | 0.8182 |
| Attribute                    | 0.9991                   | 0.8898           | 0.9230 | 0.8474 |

TABLE IV
TEST ACCURACY OF FEDMVMF AND PRIVMVMF

| FedMVMF | PrivMVMF | Diff% |
|---------|----------|-------|
| Existing User and Item       |           |       |
| Precision                     | 0.2677±0.0034 | 0.2931±0.0047 | 8.69 |
| Recall                        | 0.2092±0.0034 | 0.2208±0.0034 | 5.24 |
| FI                             | 0.2348±0.0043 | 0.2519±0.0040 | 6.76 |
| RMSE                           | 0.9624±0.0034 | 0.9648±0.0036 | 0.24 |
| Cold-start User               |           |       |
| Precision                     | 0.2668±0.0076 | 0.2728±0.0081 | 2.18 |
| Recall                        | 0.0404±0.0010 | 0.0423±0.0010 | 4.66 |
| FI                             | 0.0701±0.0017 | 0.0733±0.0017 | 4.33 |
| RMSE                           | 1.0306±0.0038 | 1.0509±0.0038 | 1.93 |
| Cold-start Item               |           |       |
| Precision                     | 0.1781±0.0077 | 0.1735±0.0082 | 2.60 |
| Recall                        | 0.2497±0.0014 | 0.2563±0.0028 | 2.56 |
| FI                             | 0.2075±0.0021 | 0.2064±0.0027 | 0.55 |
| RMSE                           | 1.0233±0.0099 | 1.0213±0.0083 | 0.19 |

(The values denote the mean ± standard deviation of the performance.)

TABLE V
TIME CONSUMPTION IN EACH PHASE (SECONDS)

| FedMVMF | SecMVMF | PrivMVMF |
|---------|---------|----------|
| Local Update | 0.62±0.0005 | 3.77±0.02 | 3.42±0.03 |
| Aggregation | 3.29±0.26 | 262.14±1.88 | 247.01±1.58 |
| Decryption | /        | 102.33±1.31 | 103.12±1.20 |
| Server Update | 0.19±0.07 | 469.65±2.28 | 0.39±0.03 |

(method within each scenario using a sample of 100 users. See Table VI for the selected method within each scenario.)

We construct the rating matrix based on explicit ratings, with each user submitting at least 20 ratings. The experiment is implemented on Ubuntu Linux 20.04 server with 32-core CPU and 128 GB RAM, where the programming language is Python.

We use Bayesian optimization [28] approach based on four-fold cross-validation to optimize the hyperparameters. Table II summarizes the hyperparameters for the experiment.

We assume that at most four decrypters could collaborate with the server, and use five decrypting clients to implement the PrivMVMF algorithm.

B. Server Attack

Solving Nonlinear System: To perform a server reconstruction attack, we first developed the equation systems described in Section IV. To solve the nonlinear systems, we experiment with the four methods [23], [24], [25], [26]: 1) modified Powell’s hybrid method, 2) Broyden’s bad method, 3) Scalar Jacobian approximation, and 4) Anderson mixing, and select the best method within each scenario using a sample of 100 users. See Table VI for the selected method within each scenario.

Smoothing \( c_{i,j} \) as a Function of \( r_{i,j} \) for InclUnc: In InclUnc, \( c_{i,j} = f(r_{i,j}) \) is not a continuous function, while the Jacobian matrix is needed for most of the iterative methods. To smooth \( f \), we design the following function:

\[
   c_{i,j} = \begin{cases} 
   1, & r_{i,j} > 1 \\
   r_{i,j}, & 1 \geq r_{i,j} > 0 \\
   0, & 0 \geq r_{i,j} 
   \end{cases} 
\]  

(Evaluation Metrics: We employ accuracy to measure the performance of server inference, with the following steps. After obtaining the estimation of \( \hat{r}(i) \) and \( \hat{x}_i \) for each user \( i \), we clipped \( \hat{r}(i) \) within \([0, R_{max}]\) and \( \hat{x}_i \) within \([0, 1]\) (\( x_i \) are all dummy variables), and then rounded the estimations to the nearest integers. The accuracy for user ratings and attributes is computed as follows:

\[
   \text{Accuracy for Rating} = \frac{1}{|\text{usr}|} \sum_{i=1}^{|\text{usr}|} \frac{|\hat{r}_i - r_i|}{|r_i|} 
\]

\[
   \text{Accuracy for Attribute} = \frac{1}{|\text{usr}|} \sum_{i=1}^{|\text{usr}|} \frac{|\hat{x}_i - x_i|}{|x_i|} 
\]

where \( |\text{usr}| \) denotes the set of all users, and \( \hat{r}_i \) and \( \hat{x}_i \) denote the transformed estimation of \( r_i \) and \( x_i \).

Result and Analysis: Table III reports the accuracy of server reconstruction attack in four scenarios, from which we can make the following observations: 1) In all cases, the server can recover the user’s private information with accuracy > 80%, which is a nonnegligible privacy concern. 2) For both SemiALS and SGD, including an uncertainty coefficient deteriorates the performance of server attack. One explanation is that \( c_{i,j} \) is not a differentiable function of \( r_{i,j} \), posing a challenge to obtaining the Jacobian matrix of the system. 3) Using SGD method to update \( P \) makes it harder for the server to infer user information given by the reduced accuracy.

Robustness Check: We consider the case when a small amount of noise is added to the gradients. With perturbed gradients, the equation systems are solved using the following steps:

1) For SemiALS (SGD with ObsOnly/SGD with InclUnc), compute the set of solutions for each \( n \in [1, K] \) (\( j \in O_i \) \( i,j \) from a randomly chosen set of items).
2) Clip the solutions to the correct domain.
3) Take the average of the solutions and round the estimations to integers.

We conduct the experiment under the four scenarios, where the Laplace noises are added to the gradients with scale \( b \) ranging
SemiALS and SecMVMF to the federated MVMF. Clients (partially) decrypts the gradients.

III

Table when the noise scale exceeds $b$. Cold-start Item

[27]

FedMVMF for consistency, slightly different from the scenario. See Fig. III

and

presents the computation time in each epoch, from Scenarios of Testing:

and Item

Existing User and Item

C.

Evaluation Metrics: The study adopted the following four evaluation metrics: 1) Root mean square error (RMSE), Precision, Recall, and F1. The accurate prediction is defined as an item recommended rated above a threshold by the given user [27]. See Appendix B for the details of evaluation metrics.

Scenarios of Testing: Both approaches can provide recommendations for new users and items. Table IV presents the performance of the three scenarios: 1) Existing User and Item, 2) Cold-start Item, and 3) Cold-start User. For Existing User and Item, the items of each user are randomly divided into 80% training set and 20% testing set. The items and users in the testing set are supposed to have a rating history. For Cold-start Item, the items are randomly divided into 90% training set and 10% testing set. The testing items are treated as new items without a rating history. For Cold-start User, a random subset of 10% users is held out as new users for testing.

Accuracy: Table IV compares the testing accuracy between FedMVMF and PrivMVMF, with each approach running for five rounds. Noted that the FedMVMF adopts the same sampling strategy as PrivMVMF for consistency, slightly different from that in Section III. It can be observed that the difference in testing accuracy is trivial in all scenarios, suggesting that the proposed framework is lossless.

Efficiency: We compare our PrivMVMF with SecMVMF, the adaption of Chai et al.’s secure federated matrix factorization method [16] to the federated MVMF. SecMVMF adopted HE to enhance the security without decrypting clients, and thus, the item latent factor and user feature latent factor were updated and stored as ciphertext on the server. The clients decrypt the parameters and perform the local update. We execute SecMVMF under PartText setting, where users only upload the gradients of the rated items. Since both methods are demonstrated to be lossless, we focus on the comparison of efficiency.

The model training in each epoch can be divided into four phases: 1) local update, 2) aggregation, 3) decryption, and 4) server update. The study evaluates the time consumption in these four phases, respectively.

1) Local Update: Clients compute the gradients and encrypt them with the public key.
2) Aggregation: Server receives and aggregates the encrypted gradients from clients.
3) Decryption: Clients (partially) decrypts the gradients.
4) Server Update: Server updates the latent factor matrix using (partially) decrypted aggregated gradients.

Table V presents the computation time in each epoch, from which we can make the following observations: 1) For our proposed PrivMVMF, the aggregation and decryption process take up most of the time. More work can be done to reduce the complexity of the operation of the encrypted gradient as well. 2) The server update takes much more time in SecMVMF, since the parameters are updated in the encrypted form, and thus, our proposed method is more efficient than the SecMVMF.

VII. Conclusion

To understand the privacy risks in federated MVMF recommender systems, this article provides a theoretical analysis of the server reconstruction attack in four scenarios. It also proposes PrivMVMF, a privacy-preserving federated MVMF framework enhanced with HE, to overcome the information leakage problem. Empirical studies on MovieLens-1M dataset show that:

1) in FedMVMF, the FL server could infer users’ rating and attribute with accuracy $> 80\%$ using plaintext gradients;
privMVMF can protect user privacy well compared with FedMVMF; the distributed decryption in PrivMVMF reduces the complexity of server update and protects data privacy against the collaboration between decrypters and server. Future work involves the following directions. First, communication time could be investigated in PrivMVMF framework. Second, it is interesting to improve the efficiency of HE since it is time-consuming to perform the operation on the encrypted gradients.

**APPENDIX**

A. Attack for SGD-InclUnc

After two epochs, the gradients FL server receives from user $i$ is given by

$$f^t(i, j) = c_{i,j} \left( r_{i,j} - p_i^t (q_j^{t-1}) \right) p_i^t, j \in [1, m]$$

$$f^{t-1}(i, j) = c_{i,j} \left( r_{i,j} - p_i^{t-1} (q_j^{t-1}) \right) p_i^{t-1}, j \in [1, m]$$

$$f^t(i, d_u) = \left( x_{i,d_u} - p_i^t (u_d_u^{t-1}) \right) p_i^t, d_u \in [1, l_x]$$

$$f^{t-1}(i, d_u) = \left( x_{i,d_u} - p_i^{t-1} (u_d_u^{t-1}) \right) p_i^{t-1}, d_u \in [1, l_x].$$

In SGD, the user latent factor is updated using (9) and (10). Plugging into the first gradient of (44), we have

$$f_i^n(i, j) = c_{i,j} \left( r_{i,j} - \left( p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i^t} \right) (q_j^{t-1} + \Delta q_j^t) \right) \times \left( p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i^{t-1}} \right).$$

Equation (45) is a multiplication of two terms. By looking at the first term, we have

$$c_{i,j} \left( r_{i,j} - \left( p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i^t} \right) (q_j^{t-1} + \Delta q_j^t) \right) \times \left( p_i^{t-1} - \gamma \frac{\partial J}{\partial p_i^{t-1}} \right).$$

where

$$c_{i,j} = -2 \gamma \left( \sum_k f_k^{t-1}(i, k) q_k^{t-1} + \lambda \sum_{d_u} f_{i,d_u}^{t-1}(i, d_u) u_{d,u,d}^{t-1} \right) (q_j^{t-1})' + 2 \lambda_2 \frac{\partial J}{\partial p_i^{t-1}}$$

The second term of (45) is given by

$$F_i^n = 2\gamma \left( \sum_k f_k^{t-1}(i, k) q_k^{t-1} + \lambda \sum_{d_u} f_{i,d_u}^{t-1}(i, d_u) u_{d,u,d}^{t-1} \right).$$

Then the multiplication gives equation system (33).

B. Evaluation Metrics for PrivMVMF

The study sets the rating threshold to be 4, and the number of items recommended to be 10 per user. The metrics are defined as

$$\text{RMSE} = \frac{1}{|\text{usr}|} \sum_i \frac{1}{|O_i|} \sum_{r_{i,j} \in O_i} (\hat{r}_{i,j} - r_{i,j})^2$$

$$\text{Precision} = \frac{1}{|\text{usr}|} \sum_i \frac{t^p_i}{t^p_i + f^t_i}$$

$$\text{Recall} = \frac{1}{|\text{usr}|} \sum_i \frac{t^p_i}{t^p_i + f^t_i}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where $t^p_i$, $f^t_i$, and $f^t_i$ denote the true positive, false positive, and false negative for user $i$, respectively.

C. Table and Figures

| TABLE VI | ACCURACY FOR ATTACK WITH FOUR METHODS |
|------------------|----------------------------------|
|                | **SemiALS** | **SGD** |
|                | **ObsOnly** | **InclUnc** | **ObsOnly** | **InclUnc** |
| Hybr            | Rating      | 0.91      | 0.60      | 0.82      |
|                  | Attribute   | 0.99      | 0.86      | 0.84      |
| Broyden         | Rating      | 0.91      | 0.53      | 0.73      | 0.71 |
|                  | Attribute   | 0.94      | 0.60      | 0.85      | 0.73 |
| Scalar          | Rating      | 0.01      | 0.89      | 0.66      | 0.24 |
|                  | Attribute   | 0.25      | 0.89      | 0.74      | 0.42 |
| Anderson        | Rating      | 0.55      | 0.49      | 0.77      | 0.57 |
|                  | Attribute   | 0.78      | 0.46      | 0.88      | 0.62 |

Hybr, Broyden, Scalar, and Anderson denote modified Powell’s hybrid method, Broyden’s bad method, Scalar Jacobian approximation, and Anderson mixing, respectively. For SemiALS update with InclUnc, Hybr does not return the solution so the value is “/.” The selected method for each scenario is marked with boldface.

![Fig. 3. Accuracy of two InclUnc cases after adding Laplace noises from 1 to 10000.](image-url)
REFERENCES

[1] P. Kairouz et al., “Advances and open problems in federated learning,” Foundations Trends Mach. Learn., vol. 14, no. 1, pp. 1–210, 2021.

[2] D. Cortes, “Cold-start recommendations in collective matrix factorization,” 2018, arXiv:1809.00366.

[3] A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2008, pp. 650–658.

[4] Y. Seroussi, F. Bohnert, and I. Zukerman, “Personalised rating prediction for new users using latent factor models,” in Proc. 22nd ACM Conf. Hypertext Hypermedia, 2011, pp. 47–56.

[5] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, “Learning attribute-to-feature mappings for cold-start recommendations,” in Proc. IEEE Int. Conf. Data Mining, 2010, pp. 176–185.

[6] A. Flanagan, W. Oyomno, A. Grigorievskiy, K. E. Tan, S. A. Khan, and M. Ammad-Ud-Din, “Federated multi-view matrix factorization for personalized recommendations,” in Machine Learning and Knowledge Discovery in Databases. Berlin, Germany: Springer, 2021, pp. 324–347.

[7] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, 2009.

[8] T. Li, C. Li, J. Luo, and L. Song, “Wireless recommendations for Internet of Vehicles: Recent advances, challenges, and opportunities,” Intell. Converged Netw., vol. 1, no. 1, pp. 1–17, 2020.

[9] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” Adv. Artif. Intell., vol. 2009, pp. 1–19, 2009.

[10] E. Duriakova et al., “PDMFRec: A decentralised matrix factorisation with tunable user-centric privacy,” in Proc. 13th ACM Conf. Recommender Syst., 2019, pp. 457–461.

[11] Y. Lin et al., “Meta matrix factorization for federated rating predictions,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2020, pp. 981–990.

[12] M. Ammad-Ud-Din et al., “Federated collaborative filtering for privacy-preserving personalized recommendation system,” 2019, arXiv:1901.09888.

[13] Y. Fang and L. Si, “Matrix co-factorization for recommendation with rich side information and implicit feedback,” in Proc. 2nd Int. Workshop Inf. Heterogeneity Fusion Recommender Syst., 2011, pp. 65–69.

[14] M. Jamali and M. Ester, “A matrix factorization technique with trust propagation for recommendation in social networks,” in Proc. 4th ACM Conf. Recommender Syst., 2010, pp. 135–142.

[15] K. Zhou, S.-H. Yang, and H. Zha, “Functional matrix factorizations for cold-start recommendation,” in Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf., 2011, pp. 315–324.

[16] D. Chai, L. Wang, K. Chen, and Q. Yang, “Secure federated matrix factorization,” IEEE Intell. Syst., vol. 36, no. 5, pp. 11–20, Sep./Oct. 2021.

[17] E. Shmueli and T. Tassa, “Secure multi-party protocols for item-based collaborative filtering,” in Proc. 11th ACM Conf. Recommender Syst., 2017, pp. 89–97.

[18] S. Meng, Z. Gao, Q. Li, H. Wang, H. Dai, and L. Qi, “Security-driven hybrid collaborative recommendation method for cloud-based IoT services,” Comput. Secur., vol. 97, 2020, Art. no. 101950.

[19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning,” ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, pp. 1–19, 2019.

[20] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proc. Int. Conf. Theory Appl. Cryptographic Techn., 1999, pp. 223–238.

[21] I. Damgård and M. Jurik, “A generalisation, A simplification and some applications of Paillier’s probabilistic public-key system,” in Proc. Int. Workshop Public Key Cryptography, 2001, pp. 119–136.

[22] T. Veugen, T. Attema, and G. Spini, “An implementation of the Paillier crypto system with threshold decryption without a trusted dealer,” Cryptol. ePrint Arch., 2019. [Online]. Available: https://eprint.iacr.org/2019/1136

[23] J. J. More, B. S. Garbow, and K. E. Hillstrom, User Guide for MINPACK-1. Argonne, IL, USA: Argonne Nat. Lab., 1980.

[24] C. G. Broyden, “A class of methods for solving nonlinear simultaneous equations,” Math. Comput., vol. 19, no. 92, pp. 577–593, 1965.

[25] W. La Cruz, J. M. Martínez, and M. Raydan, “Spectral residual method without gradient information for solving large-scale nonlinear systems of equations,” Math. Comput., vol. 75, no. 255, pp. 1429–1449, 2006.

[26] D. G. Anderson, “Iterative procedures for nonlinear integral equations,” J. ACM, vol. 12, no. 4, pp. 547–560, 1965.

[27] I. Damgård and M. Jurik, “A generalisation, A simplification and some applications of Paillier’s probabilistic public-key system,” in Proc. Int. Workshop Public Key Cryptography, 2001, pp. 119–136.

[28] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” Knowl.-Based Syst., vol. 46, pp. 109–132, 2013.

[29] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 2951–2959.

[30] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, “Parallel matrix factorization for recommender systems,” Knowl. Inf. Syst., vol. 41, no. 3, pp. 793–819, 2013.

[31] M. Boreale, “Quantifying information leakage in process calculi,” Inf. Comput., vol. 207, no. 6, pp. 699–725, 2009.