Hindawi
Security and Communication Networks
Volume 2022, Article ID 8607234, 15 pages
https://doi.org/10.1155/2022/8607234

Research Article

Efficient Personalized Recommendation Based on Federated Learning with Similarity Ciphertext Calculation

Xixian Wang, Xiaoming Wang, Binrui Huang, Mingzhan Dai, and Jianwei Li

Department of Computer Science, Jinan University, Guangzhou 510632, China

Correspondence should be addressed to Xiaoming Wang; twxm@jnu.edu.cn

Received 25 April 2022; Accepted 1 August 2022; Published 16 September 2022

Academic Editor: AnMin Fu

With the popularity of big data, people get less useful information because of the large amount of data, which makes the Recommender System come into being. However, the privacy and accuracy of the Recommender System still have great challenges. To address these challenges, an efficient personalized recommendation scheme is proposed based on Federated Learning with similarity ciphertext calculation. In this paper, we first design a Similarity calculation algorithm based on Orthogonal Matrix in Ciphertext (SOMC), which can compute the Similarity between users' demand and Items' attributes under ciphertext with a low calculation cost. Based on SOMC, we construct an efficient recommendation scheme by employing the Federated Learning framework. The important feature of the proposed approach is improving the accuracy of recommendation while ensuring the privacy of both the users and the Agents. Furthermore, the Agents with good performance are selected according to their Reliability scores to participate in the federal recommendation, so as to further make the accuracy of recommendation better. Under the defined threat model, it is proved that the proposed scheme can meet the privacy requirements of users and Agents. Experiments show that the proposed scheme has optimized accuracy and efficiency compared with existing schemes.

1. Introduction

With the rise of big data, information overload makes people get more useless information. Therefore, Recommender System is gradually popular, the purpose of Recommender System is providing users with personalized online products or service recommendations, and Recommender System has become an important way to resolve information overload issues, which brings opportunities and challenges to education, medical and other industries. However, the current Recommender System has many potential risks, of which the privacy disclosure is one of the primary concerns [1]. In general, the Recommender System consists of two parts: recommender server and users. In order to get a superior model for recommendation, the traditional Recommender System uses a central architecture and usually collects a large amount of feedback information such as user preferences [2]. But the information is often sensitive to users and can lead to serious privacy and security risks: the users' raw data may be disclosed from the feedback information from some programs [3]. For example, Recommender System only needs to obtain the records about users watching movies, and can infer some privacy information (e.g., age, income, medical history, etc.). In addition, Recommender System may collect users' personal data and share it with a third party to obtain profits [4]. Once this information is abused, the consequences are unimaginable.

As a result, people are increasingly concerned about their data privacy, and they hope that their private information will not be known by Internet applications. In existing research, there are many methods to protect data privacy, such as anonymity, differential privacy, homomorphic encryption and Federated Learning (FL). Federated Learning is a popular tool to reduce privacy risks. Thus, FL has received increasing attention. Such as [5], it uses FL to protect users' healthcare data privacy in the big data scenario. And in [6], based on the framework of FL, a recommendation scheme is proposed. The scheme defines multiple agents for

Table 1: Recommended method comparison.

Schemes	Encryption method	Cloud	Framework
Zhang et al. [12].	BGN	honest-but-curious	centralized
Xu et al. [13]	Modified Paillier	honest-but-curious	centralized
Peng et al. [10]	MTDT-PKC	honest-but-curious	centralized
Zhou et al. [6]	None	fully trusted	FL
our scheme	SOMC	honest-but-curious	FL

collaborative recommendation, and the FL framework is used to ensure the users' data privacy and to make the recommended items more reliable. However, in the system model of this scheme, the settings of the cloud are completely trusted, which is difficult to achieve in practice, and all the records are stored in the cloud in plaintext, so there is still a risk of exposing privacy.

In other aspects, the concept of similarity is often introduced in order to achieve better recommendations [7, 8]. Generally, in order to ensure the accuracy of recommended items, many Recommender Systems need to calculate two parameters [9]. One is the similarity between users' demands and the attributes of recommended items, and the other is the users' evaluations about recommendation items. For the former, the higher similarity implies that the items will be more appropriate for the needs of users. For the latter, it means that items will have higher recommendation priority. After receiving recomendation items, the users will evaluate the recommendation items, i.e., submit feedback scores. The Recommender System will collect these scores and calculate the reliability of the recommendation agents according to the evaluations.

Obviously, it is essential to protect users' privacy in the current Recommender System. When users submit requirements or estimates, they wish their information to be protected. This is because the users' demands and evaluations are usually related to the privacy of information [10]. To solve these problems, some feasible solutions were provided, such as [9, 11], but there are some problems: the encryption method is too complicated and the computation load is too heavy, and the server is set to be fully trusted, which is difficult to achieve in reality.

To address these above challenges, we first set the cloud to be semi-trusted, which means that it is possible to spy on users' privacy. This makes our scheme have better practicability. Then we take the similarity as the criterion to select the recommendation items to improve the accuracy of recommendations. Next each user evaluates the recommendation items after receiving items, and the Cloud server evaluates the reliability of the recommendation agents according to the evaluations. Last, our scheme uses the FL framework to improve recommendation accuracy and protect users' privacy. The comparison of our scheme and some studies about recommendation is discussed in Table 1.

In this paper, we focus on how to perform a more secure and effective recommendation process with a Federated Learning framework. Our contribution is summarized as follows:

(i) We design a Similarity calculation algorithm based on Orthogonal Matrix in Ciphertext (SOMC), which can not only reduce the calculation overhead, but also ensure the privacy and security of users.

(ii) Based on SOMC, we construct an efficient recommendation scheme by employing the Federated Learning model, named Efficient Recommendation Based on Federated Learning (ERBFL). ERBFL can securely aggregate the recommendation weight from the multiple Agents and calculate the similarity between users' demand and Items' attributes under ciphertext, so as to improve the accuracy of recommendation while ensuring the privacy of both the users and the Agents. Moreover, the Agents with the good performance are selected according to their Reliability scores to participate in the federal recommendation, so as to further make better the accuracy of recommendation.

(iii) Under the defined threat model, we prove the proposed scheme is to meet the privacy requirements of users and Agents. In addition, we conduct experiments that show our scheme has optimized accuracy and efficiency compared with existing schemes.

The rest of this paper is divided into six parts. State-of-the-art solutions about the privacy protection problem of Recommender System are described in Section 2. Following this we present the preliminaries of proposed work in Section 3. ERBFL scheme is discussed in detail in Section 4, and its security analysis are presented in Section 5. Comprehensive performance evaluation is given in Section 6. Conclusions are drawn in Section 7.

2. Related Works

Recent years, Recommender System helps to solve the problem of information overload while providing personalized information retrieval [14]. However, in order to improve the recommendation efficiency, the system requires the personal information of users, which is a serious privacy concern for many users [15, 16]. Recent research indicates that there are two methods to solve the privacy problem of the Recommender system: Architecture-based and Algorithms-based [17].

Architecture-based solutions usually exploit distributed data storage to minimize the threat of data leakage, but some existing schemes [18, 19] still have the risk of leaking users' privacy and increasing the computing burden of local devices. Algorithms-based solutions are different from them, usually utilize an encryption algorithm to protect the original sensitive data. Several studies have revealed that the encryption-based solution can reduce the risk of leaking users' privacy [17]. At present, some researchers have

proposed various solutions to the privacy protection recommendation system, such as [20–22]. Erkin et al. [20] prevented the malicious users and server from accessing privacy-sensitive data by using homomorphic encryption. Kaur et al. [21] introduced arbitrary distributed data based on two techniques: multi-party random masking and polynomial aggregation, which can improve the efficacy of recommender system and protect users' privacy. Ma et al. [22] realized a friend recommendation in a way of protecting privacy by utilizing social attributes and trust relationship about online social network users. However, these schemes all cause heavy computational overhead.

Besides, to ensure the accuracy of the recommendation items, the current Recommender System will take the similarity as the reference parameter of recommendation. In [23], the similarity between users is calculated to help get more accurate recommendation results. In [24], the similarity between users and items is calculated. To further improve the accuracy of recommendations, asymmetric similarity method was mentioned in [25], where weighted schemes made recommendations more accurate, taking into account the varying degrees of influence of each factor. But that will raise privacy issues. This is due to the similarity calculation in the recommendation process will involve the user's privacy information, the recommender server may be able to know the users' personal interests, which contain sensitive information. Some studies have proposed solutions to this problem, such as [12, 13, 26]. Li et al. [26] proposed a privacy-preserving scheme to implement a contextual recommendation for online social communications, which can hide users' real identities from other users by using pseudonyms. Zhang et al. [12] used BGN Cryptosystem to protect users' privacy in the recommendation process, which utilized homomorphic property to calculate the similarity between two users in the ciphertext domain. However, the scheme causes heavy computing burden on the user side, in which the computation of bilinear pairs is used. In addition, Xu et al. [13] focused on similarity and evaluation of truth to propose a privacy-preserving recommendation scheme. However, it used a centralized framework so that it is unsuitable for the scenario with multiple recommenders. FL can solve this problem very well. Many existing studies utilize FL to achieve collaboration while protecting data privacy. In [27], a scheme called EPPDA was proposed, which can resist the reverse attack by utilizing secret sharing and an efficient privacy-preserving data aggregation method for FL.

In order to address the above privacy problems and improve accuracy, Federated Recommendation System(-FedRec) is proposed, the goal of FedRec is to collaborate with multiple parties to complete the more efficient and accurate recommendation process without directly accessing each other's private data [4]. To perform the recommendation process in multiple data-owners scenarios, Zhou et al. [6] implemented a privacy-preserving contextual recommendation which used Federated Learning framework to protect users' privacy and increase efficiency of recommendation. However, in the scheme, the center server is set to be fully trusted, which is hard to meet in practice [28], and the center server processes the sensitive data in plaintexts, which raises privacy risk [29, 30]. For instance, some malicious Cloud servers may apply gradient inference attacks or model inversion attacks to damage the client's sensitive data, as described in [31, 32]. Hence, it is impractical that schemes are under the assumption that the Cloud server is full-trusted.

3. Preliminaries

3.1. Orthogonal Matrices. Orthogonal matrix is an important type of matrix [33]. When the product of a matrix and its transpose gives the value of the identity matrix, the matrix is called an orthonormal matrix.These matrices are useful in many scientific applications related to vectors.

Suppose A and B are $n \times n$ orthogonal matrices which can also be represented by $A = (\overrightarrow{a_1}, \overrightarrow{a_2}, \ldots, \overrightarrow{a_n})$, $B = (\overrightarrow{b_1}, \overrightarrow{b_2}, \ldots, \overrightarrow{b_n})$, where $\overrightarrow{a_i}, \overrightarrow{b_i} \in \mathbb{R}^n$, $1 \le i \le n$. The symbol $\mathbb{R}^n$ represents an infinitely large set of vectors, where each vector has n components. Mathematically, $\overrightarrow{v} = (v_1, v_2, \ldots, v_n)^\top$, $v_i \in \mathbb{R}; 1 \le i \le n$, where $\top$ is the transpose symbol. A is a orthogonal matrix, if and only if:

$$\overrightarrow{a_i}^\top \cdot \overrightarrow{a_j} = \sqrt{a_{i1}a_{j1} + a_{i2}a_{j2} + \cdots + a_{in}a_{jn}} = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases} \quad (1)$$

Meanwhile, let $A^\top$ and $B^\top$ are the transpose matrices of A and B, respectively. Then they have the properties as follows:

(i) $A \cdot A^\top = E$, $B \cdot B^\top = E$;
(ii) $A^{-1} = A^\top$, $B^{-1} = B^\top$;
(iii) $(AB)^\top = B^\top A^\top$;
(iv) $(AB)^{-1} = B^{-1}A^{-1}$;

where E is the identity matrix of the order $n \times n$. $(\cdot)^{-1}$ is the inverse of matrix $(\cdot)$.

Interestingly, the inverse and transpose of an orthogonal matrix are the same that make the inverse operation of the orthogonal matrix simple and fast. In light of this, it is efficient to use orthogonal matrices as cipher keys.

3.2. Differential Privacy. Differential privacy (DP) is a promising technology which can solve privacy problems [34]. We have adopted the local version of DP for users, which can protect users' privacy under untrusted Cloud server and Agents. We introduce several definitions about differential privacy as follows [35]:

Definition 1 (ε-Differential Privacy, ε-DP). A random mechanism $\mathscr{L}$ is said ε-indistinguishable if for all pairs $\overrightarrow{X}, \overrightarrow{X'} \in D^p$ which differs in only one entry, for all adversaries $\mathscr{A}$, and for all transcripts t, where D^p is p-dimensional vector data set:

$$\left| \ln \left(\frac{\Pr\left[\mathscr{T}_{\mathscr{A}}(\overrightarrow{X}) = t\right]}{\Pr\left[\mathscr{T}_{\mathscr{A}}(\overrightarrow{X'}) = t\right]} \right) \right| \le \varepsilon. \quad (2)$$

In the definition, a privacy parameter ε is predefined in order to control the privacy budget, which means that if ε is smaller, the privacy protection will be stronger.

Definition 2 (L_1-Sensitivity). Assume that f is a numeric query function, and f maps a data set D^p into a p-dimensional real space $\mathbb{R}^p$ such as $f: D^p \longrightarrow \mathbb{R}^p$. For any pair of adjacent data sets D^p and D'^p, the sensitivity about f is defined as:

$$\Delta f = \max_{D^p, D\prime^p} \left\| f(D^p) - f\left(D\prime^p\right) \right\|_{L_1}, \tag{3}$$

where $\| \cdot \|_{L_1}$ denoted the L_1 norm.

Theorem (Random-Laplace Mechanism). Let $b \in \mathbb{R}^+$, and f is a numeric query function which maps a p-dimensional domain D^p p-dimensional real space $\mathbb{R}^p$, such as $f: D^p \longrightarrow \mathbb{R}^p$. The mechanism $\mathscr{L}$ randomly selects t elements in vector $\overrightarrow{X}$, where t satisfies the following condition: $t = \lceil p/3 \rceil$.

$$\mathscr{L}(\overrightarrow{X}) = f(\overrightarrow{X}) + \left(\mathrm{Lap}_1(b), \mathrm{Lap}_2(b), \ldots, \mathrm{Lap}_p(b)\right), \tag{4}$$

provides ε-Differential Privacy, where selected subscript j: $\mathrm{Lap}_j(b) = 0$. Other $\mathrm{Lap}_i(b)$ is drawn from the Laplace distribution with scaling parameter b, where b's density function is

$$\mathrm{d}(b) = \frac{1}{2b} \exp\left(\frac{\neg|x|}{b}\right), \tag{5}$$

where $b = \Delta f/\varepsilon$ is bound by the privacy budget ε and the sensitivity of function Δf.

3.3. Digital Signature. Digital Signature is used to prevent information from being tampered with [36]. Digital Signature Λ is composed of the algorithms Λ.KeyGen, Λ.Sign, Λ.Verify defined as follows:

$\Lambda.\mathrm{KeyGen}(1^\lambda) \longrightarrow (sk, pk)$: It takes as input 1^λ, and outputs a key pair (sk, pk), where λ is a security parameter.

$\Lambda.\mathrm{Sign}(sk, m) \longrightarrow \sigma$: It takes as input the private key sk and the message m, and outputs the signature σ.

$\Lambda.\mathrm{Verify}(pk, m, \sigma) \longrightarrow \{0, 1\}$: It takes as input the public pk, the message m, and the signature σ, and it outputs 0 if the signature σ is invalid and 1 otherwise.

3.4. Federated Learning. In order to improve the accuracy of predicting users' next input, Google built a horizontal federated model, and Google first proposed a concept of Federated Learning (FL) in 2017 [37]. FL is a distributed deep learning framework, and it allows multiple clients such as IoT devices and mobile devices to train the model, but the sensitive privacy data in the device remains local, the joint model trained by the server is sent back to each client, and the client continues to learn the new model. The process iterates to get the optimal model [38]. Therefore, the privacy of the client has been protected to a certain extent.

4. Design of Proposed Scheme

Before introducing the proposed scheme, we first give the basic symbols involved in the scheme. We define U_i is Agen_i's user group, U is a set of user groups: $U = \{U_i, i = 1, 2, \ldots, |U|\}$ where $|U|$ is the size of Agents. Note that here $U_i = \left\{u_{i,j}, j = 1, 2, \ldots, |U_i|\right\}$ is a subgroup of U based on geographic location, in which $|U_i|$ is a size of users included in U_i. For each user $u_{i,j} \in U_i$, we define the vector $\overrightarrow{A_{i,j}} = (a_{ij,1}, a_{ij,2}, \ldots, a_{ij,p})^\top$ as $u_{i,j}$'s demand vector, where $a_{ij,l}$ refers to an attribute about $u_{i,j}$'s demand.

Each $U_i \in U$ has a corresponding Agen_i, and Agen_i holds a Item set $\mathscr{I}_i = \left\{I_{i,i\prime}, i' = 1, 2, \ldots, n_i\right\}$ where n_i is the size of Agen_i's Items. We define the vector $\overrightarrow{B_{i,i\prime}} = (b_{ii\prime,1}, b_{ii\prime,2}, \ldots, b_{ii\prime,p})^\top$ as $I_{i,i'}$'s attribute vector, where $b_{ii\prime,l}$ refers to an attribute about $I_{i,i\prime}$. Besides, Agen_i's recommendation weight matrix consists of p weight parameters $w_{i,r}$ $(r = 1, 2, \ldots, p)$, denoted as $W_i = \mathrm{diag}(w_{i,1}, w_{i,2}, \ldots, w_{i,p})$ which is a $p \times p$ diagonal matrix. Notes: each $w_{i,r}$ $(r = 1, 2, \ldots, p)$ represents the impact factor of the r-th attribute on the recommendation item. According to the property of diagonal matrix, we can know $W_i^\top = W_i$.

4.1. System Architecture. As shown in Figure 1, our system consists of four components as follows: users, Agent, Cloud server and Trusted authority (TA).

(i) *User*: Considering our system, the user submits an encrypted demand vector to Agent, and makes the recommendation request. After receiving the recommended Item, the user sends an encrypted feedback score to Agent.

(ii) *Agent*: Agent has its own user group and Item set. It can recommend appropriate Items to users belonging to its user group, after receiving requests the users send.

(iii) *Cloud server*: It is considered to be honest but curious. It is responsible to calculate similarity between users' demand and Items' attributes, and calculate Agent's Reliability score after receiving request. It will follow the proposed scheme for executing requests received. But we do not exclude the possibility that it will disclose the privacy of users. Additionally, the server is not allowed to collude with Agents.

(iv) *Trusted authority (TA)*: It is a fully trusted third party and responsible to generate and distribute keys to users, Agents and Cloud server.

Specially, all users submit their requests for recommendation to Agents in ciphertext, and the Agents need to cooperate with the Cloud server for recommendation. According to the Agents' Reliability scores, the server judges whether it needs to select several Agents to form a group for

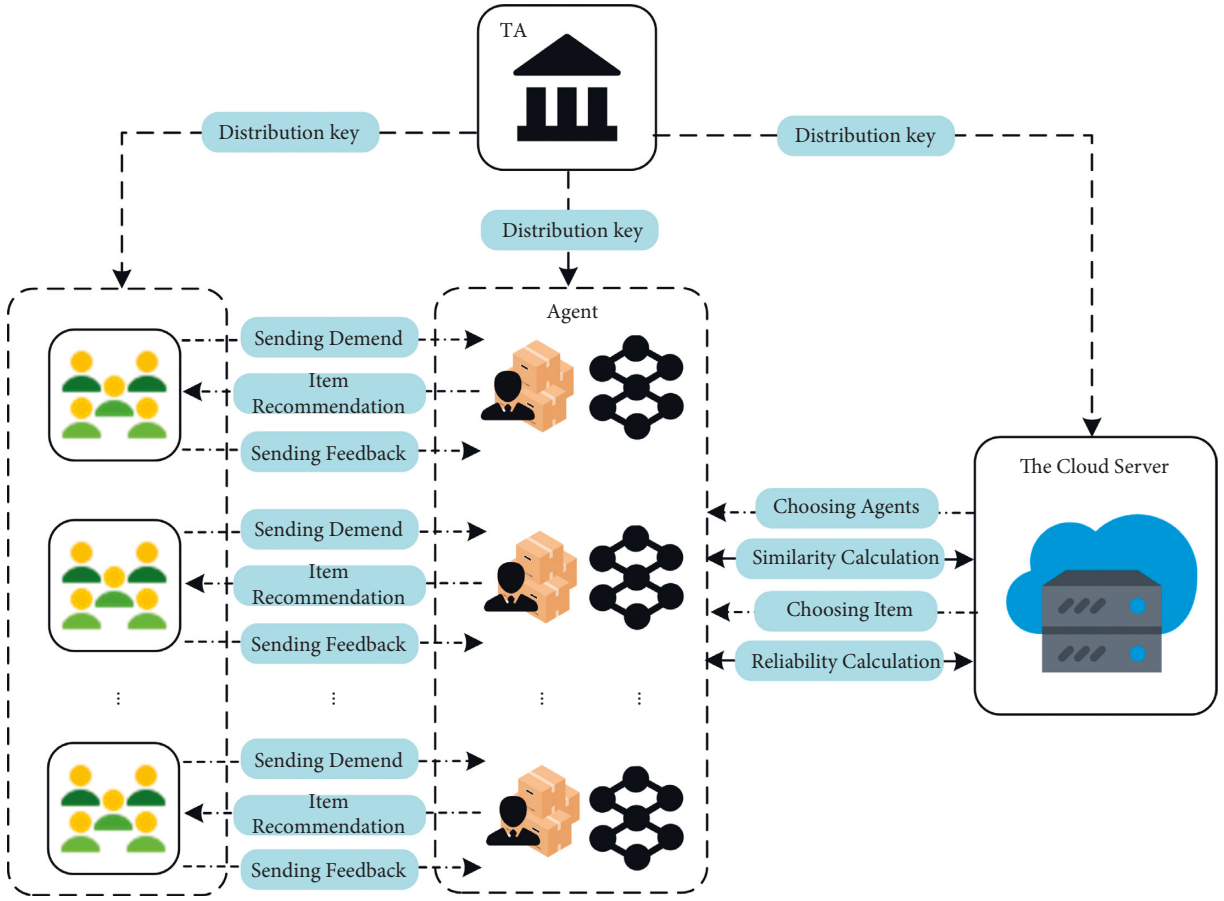

Figure 1: System model.

federal recommendation. After recommendation finished, the users will the give encrypted feedback scores to Agents, then the Agents forwards to the Cloud server, and the Cloud server updates the Agents' Reliability scores.

4.2. Similarity Calculation Algorithm Based on Orthogonal Matrix in Ciphertext. In order to improve the accuracy of recommendation items, we need to calculate the similarity between users' demand and items' attributes. Meanwhile, to strengthen the privacy preservation of recommendation, users' demand and items' information should be compared under ciphertext form. Therefore, we propose an efficient Similarity calculation algorithm based on Orthogonal Matrix in Ciphertext (SOMC), which can protect sensitive information privacy by lightweight encryption. The SOMC comprises the following three algorithms **KeyGen**, **Enc**, and **Eval**, and detailed as follows.

SOMC.KeyGen $(p) \longrightarrow (K_A, K_{s,i}, K_i, L, J_i)$: It takes as input p where p is the dimension of the matrix, and outputs the secret keys $K_A, K_{s,i}, K_i, L$ and J_i:

(i) It first takes as input p, generates $p \times p$ orthogonal matrices $K_A, K_{s,i}$ and L as follows:

$$K_A = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pp} \end{pmatrix}_{p\times p},$$
$$K_{s,i} = \begin{pmatrix} y_{11} & \cdots & y_{1p} \\ \vdots & \ddots & \vdots \\ y_{p1} & \cdots & y_{pp} \end{pmatrix}_{p\times p}, \quad (6)$$
$$L = \begin{pmatrix} q_{11} & \cdots & q_{1p} \\ \vdots & \ddots & \vdots \\ q_{p1} & \cdots & q_{pp} \end{pmatrix}_{p\times p},$$

where $x_{ll}, y_{ll}, q_{ll} \in \mathbb{R}$.

(ii) Then, generating K_i and J_i according to the following formula: $K_i = K_A \cdot K_{s,i}$ and. $J_i = K_i \cdot L$

SOMC.Enc $(SK, M) \longrightarrow C_{SK}$: It takes as input secret key SK and plaintext M, where SK is K_i or J_i, and M is a $p \times p$ matrix (weight matrix) or a p-dimensional vector (demand or item vector), and outputs the ciphertext C_{SK}.The encryption process is as follows:

$$C_{SK} = SK^{\top} \cdot M. \quad (7)$$

SOMC.Eval $(L, C_k, C_j) \longrightarrow \mathrm{sim}_{k,j}$: On input the secret key L, the ciphertext C_k and C_j, where C_k, C_j are obtained by algorithm **SOMC.Enc**(K_i, M_k) and **SOMC.Enc**(J_i, M_j), the similarities $\mathrm{sim}_{k,j}$ between M_k (item vector) and M_j (demand vector) as output. Note that here $M_k = (m_{k,1}, m_{k,2}, \ldots, m_{k,p})^\top$ and $M_j = (m_{j,1}, m_{j,2}, \ldots, m_{j,p})^\top$ are p-dimensional vectors. The calculation process is as follows:

$$\begin{aligned}\mathrm{sim}_{k,j} &= \left(C_j - L^\top C_k\right)^\top \cdot \left(C_j - L^\top C_k\right)\\ &= \left(J_i^\top M_j - L^\top K_i^\top M_k\right)^\top \cdot \left(J_i^\top M_j - L^\top K_i^\top M_k\right)\\ &= \left(J_i^\top M_j - J_i^\top M_k\right)^\top \cdot \left(J_i^\top M_j - J_i^\top M_k\right)\\ &= \left(M_j - M_k\right)^\top J_i \cdot J_i^\top \left(M_j - M_k\right)\\ &= \left(M_j - M_k\right)^\top \cdot \left(M_j - M_k\right)\\ &= \sum_{x=1}^{p} \left(m_{k,x} - m_{j,x}\right)^2.\end{aligned} \tag{8}$$

Therefore, we can calculate the similarity between C_k and C_j without obtaining the plaintext about them.

4.3. Efficient Recommendation Scheme Based on Federated Learning

4.3.1. System Initialization. In this phase, it is divided into two parts, firstly TA generates and distributes key, and then the Agents encrypt attribute information. In order to facilitate description, we take Agen_i and users $u_{i,j} \in U_i$ as an example to illustrate.

Firstly, TA uses the algorithm **SOMC.KeyGen**(p) described in the previous section to generate keys and system parameters as follows:

(i) Step 1: TA exploits algorithm **SOMC.KeyGen**(p) to generate $(K_A, K_{s,i}K_i, L, J_i)$, and then distributes $\{K_A, K_i, K_{s,i}\}$ to Agen_i; $\{L, K_{s,i}\}$ to the Cloud server; $\{J_i\}$ to each users $u_{i,j} \in U_i$ by secure channel. Note that here for Agen_a and Agen_b, if $a \neq b$, then $K_a \neq K_b, K_{s,a} \neq K_{s,b}$. Beside, each Agen_i has the same K_A.

(ii) Step 2: TA generates unique identification for each Agent and each Item. We define ID_i as Agen_i's identification and $Iid_{i,i_{l'}}$ as for Item $I_{i,i_{l'}}$.

(iii) Step 3: On input 1^λ where λ is the security parameter, TA runs the signature algorithm $\Lambda.\mathrm{KeyGen}(1^\lambda)$ to generate $(pk_{\Lambda,i,j}, sk_{\Lambda,i,j}, pk_{\Lambda,S}, sk_{\Lambda,S})$. Then, TA distributes $sk_{\Lambda,i,j}$ to $u_{i,j} \in U_i$; $sk_{\Lambda,S}$ to Cloud server. And then TA public $(pk_{\Lambda,i,j}, pk_{\Lambda,S})$.

(iv) Step 4: TA sets each Agen_i's Reliability score RES_i to 0 and its update times count_i to 0.

Secondly, Agen_i has own Item set $\mathcal{I}_i = \left\{I_{i,i_{l'}}, i' = 1, 2, \ldots, n_i\right\}$, Agen_i generates the encrypted Item information for $\left\{\overrightarrow{B_{i,i_{l'}}}, i' = 1, 2, \ldots, n_i\right\}$:

$$\overrightarrow{B'_{i,i}} = \mathbf{SOMC.Enc}\left(K_i, \overrightarrow{B_{i,i_{l'}}}\right)\left(i' = 1, 2, \ldots, n_i\right), \tag{9}$$

and then upload $\left\{\overrightarrow{B'_{i,i}}, i' = 1, 2, \ldots, n_i\right\}$ to the Cloud server for storage.

4.3.2. Item Recommendation. In this subsection, we introduce how to recommend a suitable Item to a user when the Agent cooperates with the Cloud server. The whole process can be divided into three parts: (1) User sends demand; (2) The Cloud chooses Agents; (3) Item choosing and recommendation. The pseudo-code of the Item Recommendation is given in Algorithm 1.

(1) User sends demand. To avoid the disclosure of user needs that contain a large amount of private information, user $u_{i,j}$ will exploit **SOMC.Enc** to encrypt message $\overrightarrow{A_{i,j}}$:

$$\overrightarrow{A^*_{i,j}} = \mathbf{SOMC.Enc}\left(J_i, \overrightarrow{A_{i,j}}\right) = \left(a'_1, a'_2, \ldots, a'_p\right)^\top, \tag{10}$$

To resist *brute-force exhausted attack* [39], we apply a Laplace mechanism [35] by adding noise to $\overrightarrow{A^*_{i,j}}$. Although this noise will cause an error, we will experiment later to prove that the error is in the acceptable range. First, $u_{i,j}$ randomly selects t elements in $\overrightarrow{A^*_{i,j}}$ to form a set Sub, where t satisfies the following condition: $t = \lceil p/3 \rceil$ and then adds a Laplace noise:

$$\begin{aligned}\overrightarrow{A'_{i,j}} &= \left(a''_{ij,1}, a''_{ij,2}, \ldots, a''_{ij,p}\right)^\top\\ &= \begin{cases} a''_{ij,x} = a_{ij,x}{}' + \mathrm{Lap}(b) & a_{ij,x}{}' \in \mathrm{Sub},\\ a''_{ij,x} = a_{ij,x}{}' & a_{ij,x}{}' \notin \mathrm{Sub},\end{cases}\end{aligned} \tag{11}$$

where b is Laplace parameter. To simplify, we express the noise as a vector $\overrightarrow{\mathrm{Lap}_{b(i,j)}}$. We can get $\overrightarrow{A'_{i,j}} = \overrightarrow{A^*_{i,j}} + \overrightarrow{\mathrm{Lap}_{b(i,j)}}$. Then $u_{i,j}$ generates a signature $\overrightarrow{\sigma_{i,j}} = \Lambda.\mathrm{Sign}(sk_{\Lambda,i}, \overrightarrow{A'_{i,j}})$.

Afterward, user $u_{i,j}$ submits $(\overrightarrow{A'_{i,j}}, \overrightarrow{\sigma_{i,j}})$ to Agen_i.

After receiving $(\overrightarrow{A'_{i,j}}, \overrightarrow{\sigma_{i,j}})$, Agen_i exploits **SOMC.Enc** to encrypt weight matrix:

$$W'_i = \mathbf{SOMC.Enc}(K_i, W_i), \tag{12}$$

and sends $(\overrightarrow{A'_{i,j}}, \overrightarrow{\sigma_{i,j}}, W'_i)$ and a request for collaboration about recommendation to the Cloud server.

(2) The Cloud chooses Agents. After receiving Agen_i's request, the Cloud server verifies the signature by performing the algorithm $\Lambda.\mathrm{Verify}(pk_{\Lambda,i}, \overrightarrow{A'_{i,j}}, \overrightarrow{\sigma_{i,j}})$. If it fails, the Cloud server refuses the request; otherwise, should determine whether Agen_i is eligible to make a recommendation alone or federated recommendation according to its Reliability score RES_i. The higher RES_i, the more credible the items Agen_i recommends.

If $\mathrm{RES}_i > r$, where r is a threshold set by TA, the Cloud server considers that Agen_i can recommend credible items to users alone, then go straight to **3) Item choosing and recommendation**. Otherwise, the Cloud server needs to

choose some other Agents to participate in the federal recommendation. Specific steps are as follows:

(i) Step 1: When $\mathrm{RES}_i \le r$, the Cloud server sorts all Agents according to their Reliability score by descending order firstly, then selects j Agents with a higher score to form a set AC_i with the following threshold condition: $\sum_{z=1}^{j} \mathrm{RES}_{\max-z} \le \partial \sum_{q=1}^{N} \mathrm{RES}_q$, where $\partial < 1$ is set by TA. We denote $\mathrm{RES}_{\max-z}$ is the $z-$ th max in the Reliability score of Agents.

(ii) Step 2: For each $\mathrm{Agen}_k \in AC_i$, the Cloud server calculates $PID_k = H(ID_k \| R_k)$, where R_k is a random number, then sends PID_k and recommendation request to each $\mathrm{Agen}_k \in AC_i$.

(iii) Step 3: After receiving the message from the Cloud server, $\mathrm{Agen}_k \in AC_i$ exploits **SOMC.Enc** to encrypt own weight matrix: $W_k' = \mathbf{SOMC.Enc}(K_k, W_k)$, and sends (PID_k, W_k') to the Cloud server.

(iv) Step 4: After receiving all encrypted weight matrices from $\mathrm{Agen}_k \in AC_i$, the Cloud server calculates $W'_{aggr,k}$ for each $\mathrm{Agen}_k \in AC_i$ by shared key between the Cloud server and the Agent:

$$\begin{aligned} W'_{aggr,k} &= K_{s,i} \cdot K_{s,k} \cdot W_k' \\ &= K_{s,i} \cdot K_{s,k} \cdot K_k^{\top} \cdot W_k \\ &= K_{s,i} \cdot K_{s,k} \cdot \left(K_A \cdot K_{s,k}\right)^{\top} \cdot W_k \\ &= K_{s,i} \cdot K_{s,k} \cdot K_{s,k}^{\top} \cdot K_A^{\top} \cdot W_k \\ &= K_{s,i} \cdot K_A^{\top} \cdot W_k \\ &= K_i^{\top} \cdot W_k. \end{aligned} \tag{13}$$

Notes that if the Cloud server finds some weight matrices missing according to the PID_k during collecting (PID_k, W_k'), the Cloud server will select another Agents and repeat the previous step.

Then the Cloud server aggregates weight for Federated Recommendation:

$$\begin{aligned} W_{\mathrm{fed}}' &= \frac{\sum_{\mathrm{Agen}_k \in AC_i} RES_k \cdot W_{\mathrm{aggr},k}'}{\sum_{\mathrm{Agen}_k \in AC_i} RES_k} \\ &= K_i^{\top} \cdot \frac{\sum_{\mathrm{Agen}_k \in AC_i} RES_k \cdot W_k}{\sum_{\mathrm{Agen}_k \in AC_i} RES_k}, \end{aligned} \tag{14}$$

where $\sum_{\mathrm{Agen}_k \in AC_i} \mathrm{RES}_k$ is denoted as the sum of all Reliability scores of $\mathrm{Agen}_k \in AC_i$.

We define $W_{fed} = (\sum_{\mathrm{Agen}_k \in AC_i} \mathrm{RES}_k \cdot W_k) / (\sum_{Agen_k \in AC_i} \mathrm{RES}_k)$, then we can get $W_{fed}' = K_i^{\top} \cdot W_{fed}$.

(3) Item choosing and recommendation. The process of recommending alone by Agen_i and federal recommendation is the same in this step, therefore we use W_{rec}' to represent both W_{fed}' and W_i' which are weight matrices in ciphertext form. According to $W_{fed}' = K_i^{\top} \cdot W_{fed}$, $W_i' = K_i^{\top} \cdot W_i$, we define W_{rec}: $W_{rec}' = K_i^{\top} \cdot W_{rec}$.

For more accurate recommendation, the Cloud server finds $\left\{\overrightarrow{B_{i,j'}}\right\}$ and computes the similarities $\left\{\mathrm{sim}(\overrightarrow{A_{i,j}}, \overrightarrow{B_{i,i'}}), i' = 1, 2, \ldots, n_i\right\}$ as follows:

$$\overrightarrow{A''_{i,j}} = W_{rec}'^{\top} \cdot L \cdot \overrightarrow{A'_{i,j}}, \tag{15}$$

$$\overrightarrow{B''_{i,i'}} = L \cdot W_{rec}'^{\top} \cdot \overrightarrow{B'_{i,j'}} (i' = 1, 2, \ldots, n_i), \tag{16}$$

$$\begin{aligned} &\mathrm{sim}\left(\overrightarrow{A_{i,j}}, \overrightarrow{B_{i,i'}}\right) = \mathrm{SOMC.Eval}\left(L, \overrightarrow{B''_{i,i'}}, \overrightarrow{A''_{i,j}}\right) \\ &(i' = 1, 2, \ldots, n_i) \\ &= \left(\overrightarrow{A''_{i,j}} - L^{\top}\overrightarrow{B''_{i,i'}}\right)^{\top} \cdot \left(\overrightarrow{A''_{i,j}} - L^{\top}\overrightarrow{B''_{i,i'}}\right) \\ &= \left(W_{\mathrm{rec}}^{\top} \cdot K_i \cdot L \cdot J_i^{\top} \cdot \overrightarrow{A_{i,j}} - L^{\top}L \cdot W_{\mathrm{rec}}^{\top} K_i \cdot K_i^{\top}\overrightarrow{B_{i,i'}}\right)^{\top} \cdot \\ &\left(W_{\mathrm{rec}}^{\top} \cdot K_i \cdot L \cdot J_i^{\top} \cdot \overrightarrow{A_{i,j}} - L^{\top}L \cdot W_{\mathrm{rec}}^{\top} K_i \cdot K_i^{\top}\overrightarrow{B_{i,i'}}\right) \\ &= \left(W_{\mathrm{rec}} \cdot \overrightarrow{A_{i,j}} - W_{\mathrm{rec}}\overrightarrow{B_{i,i'}}\right)^{\top} \cdot \left(W_{\mathrm{rec}} \cdot \overrightarrow{A_{i,j}} - W_{\mathrm{rec}}\overrightarrow{B_{i,i'}}\right) \\ &= \left(W_{\mathrm{rec}}\left(\overrightarrow{A_{i,j}} - \overrightarrow{B_{i,i'}}\right)\right)^{\top} \cdot \left(W_{\mathrm{rec}}\left(\overrightarrow{A_{i,j}} - \overrightarrow{B_{i,i'}}\right)\right) \\ &= \sum_{x=1}^{p} w_{\mathrm{rec},x}^2 \left(a_{ij,x} - b_{ii',x}\right)^2. \end{aligned} \tag{17}$$

Then according to $\left\{sim(\overrightarrow{A_{i,j}}, \overrightarrow{B_{i,i'}}), i' = 1, 2, \ldots, n_i\right\}$, the cloud selects the Item I_{opt} with the best similarity, and calculates: $\sigma_{ser} = \Lambda.\mathrm{Sign}(sk_{\Lambda,S}, Iid_{opt})$ where Iid_{opt} is the unique identifier of the I_{opt} . Then send $(I_{opt}, Iid_{opt}, \sigma_{ser})$ to Agen_i, Agen_i will forward the recommendation to the user $u_{i,j}$.

4.3.3. Update Reliability Score. After receiving the recommendation, the user $u_{i,j}$ sets scores according to the recommended items and sends it to the Cloud server as feedback, then the Cloud server updates the Agen_i's Reliability score according to the feedback score. The whole process can be divided into two parts: (1) User scores for recommendation; (2) The Cloud calculates Reliability score. The pseudo-code of the Item Recommendation is given in Algorithm 2.

(1) User scores for recommendation. First, the user $u_{i,j}$ verifies the signature by performing the algorithm Λ. $\mathrm{Verify}(pk_{\Lambda,S}, Iid_{opt}, \sigma_{ser})$. If it fails, $u_{i,j}$ refuses the request; otherwise, generates a feedback score matrix by calculating the square root of the scores: $R_{i,j} = \mathrm{diag}(\sqrt{r_{j,1}}, \sqrt{r_{j,2}}, \ldots, \sqrt{r_{j,p}})$, where $r_{j,l}$ represents to a score about l-th attribute of demand. The value of $r_{j,l}$ is 1 to 5.

To keep the Cloud server from knowing $u_{i,j}$'s scores, $u_{i,j}$ adds noise to the feedback score matrix $R_{i,j}$: We define a set

$Rsub_{i,j}$ contains $(p-t)$ scores which is selected by $u_{i,j}$ in the set $\left\{\sqrt{r_{j,x}}, x = 1, 2, \ldots, p\right\}$, where t satisfies the following condition: $t = \lceil p/3 \rceil$.

And then $u_{i,j}$ adds noise to the feedback scoring matrix $R_{i,j}$ to get $R^*_{i,j}$:

$$R^*_{i,j} = diag\left(r_{j,1}', r_{j,2}', \ldots, r_{j,p}'\right) = \begin{cases} r_{j,x}' = 0 + r_1 & \sqrt{r_{j,x}} \notin Rsub_{i,j}, \\ r_{j,x}' = \sqrt{r_{j,x}} & \sqrt{r_{j,x}} \in Rsub_{i,j}, \end{cases} \tag{18}$$

where r_1 is a random number. Then $u_{i,j}$ generates an indicator vector $\overrightarrow{R_{j,0}}$:

$$\overrightarrow{R_{j,0}} = \left(r_{0,1}, r_{0,2}, \ldots, r_{0,p}\right)^\top = \begin{cases} r_{0,x} = 0 & \sqrt{r_{j,x}} \notin Rsub_{i,j}, \\ r_{0,x} = 1 & \sqrt{r_{j,x}} \in Rsub_{i,j}. \end{cases} \tag{19}$$

Then We can get $\overrightarrow{R_{i,j(ture)}} = R^*_{i,j} \cdot \overrightarrow{R_{j,0}} = R_{i,j} \cdot \overrightarrow{R_{j,0}}$, where

$$\overrightarrow{R_{i,j(ture)}} = \left(r_{j,1}'', r_{j,2}'', \ldots, r_{j,p}''\right)^\top = \begin{cases} r_{j,x}'' = 0 & \sqrt{r_{j,x}} \notin Rsub_{i,j}, \\ r_{j,x}'' = \sqrt{r_{j,x}} & \sqrt{r_{j,x}} \in Rsub_{i,j}. \end{cases} \tag{20}$$

To prevent Agen_i tampering, $u_{i,j}$ calculates as follows:

$$\begin{aligned} R_{i,j}' &= \mathbf{SOMC.Enc}\left(J_i, R^*_{i,j}\right), \\ \overrightarrow{R_{j,0}'} &= J_i \cdot \overrightarrow{R_{j,0}}, \\ \overrightarrow{\sigma_{r.i,j}} &= \Lambda.Sign\left(sk_{\Lambda,i,j}, R_{i,j}' \cdot \overrightarrow{R_{j,0}}\right). \end{aligned} \tag{21}$$

And then $u_{i,j}$ sends $(R_{i,j}', \overrightarrow{R_{j,0}'}, \overrightarrow{\sigma_{r.i,j}})$ to Agen_i.

(2) The Cloud calculates Reliability score. After receiving a number of $(R_{i,j}', \overrightarrow{R_{j,0}'}, \overrightarrow{\sigma_{r.i,j}})$ from $u_{i,j} \in U_i$, Agen_i calculates

$$\overrightarrow{R_{j,0}''} = K_i^\top \cdot \overrightarrow{R_{j,0}'} \tag{22}$$

for each $u_{i,j} \in F_i$, where F_i is defined as a set contains users who send feedback to Agen_i. When the amount of feedback exceeds μ that is set by TA, Agen_i sends $\left\{(R_{i,j}', \overrightarrow{R_{j,0}''}, \overrightarrow{\sigma_{r.i,j}}), u_{i,j} \in F_i\right\}$ to the Cloud server.

After receiving $\left\{(R_{i,j}', \overrightarrow{R_{j,0}''}, \overrightarrow{\sigma_{r.i,j}}), u_{i,j} \in F_i\right\}$, the Cloud server calculates to get $\left\{\overrightarrow{R_{j,0}}, u_{i,j} \in F_i\right\}$:

$$\begin{aligned} \overrightarrow{R_{j,0}} &= L^\top \cdot \overrightarrow{R_{j,0}''} \\ &= L^\top \cdot K_i^\top \cdot \overrightarrow{R_{j,0}'} \\ &= J_i^\top \cdot J_i \cdot \overrightarrow{R_{j,0}}. \end{aligned} \tag{23}$$

Then the Cloud server verifies each the signature by performing the algorithm $\Lambda.\text{Verify}(pk_{\Lambda,i,j}, R_{i,j}' \cdot \overrightarrow{R_{j,0}'}, \overrightarrow{\sigma_{r.i,j}})$. If it fails, the Cloud server will punish Agen_i by giving it a low Reliability score; otherwise, calculates and updates Reliability score RES_i for each Agen_i as follows:

$$\begin{aligned} \%\ \text{score}_{u_{i,j}} &= \left(R_{i,j}' \cdot \overrightarrow{R_{j,0}}\right)^\top \cdot \left(R_{i,j}' \cdot \overrightarrow{R_{j,0}}\right) \\ &= \left(\overrightarrow{R_{j,0}}\right)^\top \cdot R^*_{i,j} J_i \cdot J_i^\top R^*_{i,j} \cdot \left(\overrightarrow{R_{j,0}}\right) \\ &= \left(\overrightarrow{R_{j,0}}\right)^\top \cdot R^*_{i,j} \cdot R^*_{i,j} \cdot \left(\overrightarrow{R_{j,0}}\right) \\ &= \left(\overrightarrow{R_{i,j}}\right)^\top \cdot \left(\overrightarrow{R_{i,j}}\right) \\ &= \sum_{\sqrt{r_{j,x}} \in Rsub_{i,j}} \left(\sqrt{r_{j,x}}\right)^2 \\ &= \sum_{\sqrt{r_{j,x}} \in Rsub_{i,j}} r_{j,x}, \end{aligned} \tag{24}$$

$$RES_{i(\text{new})} = \frac{\left(\text{count}_i \times RES_i + 1 \middle| F_i \middle| \sum_{u_{i,j} \in F_i} \text{score}_{u_{i,j}}\right)}{\text{count}_i + 1}, \tag{25}$$

$$\text{count}_{i(\text{new})} = \text{count}_i + 1, \tag{26}$$

where $|F_i|$ is defined as the number of users which send feedback to Agen_i, and $count_i$ is the number of times RES_i has been updated.

5. Security Analysis

In this section, we give the formal security proof of our proposed scheme, and security requirements of our scheme include verifiability and privacy. We have used the digital signature so that the verifiability is guaranteed, which has proven to be safe in [36]. The privacy of our proposed scheme is proved under the following threat model:

(i) The Cloud server and Agents are honest-but-curious: we assume that Agents and the Cloud server will follow the protocol, but may be curious about users' sensitive information.

(ii) Agents does not collude with the Cloud server.

For privacy, it includes users' data privacy and Agents' parameter privacy. We give the proof of users' data privacy in Theorem 1, and Agents' parameter privacy in Theorem 2.

Theorem 1. *Under the above threat model, our scheme meets the requirement of users' data privacy.*

Proof. Depending on our scheme, users' privacy is reflected in Item Recommendation and Update Reliability score. So, our proof is divided into two parts as follows:

(i) According to the Item Recommendation process in Algorithm 1, no additional information about the user (we use $u_{i,j}$ as an example) is sent except for the encrypted demand vector $\overrightarrow{A_{i,j}'}$.

If the attacker wants to reveal users' demand privacy information $\overrightarrow{A_{i,j}}$ from $\overrightarrow{A_{i,j}'} = J_i^\top \cdot \overrightarrow{A_{i,j}} + \overrightarrow{Lap_{b(i,j)}}$, he needs to

get J_i and $\overrightarrow{Lap_{b(i,j)}}$. So we can set up the following equation to solve: $\overrightarrow{A_{i,j}}' = J_i^\top \cdot \overrightarrow{A_{i,j}} + \overrightarrow{Lap_{b(i,j)}}$.

However, $J_i = K_i \cdot L$, if attacker is $Agen_i$, he has K_i, but no L. Similarly, if the attacker is the Cloud server, he has L, but no K_i. That means Agent and the Cloud server cannot get K_i and L at the same time. So, they cannot reveal J_i and $\overrightarrow{A_{i,j}}$.

Then we consider the known-sample attack [40] and *brute-force exhausted attack* [39]:

If the purpose of the attacker is to recover plains from encrypted demanding vectors, and the attacker gets a set of plain demanding vectors $\mathscr{Z} = \left\{\overrightarrow{A_{i,z_x}}, x = 1, 2, \ldots, |\mathscr{Z}|\right\}$, but he does not know $\left\{\overrightarrow{A_{i,z_x}}', x = 1, 2, \ldots, |\mathscr{Z}|\right\}$ which is the corresponding ciphertext of $\mathscr{Z}$.

The attacker can use *brute-force exhausted attack*: trying every possible J_i and $\overrightarrow{Lap_{b(i,j)}}$ to recover $\overrightarrow{A_{i,j}}$. We can simplify the problem a little bit by setting all the elements of $\overrightarrow{Lap_{b(i,j)}}$ to zero, then we can get $\overrightarrow{A_{i,j}} = J_i \cdot \overrightarrow{A_{i,j}}'$, and then attacker just needs to recover J_i.

Then we assume the worst-case scenario for $\mathscr{Z}$: there are at least p linearly independent vectors in $\mathscr{Z}$, and $|\mathscr{Z}| = n \geq p + 1$. The attacker can solve J_i by setting up the following equations: $\overrightarrow{A_{i,z_x}} = J_i \cdot \overrightarrow{A_{i,z_x}}'$ for $x = 1$ to p. If the cloud server who has L as attacker needs to try every possible $\boldsymbol{X}$ to recover J_i: $J_i = \mathbf{X} \cdot L$, then the attacker chooses p vectors from $\mathscr{Z}$ to form a $p \times p$ matrix $\mathscr{C}$ such that $\mathscr{C} = (\overrightarrow{A_{i,z_1}}, \overrightarrow{A_{i,z_2}}, \ldots, \overrightarrow{A_{i,z_p}})$. He has to try every possible linearly independent $\overrightarrow{A_{i,z_x}}'$ permutations from the encrypted demanding vectors which he has received to form a $p \times p$ matrices $\mathscr{D}$ such that $\mathscr{D} = (\overrightarrow{A_{i,*_1}}', \overrightarrow{A_{i,*_2}}', \ldots, \overrightarrow{A_{i,*_p}}')$. Then the attacker has $J_i^\top \cdot \mathscr{C} = \mathscr{D}$. Note that $\mathscr{C}$ is invertible since $\overrightarrow{A_{i,z_1}}, \overrightarrow{A_{i,z_2}}, \ldots, \overrightarrow{A_{i,z_p}}$ are linearly independent. He picks a set of p encrypted demanding vectors $\mathscr{D}$ randomly and sets up a hypothesis that $\mathscr{D}$ contains the corresponding encrypted demanding vectors in $\mathscr{C}$. Then he can set up equation $J_i^\top = \mathscr{C}^{-1} \cdot \mathscr{D}$ to solve for J_i and choose some vectors $\left\{\overrightarrow{A_*} | \overrightarrow{A_*} \in \mathscr{Z} \wedge \overrightarrow{A_*} \notin \mathscr{C}\right\}$ to verify the hypothesis: if $\overrightarrow{A_*} \notin \mathscr{C}$, the hypothesis cannot be correct; otherwise, the hypothesis may be true.

However, it is for the attacker that the attack is exponentially expensive. That's because there are $P_n^p = O(n^p)$ possible candidates of $\mathscr{D}$. For each candidate, the attacker performs to verify whether recovered key J_i is true which takes $O(n)$ validations. For example, if $n = 5K$, $p = 4$ and the attacker is capable to perform $2M$ validations in a second, then if the attacker wants to try all hypotheses, he must spend more than 460 years. Therefore, our simplified scheme ($\overrightarrow{Lap_{b(i,j)}} = 0$) can also resist this the known-sample brute-force exhausted attack. Noteworthy, in our scheme, $\overrightarrow{Lap_{b(i,j)}}$ is a random vector, which the number of non-zero is random and non-zero elements are also random. The blindness of the demanding vector is increased, which makes his calculation aboved more difficult. There's another scenario: $Agen_k$ as attacker wants to recover $u_{i,j}$'s demanding vector, but it is more difficult since he does not have K_i.

Similarly, the $Agen_i$ who has K_i as attacker needs to try every possible $\mathbf{Y}$ to recover J_i: $J_i = K_i \cdot \mathbf{Y}$, the proof process is similar to the above.

Therefore, in the Item Recommendation phase, the demanding privacy of users is guaranteed.

(ii) According to the Update Reliability score process in Algorithm 2, no additional information about $u_{i,j}$ is sent except for the encrypted feedback vector $\overrightarrow{R_{j,0}}'$ and matrix $R_{i,j}'$.

If the attaker wants to reveal users' preference information $\overrightarrow{R_{i,j(ture)}} = R_{i,j} \cdot \overrightarrow{R_{j,0}}$ and the total score of the feedback from $\overrightarrow{R_{j,0}}' = J_i \cdot \overrightarrow{R_{j,0}}$ and $R_{i,j}' = J_i^\top \cdot R_{i,j}$, where the non-zero elements in $\overrightarrow{R_{i,j(ture)}}$ represent that each true score in $R_{i,j}$, $\overrightarrow{R_{j,0}}$ is used to mark which scores in the feedback $R_{i,j}$ are true, and the sum of elements in $\overrightarrow{R_{i,j(ture)}}$ represents the total score of the feedback, he needs to get J_i, $R_{i,j}$ and $\overrightarrow{R_{j,0}}$. So we can set up the following equation to solve: $R_{i,j} = J_i \cdot R_{i,j}'$, $\overrightarrow{R_{j,0}} = J_i^\top \cdot \overrightarrow{R_{j,0}}'$.

However, $Agen_i$ who has K_i as attacker only knows the ciphertext $R_{i,j}'$, $\overrightarrow{R_{j,0}}'$ and the semi-decrypted result $\overrightarrow{R_{j,0}}'' = K_i^\top \cdot \overrightarrow{R_{j,0}}'$. He cannot know $\overrightarrow{R_{i,j(ture)}}$ and the sum of elements in $\overrightarrow{R_{i,j(ture)}}$. That's because $\overrightarrow{R_{i,j(ture)}} = R_{i,j} \cdot \overrightarrow{R_{j,0}}$, and he cannot get $\overrightarrow{R_{j,0}}$ without L. Meanwhile, the encryption method for $R_{i,j}$ is the same as the demand vector in Algorithm 1 so that he cannot recover $R_{i,j}$. Therefore, he cannot recover $\overrightarrow{R_{i,j(ture)}}$ from $\overrightarrow{R_{j,0}}'$ and $R_{i,j}'$. For the total score of the feedback, we define a p-dimensiona vector $\overrightarrow{V}$ with the elements being all 1, he can get the sum of $R_{i,j}$ by calculating $(R_{i,j}' \cdot \overrightarrow{V})^\top \cdot (R_{i,j}' \cdot \overrightarrow{V}) = \overrightarrow{V}^\top \cdot R_{i,j} \cdot R_{i,j} \cdot \overrightarrow{V}$, but the result is not the total score of the feedback, because $R_{i,j}$ contains random numbers. The total score of the feedback can only be obtained by calculating $score_{u_{i,}} \quad j = (R_{i,} \quad \overrightarrow{j} \cdot \overrightarrow{R_{j,0}} \quad)^\top \cdot (R_{i,j}' \cdot \overrightarrow{R_{j,0}}) = (J_i^\top \cdot R \quad _{i,j} \cdot \overrightarrow{R_{j,0}})^\top \cdot (J_i^\top \cdot R_{i,j} \cdot \overrightarrow{R_{j,0}}) = \overrightarrow{R_{i,j(ture)}}^\top \cdot \overrightarrow{R_{i,j(ture)}}'$, where $\overrightarrow{R_{i,j(ture)}}' = \mathbf{SOMC.Enc}(J_i, \overrightarrow{R_{i,j(ture)}})$. However, he cannot know $\overrightarrow{R_{j,0}}$ so that he cannot get the total score of the feedback. This means that he cannot know the $u_{i,j}$'s preference information and the total score of the feedback.

Similarly, the Cloud server who has L as attacker only knows the ciphertext $R_{i,j}'$ and the semi-decrypted result $\overrightarrow{R_{j,0}}''$. He cannot know $\overrightarrow{R_{i,j(ture)}}$, but can get the sum of elements in $\overrightarrow{R_{i,j(ture)}}$ to update Reliability score for $Agen_i$. This because

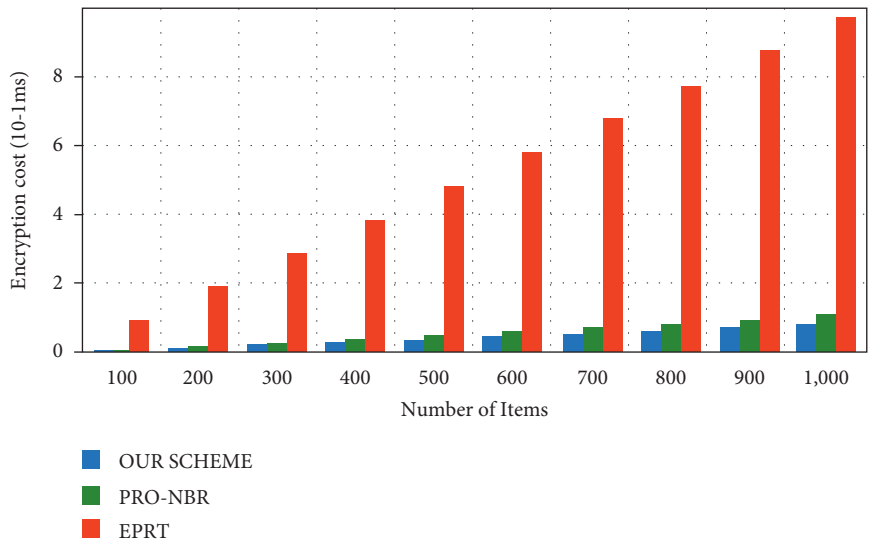

Figure 2: Comparison of the efficiency of encryption methods.

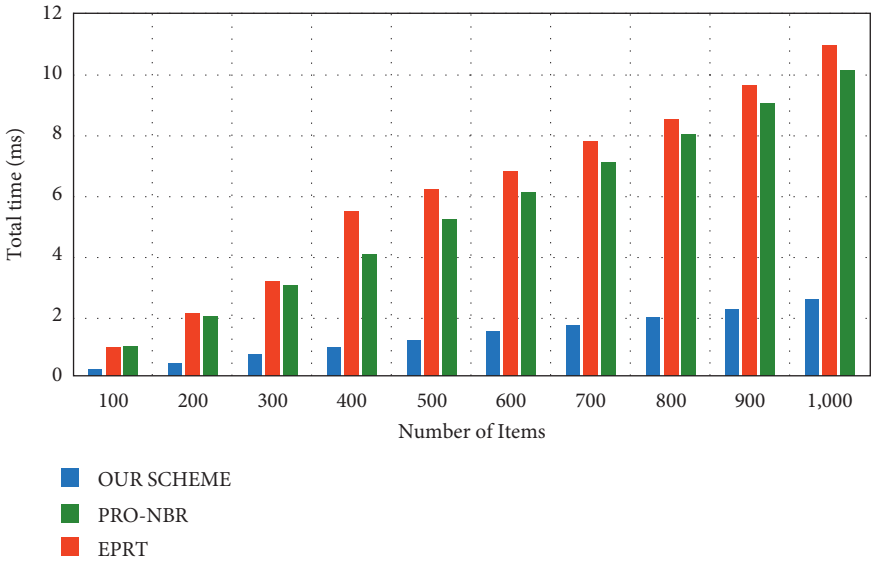

Figure 3: Comparison of the total time of recommendation phase.

$\overrightarrow{R_{i,j(ture)}} = R_{i,j} \cdot \overrightarrow{R_{j,0}}$, he can recover $\overrightarrow{R_{j,0}} = L^{\top} \cdot \overrightarrow{R''_{j,0}}$, but he cannot recover $R_{i,j}$ that is encrypted by the same method in Algorithm 1. He only get $\overrightarrow{R_{i,j(ture)}'} = R_{i,j}' \cdot \overrightarrow{R_{j,0}} = J_i^{\top} \cdot R_{i,j}^{*} \cdot \overrightarrow{R_{j,0}} = J_i^{\top} \cdot \overrightarrow{R_{i,j(ture)}} = SOMC.Enc(J_i, \overrightarrow{R_{i,j(ture)}})$. The encryption method is the same as the demand vector in Algorithm 1. Therefore, he cannot recover $\overrightarrow{R_{i,j(ture)}}$ that includes $u_{i,j}$'s preference information. But he can get the total score of the feedback: $score_{u_{i,j}} = (R_{i,j}' \cdot \overrightarrow{R_{j,0}})^{\top} \cdot (R_{i,j}' \cdot \overrightarrow{R_{j,0}}) = \overrightarrow{R_{i,j(ture)}'^{\top}} \cdot \overrightarrow{R_{i,j(ture)}'}$. This means that he can know the total score of the feedback but he cannot know the $u_{i,j}$'s preference information.

Therefore, in the Update Reliability score phase, the preference privacy of users is guaranteed.

(iii) To sum up, our scheme meets the requirement of users' data privacy. □

Theorem 2. *Under the above threat model, our scheme meets the requirement of Agents' parameter privacy.*

Proof. Depending on our scheme, Agents' parameter privacy is reflected in Item Recommendation.

According to the Item Recommendation process in Algorithm 1, no additional information about the Agent (we use $Agen_i$ as an example) is sent except for the encrypted weight matrix W_i'. We consider that the attacker is the Cloud

Input: $\overrightarrow{A_{i,j}'}, \overrightarrow{\sigma_{i,j}}, \left\{\overrightarrow{B_{i,i'}'}, i' = 1, 2, \ldots, n_i\right\}$, W_i, $\{\mathrm{RES}_i, i = 1, 2, \ldots, |U|\}$, $pk_{\Lambda,i,j}$, L, ∂ and r
Output: $(I_{opt}, Iid_{opt}, \sigma_{ser})$

(1) ***The Cloud server does:***
(2) **if** $(\Lambda.\mathrm{Verify}(pk_{\Lambda,i}, \overrightarrow{A_{i,j}'}, \overrightarrow{\sigma_{i,j}}))$ **then**
(3) **if**$(\mathrm{RES}_i > r)$**then**
(4) $W_{\mathrm{rec}}{}' = W_i'$;
(5) end
(6) else
(7) **while** *(the number of collected != j)* **do**
(8) choose j Agents which satisfy: $\sum_{z=1}^{j} \mathrm{RES}_{\max-z} \le \partial \sum_{q=1}^{N} \mathrm{RES}_q$
collecting (PID_k, W_k') from $\mathrm{Agen}_k \in AC_i$;
(9) end
(10) **for**$\mathrm{Agen}_k \in AC_i$**do**
(11) calculate $W_{\mathrm{aggr},k}'$ following equation (13);
(12) end
(13) calculate $W_{rec}' = W_{fe\,d}'$ following equation (14);
(14) end
(15) calculate $\left\{\mathrm{sim}(\overrightarrow{A_{i,j}'}, \overrightarrow{B_{i,i'}'}), i\prime = 1, 2, .., n_i\right\}$ following equation (15)–(17);
(16) choose I_{opt}: $\mathrm{sim}(\overrightarrow{A_{i,j}'}, \overrightarrow{B_{opt}'}) = \left\{\mathrm{sim}(\overrightarrow{A_{i,j}'}, \overrightarrow{B_{i,i'}'}), i\prime = 1, 2, .., n_i\right\}_{\min}$;
(17) Calculate $\sigma_{ser} \leftarrow \Lambda.\mathrm{Sign}(sk_{\Lambda,S}, Iid_{opt})$;
(18) end
(19) The Agent Agen_k does:
(20) **if** $(\mathrm{Agen}_k \in AC_i)$ **then**
(21) $W_k' \leftarrow$ **SOMC.Enc**(K_i, W_i);
$PID_k \leftarrow H(ID_k \| R_k)$;
send (PID_k, W_k') to the Cloud server
(22) end
(23) return $(I_{opt}, Iid_{opt}, \sigma_{ser})$

Algorithm 1: Item Recommendation.

server or other Agents, and he wants to recover W_i. The encryption method is the same as mentioned in Theorem 1. so that the attacker cannot recover W_i. The proof procedure is the same as Theorem 1. Here is not described again.

Thus, our scheme meets the requirement of Agents' parameter privacy. □

6. Evaluation

In this section, we conduct simulation experiments to compare the performance of the proposed SOMC and ERBFL scheme with the existing schemes. We conduct all the experiments on the server with an Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz 2.90 GHz and 16 GB of RAM and uses TensorFlow to simulate the SOMC and ERBFL algorithm in Python.

For experimental parameters, we choose appropriate and security parameters with the given initial parameter $p = 5$ and $\lambda = 128$. We can get $t = \lceil p/3 \rceil = 2$. So that the full Reliability score is 15. To facilitate the test, we choose the parameter $r = 9$, $|U| = 10$ and $|U_i| = 50$.

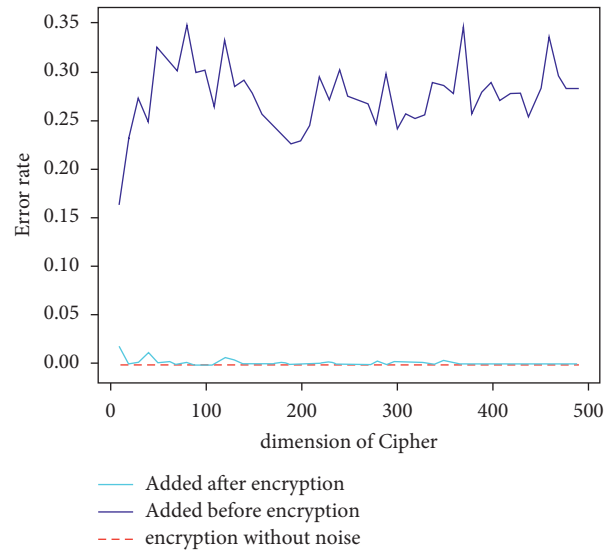

Figure 4: Influence of noise on similarity calculation.

6.1. Efficiency Comparison. In order to show the computation costs about encryption (SOMC) in ERBFL, we compare our scheme with PPO-NBR [12] and EPRT [10], which use homomorphic encryption. We compare the running time of encrypting vectors against varying number of Items. As shown in Figure 2, it can be observed that as the number of Items increasing, all of them increase. But we can see that

Input: $\left\{(R_{i,j}', \overrightarrow{R_{j,0}'}, \overrightarrow{\sigma_{r.i,j}}), u_{i,j} \in F_i\right\}$, $\{RES_i, i = 1, 2, \ldots, |U|\}$, $\{\text{count}_i, i = 1, 2, \ldots, |U|\}$, $pk_{\Lambda,i,j}$
Output: $\left\{RES_{i(new)}, i = 1, 2, \ldots, |U|\right\}$

(24) ***Each****Agen*$_i$***does***:
(25) *countnum* = 0;
for $u_{i,j} \in F_i$ **do**
(26) calculate $\overrightarrow{R_{j,0}}''$ following equation (22);
(27) *countnum* + +;
(28) **if**(*countnum* > μ)**then**
(29) send $\left\{(R_{i,j}', \overrightarrow{R_{j,0}'}, \overrightarrow{\sigma_{r.i,j}}), u_{i,j} \in F_i\right\}$ to the Cloud server;
(30) *countnum* = 0;
(31) end
(32) end
(33) *The Cloud server does*:
(34) for *eachAgen*$_i$ do
(35) **for**$u_{i,j} \in F_i$**do**
(36) calculate $\overrightarrow{R_{j,0}}$ following equation (23);
if(Λ.Verify($pk_{\Lambda,i,j}, R_{i,j}' \cdot \overrightarrow{R_{j,0}}, \overrightarrow{\sigma_{r.i,j}}$))**then**
(37) calculate $score_{u_{i,j}}$ following equation (24);
(38) end
(39) end
(40) calculate $RES_{i(\text{new})}$ following equation (25);
(41) $count_i$ + +;
(42) end
(43) return $\left\{RES_{i(\text{new})}, i = 1, 2, \ldots, |U|\right\}$

Algorithm 2: Update Reliability score.

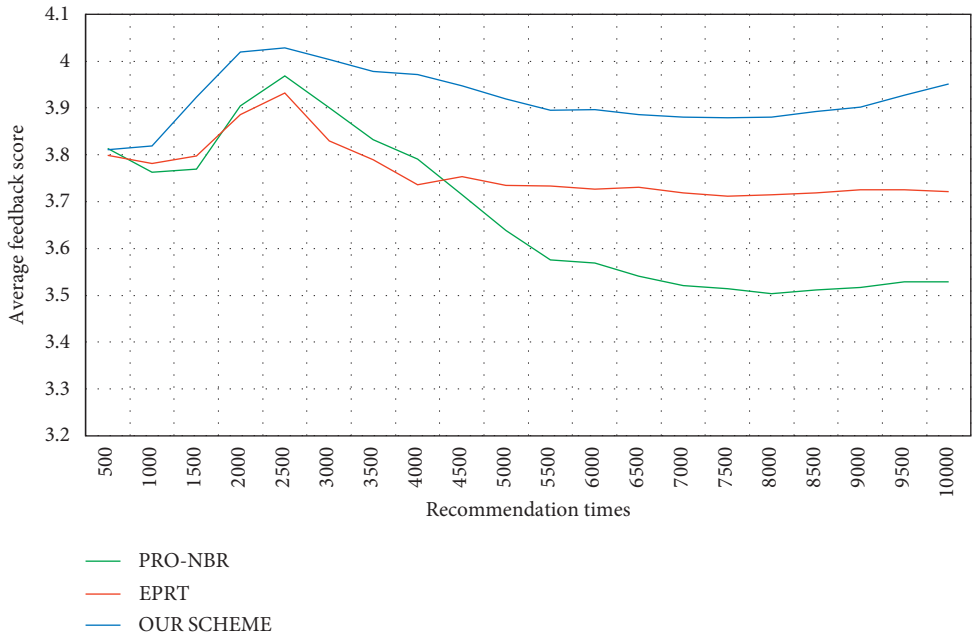

Figure 5: Comparison of Feedback score about recommendation.

our scheme takes the least time no matter how many Items. This is because the encryption form in PRO-NBR can only deal with a bit message per encryption, and EPRT can deal with an integer message, whereas our scheme can deal with p integer messages. Thus, our encryption scheme runs faster and more efficiently.

Next, we conduct experiments to compare the running time of recommendation phase in our scheme with PPO-NBR [12] and EPRT [10] as shown in Figure 3. As the number of items increases, the running time increases. However, the running time of our scheme is always the least.

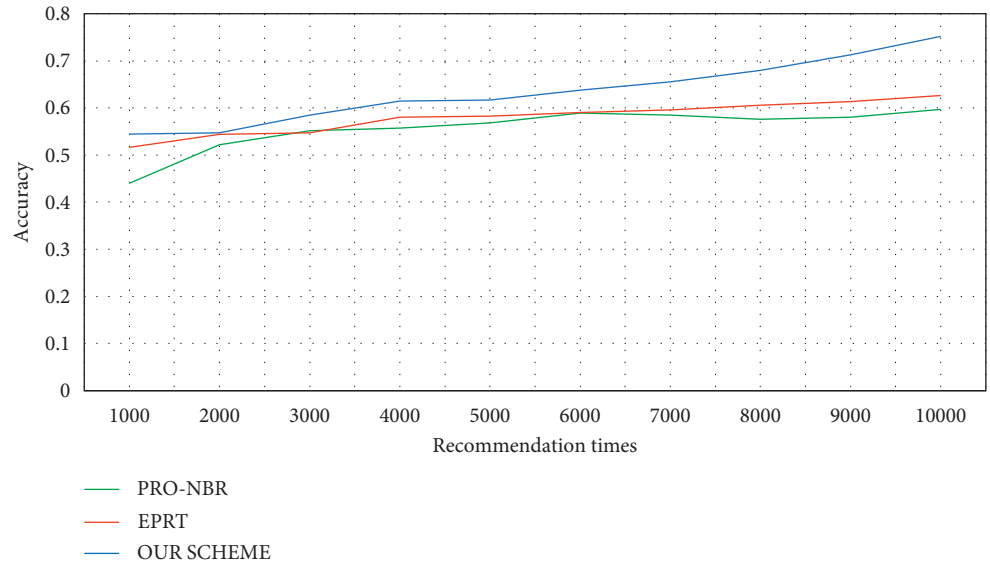

Figure 6: Comparison of Accuracy about recommendation.

6.2. Impact of Laplace Noise on Similarity Accuracy. To achieve better privacy protection, we use differential privacy Laplace noise encryption, which makes us consider the impact on the recommendation items. In fact, the accuracy of the recommended items can be known from equation (17), which is equivalent to the impact on the similarity calculation items. Therefore, we conduct the following experiments by setting three scenarios: noise added after encryption, noise added before encryption and nosieless encryption, as shown in Figure 4. It is easy to see that the effect of putting noise after encryption is almost equal to that of no noise, so the effect of our ERBFL scheme on similarity calculation is negligible, therefore, this noise will not affect the accuracy of recommendations in our scheme.

6.3. Recommendation Accuracy Comparison. We use users to score the recommended items to indicate the accuracy of the recommended items. Because if the recommendation items are more accurate, the users will be more satisfied and the feedback score will be higher. Then we use a data set MovieLens-100k [41], which includes 100000 movie ratings from about 900 users on over 1600 movies, where the value of all scores is 1 to 5. Since the data does not include all users' scores for all movies, we fill in the missing scores with the mean of two known values, where one of two known values is the average of the corresponding user has scored, another value is the average of the corresponding item has been scored. We use the data as the users' scores, the more satisfied with the recommended items, the higher the score. We repeat the recommendation process many times, find the corresponding score from the data for each time, and compare different schemes of average score. In addition, an evaluation indicator called accuracy is also used. Accuracy is defined as the fraction of correct recommendations in the total number of recommendations, and accuracy is generally considered to evaluate whether a Recommender System can recommend accurate items. Higher accuracy means that the Recommender System can recommend more comfortable items. Then, due to the scores in data set MovieLens-100k [41] are all real users scoring, we assume that an item with a score above 4 is the correct recommendation. We repeat the recommendation process many times, count the number of correct recommendations, and compare the accuracy of different schemes.

As shown in Figure 5, it can be seen that the items recommended by our scheme always has a higher feedback score. And as shown in Figure 6, it also can be seen that our scheme always has the best performance in terms of accuracy. On the one hand, our scheme is designed based on a federal learning framework, which is different from EPRT [10] and PPO-NBR [12] used centralized recommendation. On the other hand, our scheme uses similarity as a recommended criterion, which is different from PPO-NBR [12]. Therefore, our scheme can recommend the item that makes users more satisfactory.

7. Conclusion

In this paper, we propose ERBFL—an Efficient Recommendation Based on Federated Learning for helping users find an appropriate item. To protect privacy and improve accuracy of recommendation, we design a privacy-preserving ciphertext calculation for similarity calculation, and by employing the Federated Learning framework. We analyze the security of our scheme for achieving the design secure goals and ensuring the accuracy of the recommended items. In addition, the experiment proves that ERBFL is more efficient than the existing scheme. The primary constraint we faced was the high dimension of user or item information, which can affect the execution time, and the performance of our scheme. Furthermore, in our scheme, we assume that the Cloud and agents cannot collude. Therefore, our future plans are to extend our scheme to be applicable more complex scenarios and to be able to resist collusive attacks.

Data Availability

The processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grant No. 61932010. The authors appreciate the anonymous reviewers for their thorough comments and suggestions for this paper.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[2] Y. Du, D. Zhou, Y. Xie, J. Shi, and M. Gong, "Federated matrix factorization for privacy-preserving recommender systems," *Applied Soft Computing*, vol. 111, Article ID 107700, 2021.

[3] F. Zhang, V. E. Lee, and K. K. Raymond Choo, "Jo-dpmf: Differentially private matrix factorization learning through joint optimization," *Information Sciences*, vol. 467, pp. 271–281, 2018.

[4] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang, "Federated recommendation systems," in *Federated Learning*, pp. 225–239, Springer, China, 2020.

[5] G. Dhiman, S. Juneja, H. Mohafez et al., "Federated learning approach to protect healthcare data over big data scenario," *Sustainability*, vol. 14, no. 5, p. 2500, 2022.

[6] P. Zhou, K. Wang, L. Guo, S. Gong, and B. Zheng, "A privacy-preserving distributed contextual federated online learning framework with big data support in social recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, p. 1, 2019.

[7] X. Wu, B. Cheng, and J. Chen, "Collaborative filtering service recommendation based on a novel similarity computation method," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 352–365, 2017.

[8] C. Yang, X. Chen, L. Liu, T. Liu, and S. Geng, "A hybrid movie recommendation method based on social similarity and item attributes," in *International Conference on Swarm Intelligence*, pp. 275–285, Springer, Bejing, 2018.

[9] C. Huang, R. Lu, H. Zhu, J. Shao, and X. Lin, "Fssr: Fine-grained ehrs sharing via similarity-based recommendation in cloud-assisted ehealthcare system," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pp. 95–106, Bejing China, September 2016.

[10] C. Peng, D. He, J. Chen, N. Kumar, and M. K. Khan, "Eprt: An efficient privacy-preserving medical service recommendation and trust discovery scheme for ehealth system," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 3, pp. 1–24, 2021.

[11] X. Li, Y. Wang, P. Vijayakumar, D. He, N. Kumar, and J. Ma, "Blockchain-based mutual-healing group key distribution scheme in unmanned aerial vehicles ad-hoc network," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11309–11322, 2019.

[12] M. Zhang, Y. Chen, and J. Lin, "A privacy-preserving optimization of neighbourhood-based recommendation for medical-aided diagnosis and treatment," *IEEE Internet of Things Journal*.

[13] C. Xu, J. Wang, L. Zhu, C. Zhang, K. Sharif, and Ppmr, "PPMR: A Privacy-Preserving Online Medical Service Recommendation Scheme in eHealthcare System," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5665–5673, 2019.

[14] D. Li, Q. Lv, X. Xie et al., "Interest-based real-time content recommendation in online social communities," *Knowledge-based systems*, vol. 28, pp. 1–12, 2012.

[15] T. B. Ogunseyi and C. Yang, "Survey and analysis of cryptographic techniques for privacy protection in recommender systems," *International Conference on Cloud Computing and Security*, vol. 23, pp. 691–706, 2018.

[16] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020.

[17] J. Qin, B. Liu, and J. Qian, "A novel privacy-preserved recommender system framework based on federated learning," in *Proceedings of the 2021 The 4th International Conference on Software Engineering and Information Management*, pp. 82–88, China Uk, June 2021.

[18] P. Han, B. Xie, F. Yang, and R. Shen, "A scalable p2p recommender system based on distributed collaborative filtering," *Expert systems with applications*, vol. 27, no. 2, pp. 203–210, 2004.

[19] D. Rosaci, G. M. L. Sarné, and S. Garruzzo, "Muaddib: A distributed recommender system supporting device adaptivity," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 4, pp. 1–41, 2009.

[20] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE transactions on information forensics and security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[21] H. Kaur, N. Kumar, and S. Batra, "An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system," *Future Generation Computer Systems*, vol. 86, pp. 297–307, 2018.

[22] X. Ma, J. Ma, H. Li, Q. Jiang, S. Gao, and Armor, "ARMOR: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks," *Future Generation Computer Systems*, vol. 79, pp. 82–94, 2018.

[23] M. Kolahkaj, A. Harounabadi, A. Nikravanshalmani, and R. Chinipardaz, "Incorporating multidimensional information into dynamic recommendation process to cope with cold start and data sparsity problems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9535–9554, 2021.

[24] F. Casino, C. Patsakis, D. Puig, and A. Solanas, "On privacy preserving collaborative filtering: Current trends, open problems, and new issues," in *Proceedings of the 2013 IEEE 10th International Conference on e-Business Engineering*, pp. 244–249, IEEE, Coventry, UK, September 2013.

[25] M. Kolahkaj, A. Harounabadi, A. Nikravanshalmani, and R. Chinipardaz, "Dbcacf: A multidimensional method for tourist recommendation based on users' demographic, context and feedback," *Inf Syst Telecommun*, vol. 4, no. 6, pp. 209–219, 2019.

[26] D. Li, Q. Lv, L. Shang, and N. Gu, "Efficient privacy-preserving content recommendation for online social communities," *Neurocomputing*, vol. 219, pp. 440–454, 2017.

[27] J. Song, W. Wang, T. R. Gadekallu, J. Cao, and Y. Liu, "Eppda: An efficient privacy-preserving data aggregation federated learning scheme," *IEEE Transactions on Network Science and Engineering*, vol. 2022, p. 1, Article ID 3153519, 2022.

[28] G. Han, T. Zhang, Y. Zhang, G. Xu, J. Sun, and J. Cao, "Verifiable and privacy preserving federated learning without fully trusted centers," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 3, pp. 1431–1441, 2021.

[29] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2003, https://arxiv.org/abs/2003.02133.

[30] M. Song, Z. Wang, Z. Zhang et al., "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[31] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[32] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, vol. 12, pp. 148–162, 2019.

[33] J. Ahmad, M. A. Khan, S. O. Hwang, and J. S. Khan, "A compression sensing and noise-tolerant image encryption scheme based on chaotic maps and orthogonal matrices," *Neural Computing and Applications*, vol. 28, no. S1, pp. 953–967, 2017.

[34] C. Dwork, A. Roth, and L. Chen, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013.

[35] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of cryptography conference*, vol. 3, pp. 265–284, 2006.

[36] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International conference on the theory and application of cryptology and information security*, pp. 514–532, Springer, Korenio City, 2001.

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics*, vol. 12, pp. 1273–1282, 2017.

[38] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Computers & Security*, vol. 96, Article ID 101889, 2020.

[39] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 139–152, NY City, July 2009.

[40] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 297–308, Springer, Berlin, Germany, September 2006.

[41] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2016.