Privacy-preserving Hybrid Recommender System[*]

Qiang Tang and Husen Wang
Luxembourg Institute of Science and Technology
Luxembourg
{tonyrhul, wanghs.thu}@gmail.com

ABSTRACT

Privacy issues in recommender systems have attracted the attention of researchers for many years. So far, a number of solutions have been proposed. Unfortunately, most of them are far from practical as they either downgrade the utility or are very inefficient. In this paper, we aim at a more practical solution (particularly in the sense of relieving the tension between utility and privacy), by proposing a privacy-preserving hybrid recommender system which consists of an incremental matrix factorization (IMF) component and a user-based collaborative filtering (UCF) component. The IMF component provides the fundamental utility while allows the service provider to efficiently learn feature vectors in plaintext domain, and the UCF component improves the utility while allows users to carry out their computations in an offline manner. Leveraging somewhat homomorphic encryption (SWHE) schemes, we provide privacy-preserving candidate instantiations for both components. Interestingly, as a side effect of the hybrid design, individual components can enhance each other's privacy guarantees. With respect to efficiency, our experiments demonstrate that the hybrid solution is much more efficient than existing solutions.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Algorithms, Security

Keywords

Recommender System, Homomorphic Encryption, Privacy, Accuracy

1. INTRODUCTION

Recommender system predicts the preferences that users would give to an item, so that it enables users to make the most appropriate choices from the immense variety of items that are available. Today, recommender systems play an important role in every corner of our daily life. Generally speaking, two representative types of recommender systems are neighborhood-based and model-based. With a neighborhood-based recommender system, in order to predict a user Alice's rating for an item *i*, the system first chooses a neighborhood for Alice or the item *i* then computes the prediction based on data from the neighborhood. With a model-based recommender system, the system first trains a model using all available data then computes the predictions based on the model. Due to their respective (dis)advantages, recommender service providers often adopt a hybrid approach, which can combine neighborhood-based method and model-based method in different ways [6].

In real-world deployment, most existing recommender systems are centralized in the sense that a service provider will collect the inputs from all users and compute recommendations for them. The collected data range from explicit inputs such as ratings to implicit behavior data such as browsing histories and locations. This makes recommender systems very privacy invasive to individual users. For instance, Narayanan and Shmatikov [20] presented a robust de-anonymization attack against anonymized Netflix dataset. Weinsberg et al. [29] demonstrated that what has been rated by a user can already potentially help an attacker identify this user. Calandrino et al. [7] pointed out inference attacks which allow an attacker with some auxiliary information to infer a user's transactions from temporal changes in the public outputs of a recommender system. Today, privacy has become a troublesome issue for both the service provider and end users.

1.1 Related Work

In the past decade, researchers have actively investigated privacy-preserving recommender systems. Existing solutions can be categorized into two groups. The cryptographic solutions (e.g. [1, 8, 16, 22, 28]) often aim at securing the procedure of underlying recommender protocols, by using cryptographic tools such as homomorphic encryption schemes and zero-knowledge proof protocols. The data-obfuscation solutions (e.g. [2, 13, 18, 19, 23, 24]) adopt the concept of differential privacy and rely on adding noise to the original data or computation results to restrict the information leakage from recommender outputs. These two types of solutions are somehow complementary as they try to prevent information leakage from different sources.

Due to the large underlying user population, neighborhood selection and model training procedures often make the cryptographic solutions very inefficient even with effi-

[*]The work is mostly done when the authors are at University of Luxembourg.

cient cryptographic building blocks. Some cryptographic solutions assume more than one semi-trusted servers. A serious risk for these solutions is that if one server (e.g. the crypto service provider (CSP) in [16, 22]) is compromised then all users' private data may be completely exposed because the data is protected by a single key from this server. The obfuscation-based solutions usually assume a fully trusted third party, which is responsible for calibrating noises into the computation. Technically, this third party can be replaced by a secure multiparty computation protocol. But, the cost can be too high to be practical. Moreover, even though differential privacy is a mathematically rigorous concept, it has many limitations in practical usage, e.g. [10, 14].

It is worth noting that there exist relatively more efficient cryptographic solutions, which are based on additional setup assumptions. For instance, the authors from [15, 26] proposed the concept of friendship-based recommender systems, which leverage the background social network information among users. Unfortunately, these solutions do not generalize to broader settings. Moreover, friendship information may be sensitive in the first place, so that these solutions have their inherent privacy drawbacks.

To sum up, it remains as an interesting problem to construct a pragmatic privacy-preserving recommender system, which preserves reasonable utility and is efficient enough in practice.

1.2 Our Contribution

In order to obtain a privacy-preserving solution for data-intensive machine learning algorithms, a very important direction is to reduce the size of dataset involved in the computation. For instance, Shokri and Shmatikov proposed privacy-preserving deep learning solutions by training models with sub-datasets first and then aggregate the results [25].

In this paper, we adopt the above philosophy in an attempt to investigate practical privacy-preserving recommender systems. To this end, we propose a hybrid recommender system, which accommodates both neighborhood-based and model-based components. The final predictions for a user Alice is a combination of the predictions from the individual components, as depicted in Figure 1. This corresponds to the system architecture shown in Figure 2 from Section 3. To facilitate

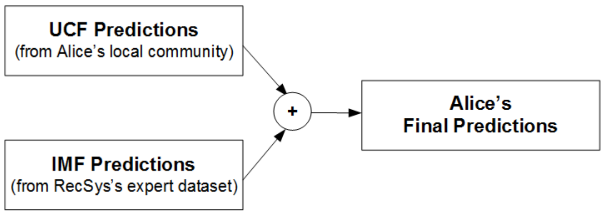
UCF Predictions
(from Alice's local community)
+
Alice's
Final Predictions
IMF Predictions
(from RecSys's expert dataset)

Figure 1: Hybrid Predictions

privacy protection, we instantiate the neighborhood-based component via user-based collaborative filtering (UCF), and instantiate the model-based component via incremental matrix factorization (IMF). The IMF allows RecSys to perform matrix factorization based on expert rating dataset, and a user Alice can perform a privacy-preserving IMF protocol to learn her feature vector. With our simulated environment, we show that the hybrid system provides more accurate recommendation results than the individual components. In contrast, many existing solutions sacrifice the recommendation accuracy by using tailored recommendation techniques (e.g. adding random noises in differential privacy solutions or simplifying existing algorithms).

Relying on somewhat homomorphic encryption (SWHE) schemes, we propose privacy-preserving protocols for both components.

- We propose a privacy-preserving UCF protocol by introducing a third party, named Proxy. Different from most existing works, it is not necessary to require the Proxy to be semi-honest.

- We propose two privacy-preserving IMF protocols. The server-centric protocol pushes most of the computation workload to the server, and the balanced protocol is more computation balanced for the participants.

We evaluate the privacy properties of the hybrid system with respect to a number of attack scenarios, ranging from pure semi-honest adversaries to fully colluded ones.

As the privacy-preserving UCF protocol will be less frequently executed than the privacy-preserving IMF protocol in the hybrid solution, in this paper we mainly study the performances of the IMF protocols, based on Microsoft SEAL library [17]. The results show that the server-centric protocol is extremely efficient for the end users, while it is also very efficient for RecSys in comparison to other existing solutions. The balanced IMF protocol is less efficient for both participants because of its stronger privacy guarantees. Due to the complexity of parameter selection, we only give some brief performance figures for the balanced IMF protocol and leave the detailed evaluation as a future work.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we present preliminaries on notation and building blocks. In Section 3, we present the hybrid system and its evaluation results. In Section 4, we give a high-level description of our privacy-preserving hybrid system. In Section 5, we describe the privacy-preserving UCF protocol. In Section 6, we describe two privacy-preserving IMF protocols. In Section 7, we provide security and efficiency analysis for our solution. In Section 8, we conclude the paper.

2. PRELIMINARY

Suppose the item set is denoted by $\mathbf{I} = (1, \cdots, M)$. For a user x, its ratings are denoted by $\mathbf{R}_x = (r_{x,1}, \cdots, r_{x,M})$. The rating value is often an integer from $\{0, 1, 2, 3, 4, 5\}$. If item i has not been rated, then $r_{x,i}$ is set to be 0. User x's average rating is denoted by $\overline{r_y}$. We use bold letters such as $\mathbf{u}, \mathbf{v}$ to denote vectors, and use $\langle \mathbf{u}, \mathbf{v} \rangle$ to denote the inner product.

Many metrics can be used to measure the recommendation quality of a recommender protocol. In this paper, we use Root Mean Square Error (RMSE), defined as follows.

$$RMSE = \sqrt{\frac{1}{|\Gamma|} \sum_{\hat{r}_{u,i} \in \Gamma} (\hat{r}_{u,i} - r_{u,i})^2},$$

where Γ is the set of predicted ratings, $\hat{r}_{u,i}$ is the predicted rating and $r_{u,i}$ is the real rating value. Note that lower RMSE implies more accurate recommendations/predictions.

2.1 User-based Collaborative Filtering

For two rating vectors $\mathbf{R}_x, \mathbf{R}_y$, their Cosine similarity is denoted as $Sim_{x,y}$, where

$$\begin{aligned} Sim_{x,y} &= \frac{\langle \mathbf{R}_x, \mathbf{R}_y \rangle}{||\mathbf{R}_x|| \times ||\mathbf{R}_y||} \\ &= \langle \frac{\mathbf{R}_x}{||\mathbf{R}_x||}, \frac{\mathbf{R}_y}{||\mathbf{R}_y||} \rangle \end{aligned}$$

Suppose we want to compute predictions for a user x based on the rating data from a user group $\mathcal{G}$, then the formula is as follows.

$$p_{x,i} = \overline{r_x} + \frac{\sum_{y \in \mathcal{G} \wedge r_{y,i} \neq 0} Sim_{x,y}(r_{y,i} - \overline{r_y})}{\sum_{y \in \mathcal{G} \wedge r_{y,i} \neq 0} Sim_{x,y}}$$

We select UCF as a building block for our hybrid system, because it provides better accuracy than other algorithms in case of a small user population.

2.2 Matrix Factorization

Given a user set $\mathcal{U} = \{1, 2, \cdots, N\}$ and their rating vectors $\mathbf{R}_x$ for $x \in \mathcal{U}$, let $\mathcal{R}$ denote the set of (x, j) such that $r_{x,j} \neq 0$. One of the most popular collaborative filtering algorithms is based on low-dimensional factor models, which derive two feature matrices $\mathbf{U}$ and $\mathbf{V}$ from the rating dataset. The feature vector $\mathbf{u}_x$ denotes user x's interest and the feature vector $\mathbf{v}_j$ denotes item j's characteristics. Every feature vector has very low dimension k, which is often a much smaller integer than M and N.

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} \quad \text{and} \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_M \end{bmatrix}$$

In practice, $\mathbf{U}$ and $\mathbf{V}$ are often computed by minimizing the following Regularized least Squares Error (RSE) function:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{|\mathcal{R}|} \sum_{(x,j) \in \mathcal{R}} (r_{x,j} - \langle \mathbf{u}_x, \mathbf{v}_j \rangle)^2 + \lambda \sum_{x \in \mathcal{U}} ||\mathbf{u}_x||_2^2 + \mu \sum_{j \in \mathbf{I}} ||\mathbf{v}_j||_2^2 \quad (1)$$

for some positive parameters λ, μ. Using the standard gradient descent method, $\mathbf{U}$ and $\mathbf{V}$ can be learned through recursively applying the updating rules. Every round of updating is called an *epoch*, and the total number of epochs is denoted as MAX_{epoch}.

$$\mathbf{u}_x^{(t)} = \mathbf{u}_x^{(t-1)} - \gamma \nabla_{\mathbf{u}_x} F(\mathbf{U}^{(t-1)}, \mathbf{V}^{(t-1)}) \quad (2)$$

$$\mathbf{v}_j^{(t)} = \mathbf{v}_j^{(t-1)} - \gamma \nabla_{\mathbf{v}_j} F(\mathbf{U}^{(t-1)}, \mathbf{V}^{(t-1)}) \quad (3)$$

where $\gamma > 0$ is a small gain factor and

$$\nabla_{\mathbf{u}_x} F(\mathbf{U}, \mathbf{V}) = -2 \sum_{j:(x,j) \in \mathcal{R}} \mathbf{v}_j (r_{x,j} - \langle \mathbf{u}_x, \mathbf{v}_j \rangle) + 2\lambda \mathbf{u}_x \quad (4)$$

$$\nabla_{\mathbf{v}_j} F(\mathbf{U}, \mathbf{V}) = -2 \sum_{x:(x,j) \in \mathcal{R}} \mathbf{u}_x (r_{x,j} - \langle \mathbf{u}_x, \mathbf{v}_j \rangle) + 2\mu \mathbf{v}_j \quad (5)$$

2.3 Somewhat Homomorphic Encryption

Since the breakthrough work of Gentry [11], many somewhat homomorphic encryption (SWHE) schemes have been proposed (e.g. the BV scheme [5], BGV scheme [4], YASHE scheme [3]). A SWHE scheme can be described by three algorithms (Keygen, Enc, Dec).

- Keygen(λ, L): With the security level λ, the multiplication depth L, this algorithm outputs a public/private/evaluation key pair (PK, SK, EVK).
- Enc(PK, m): this algorithm outputs a ciphertext c.
- Dec(SK, c): this algorithm outputs a plaintext m or an error $\perp$.

Throughout the paper, given a key pair (PK_u, SK_u) for some user u, we use $[m]_u$ to denote a ciphertext of the message m under public key PK_u. When $\mathbf{m}$ is a vector of messages, we use Enc$(PK_u, \mathbf{m})$ to denote the vector of ciphertexts, where encryption is done for each element independently. Given two ciphertexts $[m_1]_u$ and $[m_2]_u$, we have the following notations.

- We use $[m_1]_u \oplus [m_2]_u$ and $[m_1]_u \otimes [m_2]_u$ to denote the homomorphic addition and multiplication respectively.
- We use the notation $\sum_{1 \leq i \leq N} [m_i]_u$ to denote the result of sequentially applying $\oplus$ to the cipheretxts.
- We use **ReRand** to denote a rerandomization algorithm. Given a ciphertext $[m]_u$, no attacker can tell whether **ReRand**$([m]_u)$ and $[m]_u$ encrypt the same plaintext or not. Generically, his operation is equivalent to adding with one encrypted zero.
- With respect to $\oplus$, we use the notation $\ominus$ to denote the homomorphic sunstraction operator. It is clear that we can implement $\ominus$ based on $\oplus$.
- We also use $\oplus$, $\otimes$, and $\ominus$ to denote partial homomorphic operations, in which case one of the arguments is in plaintext form. The distinction should be clear from the contexts. Unlike $\otimes$, the partial $\otimes$ has less noise increase and doesn't need the costly relinearization, which makes it much faster.

2.4 Approximate Integer Division

Assume a two-party setting, where Alice holds a SWHE public/private key pair (PK_a, SK_a) and the RecSys holds two ciphertexts $[X]_a$ and $[Y]_a$ satisfying $X < 2^{L+1}Y$ for some integer L. We design an algorithm for the RecSys to compute $[\lfloor \frac{X}{Y} \rfloor]_a$. The algorithm is based on a secure integer comparison protocol COM [27]: given inputs $([A]_a, [B]_a; SK_a)$ from RecSys and Alice respectively, COM outputs $([b]_a, \emptyset)$ to the participants respectively. Note that $b = 1$ if $A \geq B$ and $b = 0$ otherwise.

Algorithm 1: Approximate Integer Division

```
1 $t = L$
2 while $t \geq 0$ do
3     $([b_t]_a; \emptyset) = COM([X]_a, [2^t Y]_a; SK_a)$
4     $[X]_a = [X]_a \ominus (2^t \otimes [b_t]_a \otimes [Y]_a)$
5     $t = t - 1$
6 $[\lfloor \frac{X}{Y} \rfloor]_a = \sum_{0 \leq t \leq L} 2^t \otimes [b_t]_a$
```

The complexity of this protocol is breifly summarized in Table 1. Note that we use par. $\otimes$ to denote partial $\otimes$.

	$\oplus$	$\ominus$	$\otimes$	par. $\otimes$	COM
Alice	0	0	0	0	L+1
Server	L	1	L+1	2L+1	L+1

Table 1: Division Complexity

3. HYBRID RECOMMENDER SYSTEM

In this section, we first describe our hybrid recommender system and also the motivation behind it. Then, we study its performance in terms of recommendation accuracy.

3.1 Hybrid System Design

There are two types of entities in the proposed recommender systems: recommender service provider (RecSys) and users. The overall system structure is shown in Figure 2.

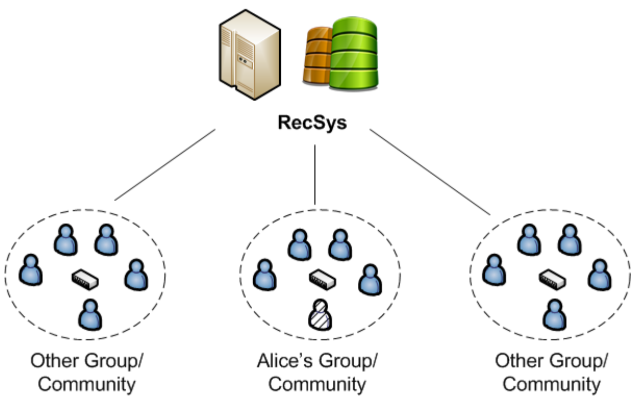

Figure 2: Hybrid System Framework

The RecSys is a commercial organization who collects item ratings from professionals and build models to help the users find their best recommendations. As a return, the RecSys may ask for some monetary reward for the service. The users are customers of the Recsys, and they want to receive good recommendations without sacrificing their privacy (i.e. disclosing their rating vectors to the RecSys or others). We further assume the users are clustered into groups, depending on demographic information such as locations. For each user group, we assume the existence of a third party named Proxy, which will facilitate the protocol execution. In practice, if RecSys represents an international company then a Proxy can be regarded as a branch or a broker in an individual country. This system structure aligns well with the practice in recommendation industry.

To compute recommendations for a user Alice of a certain group, we will take into account both local influences and global influences. The local influence comes from the users from Alice's group, and we employ the UCF to benefit from the similarities between users in a group. For each group, the execution of UCF is coordinated by the Proxy. The global influence comes from the expert dataset from the RecSys, and we employ IMF to learn item features from the dataset. Note that the expert dataset is from professional critics who may make a living for their opinions. In addition, they usually disclose their dataset to RecSys for some monetary reward. Finally, the local and global influences are combined to provide hybrid recommendations. In more detail, for a user Alice, her recommendations are computed in three steps.

1. Suppose the user group that Alice is involved in is denoted as $\mathcal{G}$. Then we compute prediction $p_{a,i}^{(ucf)}$ based on UCF as shown in Section 2.1.

2. For Alice, let the indices of the unzero ratings (in the form (a, j)) be denoted by $\mathcal{R}_a$. We employ an incremental matrix factorization (IMF) algorithm (i.e. Algorithm 2) to learn Alice's feature vector $\mathbf{u}_a$. Then, the prediction $p_{a,i}^{(imf)}$ can be computed as an inner product $\langle \mathbf{u}_a, \mathbf{v}_i \rangle$.

3. With both $p_{a,i}^{(imf)}$ and $p_{a,i}^{(ucf)}$, we can compute a hybrid predication score $p_{a,i}$ with a parameter α, which denotes the percentage that IMF contributes to the final prediction.

$$p_{a,i} = \alpha \cdot p_{a,i}^{(imf)} + (1-\alpha) \cdot p_{a,i}^{(ucf)} \tag{6}$$

This can be regarded as the simplest way to generate hybrid predictions. We can adapt it for better security, as shown in Section 7.1.

Algorithm 2: Incremental Matrix Factorization

```
1  INPUT: $\mathbf{R}_a$, $\mathbf{U}$ and $\mathbf{V}$
2  Initialize $\mathbf{u}_a = \mathbf{u}_a^{(0)}$
3  $\mathbf{U}^{(0)} = \binom{\mathbf{U}}{\mathbf{u}_a}$, $\mathbf{V}^{(0)} = \mathbf{V}$, $\mathcal{R} = \mathcal{R} \cup \mathcal{R}_a$
4  t=1
5  while $t \le MAX_{epoch}$ do
6      $\mathbf{u}_a^{(t)} = \mathbf{u}_a^{(t-1)} - \gamma \nabla_{\mathbf{u}_a} F(\mathbf{U}^{(t-1)}, \mathbf{V}^{(t-1)})$
7      $\mathbf{U}^{(t)} = \binom{\mathbf{U}}{\mathbf{u}_a^{(t)}}$
8      $\mathbf{V}^{(t)} = \mathbf{V}$
9      $t = t + 1$
10 OUTPUT: $\mathbf{u}_a = \mathbf{u}_a^{(MAX_{epoch})}$
```

3.2 System Performance

Since we do not have a real-world dataset at hand, we investigate the performance (i.e. prediction accuracy) in a carefully simulated environment. The simulation is based on the Movielens 1M dataset [12], which has 6040 users and 3952 items. The source codes for the experiments are in Github[1].

First, the expert dataset and user groups are generated as follows.

1. We first rank the rating vectors based on their hamming weight (i.e. how many items have been rated) and take the top 4000 vectors as the expert rating dataset. This is a reasonable choice since experts usually rate more items than individual users in reality.

2. With the left 2040 rating vectors, we run K-means clustering algorithms to divide them into 10 groups. The distance metric between two vectors is measured by their Euclidean distance. Each group contains around 200 users, and some statistical information is shown in Table 2.

[1]https://github.com/ClaudioCimarelli/OnlineKMF-RS

	Group 1	Group 2	Group 3	Group 4	Group 5
UserNum	150	147	230	159	161
Density %	1.0093	0.9159	0.9692	0.9566	0.9164

	Group 6	Group 7	Group 8	Group 9	Group 10
UserNum	153	238	174	401	227
Density %	0.9723	0.8905	0.8334	0.7911	0.8982

Table 2: Information of Groups

Then, we compute the RMSE for three settings. One is UCF setting, where a user Alice's predictions are solely generated by the UCF algorithm based on data from Alice's group. The other is IMF setting, where a user Alice's predictions are solely from the IMF algorithm. The results are shown in Table 3 and 4 respectively.

Group 1	Group 2	Group 3	Group 4	Group 5
0.9539	1.0712	1.0651	1.0367	1.1249
Group 6	Group 7	Group 8	Group 9	Group 10
0.9891	1.1133	1.1376	1.1585	0.9857

Table 3: RMSE in UCF Setting

Group 1	Group 2	Group 3	Group 4	Group 5
0.8372	0.9358	0.9418	0.9413	0.9770
Group 6	Group 7	Group 8	Group 9	Group 10
0.9091	0.9733	0.9907	1.0224	0.8898

Table 4: RMSE in IMF Setting

The last one is the hybrid setting, where Alice's predictions are generated by combining the predictions in the previous two settings, as shown in Equation (6). With the factor α as a variable, we plot the RMSE for the hybrid system in Figure 3. It is clear that IMF provides better recommender accuracy than UCF. However, the hybrid system provide the best recommendation accuracy when IMF contributes approximate 80% to the final predictions.

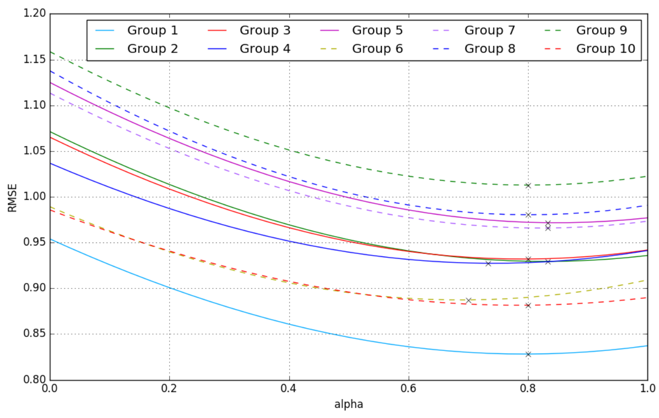

Figure 3: RMSE of Hybrid System

4. PRIVACY-PRESERVING SYSTEM

Referring to the hybrid system description in Section 3.1, we can construct a privacy-preserving hybrid version based on privacy-preserving UCF and IMF protocols. We leave the description of these protocols to Section 5 and 6, and focus on a general description for the hybrid system here. We perform detailed analysis in Section 7.

4.1 Setup Phase and Assumptions

We assume RecSys possesses a SWHE key pair (PK_s, SK_s). Every user, say Alice, registers at RecSys and the Proxy of her group with an identifier ID_a and SWHE key pair (PK_a, SK_a).

We assume that the communication among users is integrity protected, and the communication between any user and the Proxy is through an anonymous network and confidentiality protected. Therefore, it is difficult for the Proxy to identify users purely from the data sources.

4.2 Privacy-preserving System

Suppose a user Alice wants to compute the predictions for all the items, then the protocol is as follows.

1. *UCF contribution.* If necessary, Alice initiates the privacy-preserving UCF protocol from Section 5. At the end of the protocol, the Proxy in Alice's group sends $[p^{(ucf)}_{a,i}]_a$ for every item $1 \le i \le M$ to RecSys. We remark on this below.

2. *IMF contribution.* Alice initiates one of the privacy-preserving IMF protocols in Section 6. At the end of the protocol, RecSys obtains $[p^{(imf)}_{a,i}]_a$ for every $1 \le i \le M$.

3. *Hybrid contribution.* RecSys chooses a contribution factor α and computes the hybrid predictions $[p_{a,i}]_a$ $(1 \le i \le M)$.
$$[p_{a,i}]_a = \alpha \otimes [p^{(imf)}_{a,i}]_a \oplus (1-\alpha) \otimes [p^{(ucf)}_{a,i}]_a$$

4. After receiving $[p_{a,i}]_a$ $(1 \le i \le M)$, Alice can decrypt them and obtain the plaintext predictions.

Remark. It is common that individual users do not update their ratings frequently. In fact, adding a few ratings values does not affect the final predictions (or improve the recommendation accuracy). This means that, it is unnecessary to frequently execute the privacy-preserving UCF protocol in the hybrid system. As such, RecSys can store $[p^{(imf)}_{a,i}]_a$ for every $1 \le i \le M$ at the end of the first step, and use them for a pre-defined period. In contrast, RecSys, as the professional service provider, is incentivized to incorporate more expert data to learn better feature vectors for the end users. We demonstrate the effect by plotting the RMSE changes in Figure 4. This will in the end result in better satisfaction and increase its revenue. As such, the privacy-preserving IMF protocol should be executed every time Alice wants to retrieve new predictions. When evaluating the computational complexity of the hybrid system, we should put more emphasis on step 2. Note also that step 1 (i.e. the privacy-preserving UCF protocol) does not need to involve RecSys, Alice's group can carry out the computation in an offline manner. Therefore, even if the participants have high workload in step 1, it will not affect the practicality of the solution.

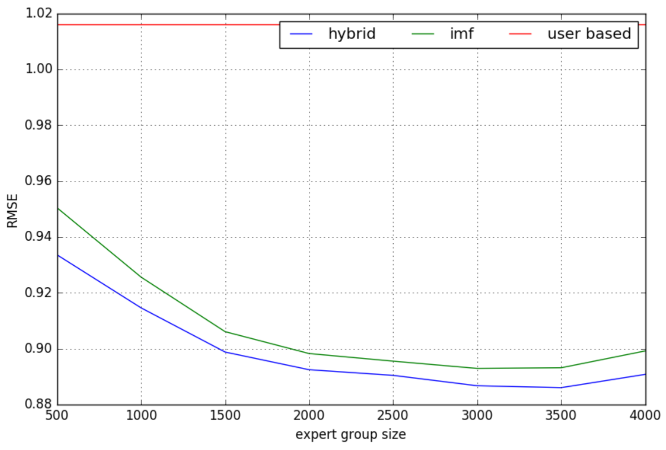

Figure 4: RMSE Influence w.r.t. Experts' Set Size

4.3 Security Model(s)

If we assume RecSys and the Proxy are semi-honest, then the security model will be essentially the same as the IND-CPA security for encryption. Consequently, the IND-CPA security of the underlying encryption scheme will provide the necessary guarantee. We, as in the case of [16, 22], avoid a straightforward description for the model. If the RecSys and the Proxy are compromised or malicious, then things become very complex and tedious to describe. Our setting is almost identical to that in [27], so that we refer the reader to Section III of [27] for detailed description of the security models.

Note that we do not consider Denial of Service (DoS) and robustness attacks in this work. Nevertheless, these attacks pose serious threat to recommender systems in practice. We leave a formal investigation of these attacks as a future work.

5. PRIVACY-PRESERVING UCF

The overall structure is shown in Figure 5 where the numbers correspond to the steps below. Suppose the users in

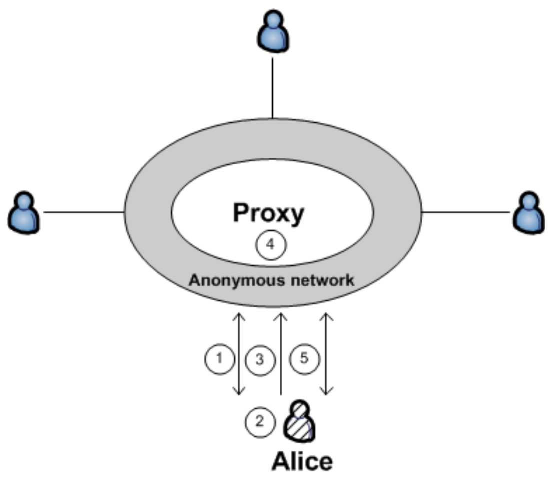

Figure 5: User-based CF

Alice's group consists of user i for $1 \leq i \leq T$. Referring to the setup information in Section 4.1, the protocol runs as follows. For the sake of simplicity, in the first three steps, we only describe the computations for Alice while implicitly assuming all other users in the group will perform similar computations.

1. Alice broadcasts her identifier ID_a together with the encrypted vectors $[\frac{\mathbf{R}_a}{||\mathbf{R}_a||}]_a$ and $[(r_{a,1} - \overline{r_a}, \cdots, r_{a,M} - \overline{r_a})]_a$. Note that $\frac{\mathbf{R}_a}{||\mathbf{R}_a||}$ is the normalized rating vector.

2. After receiving the broadcasted messages from other users, for every user $1 \leq y \leq T$, Alice computes encrypted similarity $[Sim_{a,y}]_y$, where
$$[Sim_{a,y}]_y = \langle \frac{\mathbf{R}_a}{||\mathbf{R}_a||}, [\frac{\mathbf{R}_y}{||\mathbf{R}_y||}]_y \rangle$$

3. For every item i $(1 \leq i \leq M)$, Alice does the following.
   - When $r_{a,i} \neq 0$, Alice computes her contribution $Con_{a \xrightarrow{i} y}$ to every user $1 \leq y \leq T$.
$$NOM_{a \xrightarrow{i} y} = \mathsf{ReRand}([Sim_{a,y}]_y \otimes (r_{a,i} - \overline{r_a}))$$
$$DEN_{a \xrightarrow{i} y} = \mathsf{ReRand}([Sim_{a,y}]_y)$$
$$Con_{a \xrightarrow{i} y} = (ID_y,\ i,\ NOM_{a \xrightarrow{i} y},\ DEN_{a \xrightarrow{i} y})$$
   - When $r_{a,i} = 0$, Alice sets $Con_{a \xrightarrow{i} y}$ for every user $1 \leq y \leq T$, as follows.
$$Con_{a \xrightarrow{i} y} = (ID_y, i, NOM_{a \xrightarrow{i} y} = [0]_y, DEN_{a \xrightarrow{i} y} = [0]_y)$$

   Alice sends the computed values to the Proxy. In addition, Alice sends ID_a, $[\overline{r_a}]_a$ as well.

4. After receiving the values from Alice and all other users, the Proxy computes the encrypted prediction for every user and item *in a partial form*. In total, there are $M \cdot (T+1)$ partial predictions.

   For instance, the partial prediction for Alice with respect to item i is computed as follows.
$$([\overline{r_a}]_a, \sum_{1 \leq x \leq T} NOM_{x \xrightarrow{i} a}, \sum_{1 \leq x \leq T} DEN_{x \xrightarrow{i} a})$$

5. The Proxy runs the approximate division protocol from Section 2.4 to compute the complete predictions for Alice and all other users. In total, there are $M \cdot (T+1)$ predictions.

   For instance, for Alice with respect to the item i, the Proxy runs Algorithm 1 from Section 2.4 to compute
$$\lfloor \frac{\sum_{1 \leq x \leq T} NOM_{x \xrightarrow{i} a}}{\sum_{1 \leq x \leq T} DEN_{x \xrightarrow{i} a}} \rfloor,$$
   which abuses the notation to denote a ciphertext of the approximated division result. Then, the final encrypted prediction $[p^{(ucf)}_{a,i}]_a$ is simply
$$[p^{(ucf)}_{a,i}]_a = [\overline{r_a}]_a \oplus \lfloor \frac{\sum_{1 \leq x \leq T} NOM_{x \xrightarrow{i} a}}{\sum_{1 \leq x \leq T} DEN_{x \xrightarrow{i} a}} \rfloor$$

   The Proxy sends all these encrypted predictions to RecSys.

Complexity. Let $\ell = M\rho$ be the total number of non-zero ratings, where ρ is the rating density, and let AID denote the complexity numbers from Table 1. The complexity of this protocol is briefly summarized in Table 5. We note that the Enc operations for Alice can be done offline.

	par. $\otimes$	$\oplus$	Enc
Alice	$(\rho+1)MT$	$T(M-1)$	$2(1-\rho)MT - 2M + 1$
Proxy	0	$M(2T^2+T-1)$	0
	AID	ReRand	
Alice	$M(T+1)$	$2\rho MT$	
Proxy	$M(T+1)$	0	

Table 5: UCF Complexity

Security. With respect to the security for Alice, we consider two scenarios.

- Suppose that the Proxy is a semi-honest player, then it learns no information about the users' rating values since everything is encrypted. No user learns anything about another user due to the encryption. Referring to Step 3, the anonymous communication network prevents the Proxy from learning who has sent the contribution. The application of ReRand is essential to hide which items Alice has rated. Without this, the Proxy might be able to use the technique from [29] to re-identify Alice in some attack scenarios.

- Suppose the Proxy is malicious (or compromised) and colludes with some of the users in the group, then it is still difficult for it to recover and link Alice's rating values. The anonymous communication network makes it hard for the Proxy alone to link Alice's contributions, say $Con_{a \xrightarrow{i} y}$, to her identifier ID_a. If the Proxy colludes with user y, it will be able to decrypt $Con_{a \xrightarrow{i} y}$ for all $1 \le i \le M$. However, it is not immediate that the Proxy can conclude that these $Con_{a \xrightarrow{i} y}$ values belong to the same user Alice. As long as there are other honest users who have rated similar items to Alice, then the Proxy still needs to try all combinations to determine whether two contribution values belong to the same honest user. In the extreme situation that all users collude with the Proxy, then they can learn learn which items have been rated by Alice and the rating deviations $r_{a,i} - \overline{r_a}$. Subsequently, they may succeed in the re-identifying Alice. We know that unless calibrating noise into the computation, there is nothing more we can do when all participants collude. In the context of our hybrid solution, we present more analysis in Section 7.1.

6. PRIVACY-PRESERVING IMF

In this section, we present two privacy-preserving IMF protocols of different flavors. One protocol is server-centric, meaning that most computations are pushed to the RecSys's side. The other protocol is balanced, meaning that the computations are shared by both parties.

6.1 Server-centric IMF Protocol

We present a new protocol based on the following observation. For two column vectors $\mathbf{X}, \mathbf{Y}$, we have $\mathbf{X}\mathbf{Y}^t\mathbf{X} = \mathbf{X}\mathbf{X}^t\mathbf{Y}$.

$$\begin{aligned}
\mathbf{X}\mathbf{Y}^t\mathbf{X} &= \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \times \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\
&= \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \times \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \times \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \\
&= \mathbf{X}\mathbf{X}^t\mathbf{Y}
\end{aligned}$$

In the following , for Alice, if the rating $r_{a,j} = 0$ we set $d_{a,j} = 0$, otherwise we set $d_{a,j} = 1$. The updating rule can be rewritten as follows.

$$\begin{aligned}
\mathbf{u}_a^{(t)} &= (1-2\gamma\lambda)\mathbf{u}_a^{(t-1)} + 2\gamma \sum_{j:(a,j)\in\mathcal{R}} \mathbf{v}_j(r_{a,j} - \langle \mathbf{u}_a^{(t-1)}, \mathbf{v}_j\rangle) \\
&= (1-2\gamma\lambda)\mathbf{u}_a^{(t-1)} + 2\gamma(\sum_{1\le j\le M} \mathbf{v}_j r_{a,j} - \sum_{1\le j\le M} d_{a,j}\mathbf{v}_j\langle \mathbf{u}_a^{(t-1)}, \mathbf{v}_j\rangle) \\
&= (1-2\gamma\lambda)\mathbf{u}_a^{(t-1)} + 2\gamma(\sum_{1\le j\le M} \mathbf{v}_j r_{a,j} - (\sum_{1\le j\le M} d_{a,j}\mathbf{v}_j\mathbf{v}_j^t)\mathbf{u}_a^{(t-1)}) \\
&= ((1-2\gamma\lambda)\mathbf{I}_k + 2\gamma(\sum_{1\le j\le M} d_{a,j}\mathbf{v}_j\mathbf{v}_j^t))\mathbf{u}_a^{(t-1)} + 2\gamma \sum_{1\le j\le M} \mathbf{v}_j r_{a,j} \\
&= \Delta\mathbf{u}_a^{(t-1)} + 2\gamma \sum_{1\le j\le M} \mathbf{v}_j r_{a,j}
\end{aligned}$$

where $\Delta = ((1-2\gamma\lambda)\mathbf{I}_k - 2\gamma(\sum_{1\le j\le M} d_{a,j}\mathbf{v}_j\mathbf{v}_j^t))$ and $\mathbf{I}_k$ is a $k \times k$ identity matrix. Let $e = MAX_{epoch}$. To further optimize the training, we can expand all the epoches to get the final vector.

$$\begin{aligned}
\mathbf{u}_a^{(e)} &= \Delta^e\mathbf{u}_a^{(0)} + 2\gamma \sum_{1\le j\le M} (\sum_{i\in[0,e)} \Delta^i\mathbf{v}_j^t) r_{a,j} \\
&= \Delta^e\mathbf{u}_a^{(0)} + (2\gamma \sum_{i\in[0,e)} \Delta^i) \sum_{1\le j\le M} \mathbf{v}_j^t r_{a,j}
\end{aligned}$$

In the proposed protocol, we choose to precalculate Δ interactively, then do the training. The protocol is as follows.

1. RecSys computes $\mathbf{U}$ and $\mathbf{V}$ based on its expert dataset.

2. Alice and RecSys first interactively compute the matrix Δ and then RecSys runs the Algorithm 2 in the encrypted form, as shown in Figure 6.

3. RecSys can compute the predictions for Alice in the encrypted form. For instance, for the item i, the prediction is $[p_{a,i}^{(imf)}]_a = \langle [\mathbf{u}_a]_a, \mathbf{v}_i\rangle$.

	$\oplus/\ominus$	par. $\otimes$	Enc	Dec
Alice	$(k^2+k)\cdot(\rho M - 1)$	$(k^2+k)\cdot\rho M$	0	0
RecSys	0	0	$(k^2+k)M$	0

Table 6: Server-centric Complexity (offline)

	$\oplus/\ominus$	par. $\otimes$	Enc	Dec
Alice	$2k$	0	k	0
RecSys	k^2+k	k^2	0	k

Table 7: Server-centric Complexity (online)

Alice **RecSys**

(PK_a, SK_a) (PK_s, SK_s)

$[\mathbf{v}_j]_s, [\mathbf{v}_j\mathbf{v}_j^t]_s \ (1 \leq j \leq M)$

$[\Delta_1]_s = [\mathbf{v}_1\mathbf{v}_1^t]_s \otimes d_{a,1}$

$\ldots$

$[\Delta_M]_s = [\mathbf{v}_M\mathbf{v}_M^t]_s \otimes d_{a,M}$

$[\Delta]_s = \sum_{j:(a,j)\in\mathcal{R}} [\Delta_j]_s$

$\xrightarrow{[\Delta]_s}$

$\Delta = (1 - 2\gamma\lambda)\mathbf{I}_k - 2\gamma\Delta$

$[\mathbf{v}^t]_s = \sum_{j:(a,j)\in\mathcal{R}} [\mathbf{v}_j^t]_s \otimes r_{a,j}$

Choose a random vector $\mathbf{r}_a$

$[\mathbf{v}'^t]_s = [\mathbf{v}^t]_s \oplus \mathbf{r}_a$

$\xrightarrow{[\mathbf{v}'^t]_s}$

Choose a random vector $\mathbf{r}_s$

$\mathbf{v}''^t = \mathbf{v}'^t + \mathbf{r}_s$

$\xleftarrow{\mathbf{v}''^t}$

$[\mathbf{v}'^t]_a = [\mathbf{v}''^t - \mathbf{r}_a]_a$

$\xrightarrow{[\mathbf{v}'^t]_a}$

$[\mathbf{v}^t]_a = [\mathbf{v}'^t]_a \ominus \mathbf{r}_s$

$[\mathbf{v}^t]_a = (2\gamma \sum_{i\in[0,e)} \Delta^i)[\mathbf{v}^t]_a$

$[\mathbf{u}_a^{(e)}]_a = \Delta^e \mathbf{u}_a^{(0)} \oplus [\mathbf{v}^t]_a$

Figure 6: Server-centric IMF Protocol

Complexity. Note that Alice can compute $[d_{a,j}]_a, [r_{a,j}]$ $(1 \leq j \leq M)$ in the offline manner, and RecSys can compute $\mathbf{U}, \mathbf{V}$ and encrypt $\mathbf{V}$ in an offline manner. Let k, e, and ℓ be be defined as before, the number of offline and online computations are briefly summarized in Table 6 and 7 respectively.

Security. With respect to security, RecSys does not leak information to Alice except what can be inferred from the predictions. On the other hand, Alice will only leak the information that can be inferred from Δ. In the worst case, $d_{a,j}$ $(1 \leq j \leq M)$ might be leaked to RecSys. However, we argue that the leakage would be much smaller in reality. For a practical recommender system (e.g. our evaluation in Section 7.2 or Netflix), we can always assume that $M \gg k^2$. In this case, for a semi-honest RecSys, computing $d_{a,j}$ $(1 \leq j \leq M)$ from Δ is equivalent to some variant of sparse subset problem, which is shown to be hard in [21], unless that the density $M/\log(max(\Delta)) < 0.94$[9], which is around several hundreds in our case. Besides, the small k may not give a unique solution. When the RecSys is malicious (i.e. it can randomly set the $\mathbf{v}_j$ $(1 \leq j \leq M)$), it will be easy for RecSys to recover $d_{a,j}$ $(1 \leq j \leq M)$. However, we do not see any incentive for this kind of attack because it will disrupt the service and expose the attack to Alice, thus lose RecSys's customers.

In theory, it is easy to require Δ to be encrypted under PK_a so that RecSys needs to compute the Δ^t $(1 \leq t \leq e)$ in the encrypted form. However, consider the practical size of k and e, this will significantly increase the circuit depth for the SWHE scheme and the complexity will dramatically increase accordingly. In our design, we have made a tradeoff between security and efficiency on purpose.

6.2 Balanced IMF Protocol

As before, if the rating $r_{a,j} = 0$ we set $d_{a,j} = 0$, otherwise we set $d_{a,j} = 1$. The balanced protocol is as follows.

1. RecSys computes $\mathbf{U}$ and $\mathbf{V}$ based on its expert dataset, and publishes $[\mathbf{V}]_s$.
2. Alice and RecSys interactively run the Algorithm 2 in a privacy-preserving manner. Alice and RecSys perform the protocol shown in Figure 7. At the end of the last epoch, RecSys obtains $[\mathbf{u}_a]_a = [\mathbf{u}_a^{(e)}]_a$
3. RecSys can compute the prediction for Alice in the encrypted form. For instance, for the item i, the prediction is $[p_{a,i}^{(imf)}]_a = \langle[\mathbf{u}_a]_a, \mathbf{v}_i\rangle$.

Alice **RecSys**

$[\mathbf{v}_j]_s$, $[\mathbf{v}_j\mathbf{v}_j^t]_s \ (1 \leq j \leq M)$ (PK_s, SK_s)

$[\Delta_1]_s = [\mathbf{v}_1\mathbf{v}_1^t]_s \otimes 2\gamma d_{a,1}$

$\ldots$

$[\Delta_M]_s = [\mathbf{v}_M\mathbf{v}_M^t]_s \otimes 2\gamma d_{a,M}$

$[\Delta]_s = \sum_{j:(a,j)\in\mathcal{R}} [\Delta_j]_s$

$[\Psi]_s = \sum_{j:(a,j)\in\mathcal{R}} 2\gamma r_{a,j} \otimes [\mathbf{v}_j]_s$

$\xleftarrow{[\mathbf{u}_a^{(0)}]_s}$

// Interactive Training

For t = 1 : e-1

$[\Theta]_s = (1 - 2\gamma\lambda) \otimes [\mathbf{u}_a^{(t-1)}]_s$

$[\mathbf{u}_a^{(t)}]_s$

$= [\Theta]_s \oplus [\Psi]_s \ominus ([\Delta]_s \otimes [\mathbf{u}_a^{(t-1)}]_s)$

Choose a random vector $\mathbf{r}_a$

$[\mathbf{u}_a^{(t)}]_s \oplus \mathbf{r}_a$

$\xrightarrow{[\mathbf{u}_a^{(t)}+\mathbf{r}_a]_s}$

$\mathbf{u}_a^{(t)} + \mathbf{r}_a$

$\xleftarrow{[\mathbf{u}_a^{(t)}+\mathbf{r}_a]_s}$

$[\mathbf{u}_a^{(t)}]_s = [\mathbf{u}_a^{(t)} + \mathbf{r}_a]_s \ominus \mathbf{r}_a$

For t = e

$[\Theta]_s = (1 - 2\gamma\lambda) \otimes [\mathbf{u}_a^{(t-1)}]_s$

$[\mathbf{u}_a^{(t)}]_s$

$= [\Theta]_s \oplus [\Psi]_s \ominus ([\Delta]_s \otimes [\mathbf{u}_a^{(t-1)}]_s)$

Randomly choose $\mathbf{r}_a$

$[\mathbf{u}_a^{(t)}]_s \oplus \mathbf{r}_a$

$\xrightarrow{[\mathbf{u}_a^{(t)}+\mathbf{r}_a]_s}$

$\mathbf{u}_a^{(t)} + \mathbf{r}_a$

Randomly choose $\mathbf{r}_s$

$\xleftarrow{[\mathbf{u}_a^{(t)}+\mathbf{r}_a+\mathbf{r}_s]_a}$

$\mathbf{u}_a^{(t)} + \mathbf{r}_a + \mathbf{r}_s$

$\xleftarrow{[\mathbf{u}_a^{(t)}+\mathbf{r}_s]_a}$

$[\mathbf{u}_a^{(e)}]_a = [\mathbf{u}_a^{(t)} + \mathbf{r}_s]_a \ominus \mathbf{r}_s$

Figure 7: Balanced IMF Protocol

Complexity. As in the previous analysis, we assume RecSys can compute $\mathbf{U}, \mathbf{V}$ and encrypt $\mathbf{V}$ in an offline manner. The offline computation is the same as in the server-centric protocol, so we refer to Table 6 for offline complexity. Let k, e, and ℓ be be defined as before, the online complexity is briefly summarized in Table 8, where ρ is Alice's rating density.

	$\oplus/\ominus$	$\otimes$	par. $\otimes$	Enc	Dec
Alice	$(k^2+5k)\cdot e$	$k^2\cdot e$	$k\cdot e$	0	0
RecSys	0	0	0	$k\cdot e$	$k\cdot e$

Table 8: Balanced Complexity (online)

Security. With respect to security, due to the encryption, Alice and RecSys do not leak any information to each other even if both parties are malicious.

7. EVALUATING THE SOLUTION

In this section, we present additional security and efficiency analysis for the hybrid system from Section 4.

7.1 Security Analysis

For the hybrid recommender system, we are interested in the security for both an end user Alice and RecSys. With respect to Alice, referring to the system structure in Figure 2, it is clear that the behavior of the users from other groups do not affect Alice's security. We consider four attack scenarios, shown in Figure 8, which are concerned with Alice's rating values and the received predictions.

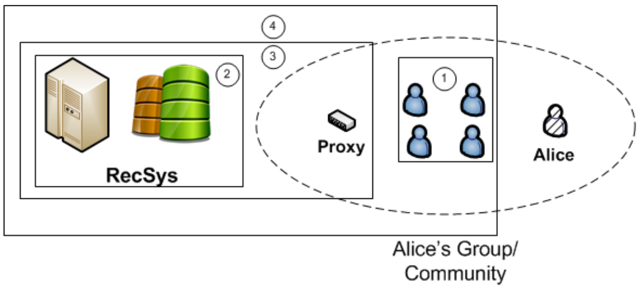

Figure 8: Attack Scenarios against Alice

- In the first scenario, all other users in Alice's group are malicious and collude with each other to figure out Alice's information. In this case, the malicious users will only have access to the encrypted values sent by Alice in the first step of the privacy-preserving UCF protocol, plus the final recommendation results. Based on the IND-CPA security of the encryption scheme, this means that the malicious users only gain what can be inferred from the legitimate predictions they receive.
- In the second scenario, the attacker is RecSys. In this case, the only possible place for information leakage is the privacy-preserving IMF protocol execution. Suppose that RecSys is semi-honest, then it learns no information about Alice's rating values in balanced protocol and learns very little in the server-centric protocol, according to the analysis in Section 6. If RecSys is malicious which means it can deviate from the protocol execution, then it does not learn anything in the balanced protocol and can learn which items Alice has rated. As indicated at the end of Section 6.1, RecSys is not incentivized to be malicious.
- In the third scenario, RecSys and the Proxy collude. Note the fact that the Proxy only aggregate the encrypted data in the privacy-preserving UCF protocol, we have the same conclusions as in the second scenario.
- In the fourth scenario, all players except for Alice collude. In this case, even if these players follow the protocol specification, they will learn both Alice's rating values and the received predictions simply by decrypting the encrypted data in step 1 of the privacy-preserving UCF protocol. This is an extreme scenario, and the consequence is implied by the recommendation utility.

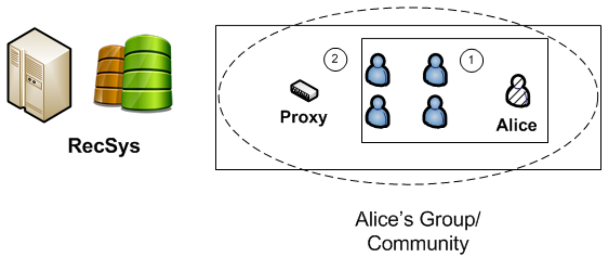

Figure 9: Attack Scenarios against RecSys

With respect to the security for RecSys, we consider two scenarios, shown in Figure 9. For the sake of simplicity, we only consider one user group.

- In one scenario, all users in the group are malicious and collude. The information leakage comes from the privacy-preserving IMF protocol. At the end, the malicious users can recover the some of the feature vectors $\mathbf{v}_j$ $(1 \leq j \leq M)$. The attack is simple, by trivially setting their user feature vectors $\mathbf{u}_i$ to be from $\mathbf{v}_j$ $(1 \leq j \leq M)$. Given the fact that all users collude, they can compute the UCF's contributions in the hybrid predictions they receive, and then they can obtain the selected $\mathbf{v}_j$ $(1 \leq j \leq M)$. However, when not all users collude, then the attack does not trivially work anymore. Since the malicious users do not know exactly the UCF's contributions in the hybrid predictions they receive, then it is much harder for them to recover some of the $\mathbf{v}_j$ $(1 \leq j \leq M)$. An aditional barrier is that each user can only contribute one user feature vector through the privacy-preserving protocol, therefore, M users need to collude in order to recover all $\mathbf{v}_j$ $(1 \leq j \leq M)$.
- In the other scenario, the Proxy also collude with the users. In this case, the consequence is the same as the above scenario when all users collude. If the Proxy is a branch of RecSys or has some commercial agreement, then it will not be incentivized to collude and leak RecSys's information.

Two Simple Enhancements. Assuming the Proxy and RecSys are semi-honest, we can introduce some uncertainty in two steps of the hybrid system from Section 4.

- One is in step 1, which executes the privacy-preserving UCF protocol. Referring to the specification in Section 5, the Proxy does not aggregate everything to the predictions. For instance, the partial prediction for Alice with respect to the item i is computed as follows. First randomly choose a set $\mathcal{S}$ to include some percentage of the users, and then compute the prediction as follows.

$$([\overline{r_a}]_a, \sum_{x \in \mathcal{S}} NOM_{x \xrightarrow{i} a}, \sum_{1 \le x \le T} DEN_{x \xrightarrow{i} a})$$

If the percentage is close to 1, then the recommendation accuracy will not be affected. As a result, the colluded users will not be able to know the exact UCF contributions. This enhancement mitigates the attacks against both Alice (by decreasing what other users can infer from the received predictions) and RecSys (making it harder to recover $\mathbf{v}_j$ $(1 \le j \le M)$).

- The other is in step 3 of the hybrid system from Section 4. In this case, RecSys can randomly set the contributing factor α for every item and every user. If the derivation of the α values is small, then this will not affect the recommendation accuracy much. As a result, this will make it very difficult for the malicious users to precisely recover $\mathbf{v}_j$ $(1 \le j \le M)$. This enhances the effect of the first enhancement.

7.2 Complexity Analysis

As in Section 3.2, we use the MovieLens 1M dataset. We set the feature dimension to be $k = 10$ and set the MAX_{epoch} to be $e = 30$, $T = 50$. Alice's rating vector has the density $\rho = 0.9\%$ and RecSys's expert dataset has the density $\rho' = 4.19\%$. We evaluate our scheme with Microsoft SEAL library [17] based on YASHE scheme. We select the ciphertext modulus $q = 2^{226} - 2^{26} + 1$, the polynomial modulus $p(x) = x^{8192} + 1$. Using Chinese Reminder Theorem, we select two 40-bit primes to represent the plaintext space of 2^{80}. The primes are 1099511922689 and 1099512004609. By packing 8192 plaintexts into one ciphertext, we can process 8192 multiplications in one homomorphic multiplication. Based on an Intel(R) Core(TM) i7-5600U CPU 2.60GHz, 8GB RAM, we have the timing complexity for Enc (52.43 ms), Dec (39.63 ms), $\otimes$ (207.76 ms), $par.\otimes$ (70.28 ms), $\oplus/\ominus$ (742 μs).

Considering that the homomorphic cryptosystem requires fix-point computation, we need to round all the float points during the training epochs. For the matrix $\Delta^e \mathbf{u}_a^{(0)}$ (denoted as Δ_e) in the protocol from Figure 6, we transform the float-point vector into an integer one by rounding the matrix to be $\Delta_r = \lfloor 2^{10} \Delta_e \rfloor$. We transform all float point matrices or vectors in the same way to integer ones.

	Alice (offline)	Alice (online)	RecSys (offline)	RecSys (online)
Server-centric	10.10	1.08	1110	14.30
Balanced	10.10	213.87	1110	5.52

Table 9: Timing Cost (Seconds)

The timing costs for Alice and RecSys in the protocols from Section 6.1 and 6.2 are summarized in Table 9. we have the following additional observations.

- The offline computational cost for RecSys is mainly the encryption of $[\mathbf{v}_j]_s$ and $[\mathbf{v}_j \mathbf{v}_j^t]_s$, only once for multiple users. The offline computation for Alice is the computation of $[\Delta]_s$ and $[\mathbf{v}^t]_s$ (or $[\Psi]_s$), which can be done any time and less frequently.

- For the server-centric protocol, RecSys can pack one matrix $[\mathbf{v}_j \mathbf{v}_j^t]_s$ or a vector $[\mathbf{v}_j]_s$ into one ciphertext so that the multiplication with $d_{a,j}$ is equivalent to one multiplication and the communication cost can be further reduced.

- For the balanced protocol, the server's computational cost is mainly the decryption and encryption, so we reduce it by letting Alice pack $\mathbf{u}_a^{(t)}$ into one ciphertext.

From the numbers, it is clear that the costs for both participants in the balanced protocol are more balanced than the server-centric protocol (considering both online and offline computations). The server-centric protocol performs better, at the risk of revealing Alice's rated items when RecSys is malicious.

7.3 Comparison

Compared with the schemes from [16, 22], we solve a cryptographically different problem since we only protect the incremental part rather than all the training process. As a result of the simplification, we get rid of the extra third-party CSP. Regarding the complexity, our solution performs significantly better.

- The communicational cost is reduced from 580 MB [16] to 5.73 MB in the server-centric Protocol, with 27 MB in the balanced protocol.

- The computational cost per round in [16] is 1.5 minutes on a PC with 3.4 GHz 6-core 64GB RAM, while the one in [22] costs 170 minutes on two servers with 1.9 GHz 16-cores 128 GB RAM. Our computational cost is much less than theirs.

While the schemes in [16, 22] deal with factorizing the whole rating matrix, which generates the $\mathbf{U}$ matrix for multiple users, our scheme generates one feature vector for a user Alice by default. However, if Alice is the holder of multiple users' data, it's easy to process them in one time by packing the Δ matrix from multiple users.

7.4 Complexity Analysis of UCF

Considering the simplicity of operations in UCF, we can select small parameters for the SWHE. We select the ciphertext modulus $q = 2^{190} - 2^{30} + 1$, the polynomial modulus $p(x) = x^{4096} + 1$. The plaintext space is 2^9. Based on an Intel(R) Core(TM) i7-5600U CPU 2.60GHz, 8GB RAM, we have the timing complexity for Enc (41.78 ms), Dec (41.22 ms), $par.\otimes$ (0.398 ms, the plaintext is much smaller), $\oplus/\ominus$ (85 μs). The time cost for Alice is 23046 seconds and for Proxy is 4106 seconds. But considering UCF is performed much less frequently than IMF, this is acceptable.

8. CONCLUSION

This paper has described a hybrid design for privacy-preserving recommender systems. Instead of focusing on a single recommender algorithm, we have proposed a new design in an attempt to facilitating privacy protection and enhancing recommendation accuracy. We have also attempted to improve the efficiency by slightly relax the privacy guarantees in the server-centric privacy-preserving IMF protocol. It shows that the efficiency can be dramatically improved while the users' privacy might not be affected in practical settings. Besides privacy, other issues such as robustness, are equally important or more important from the perspective of service providers. Note that the typical operations such as data cleaning becomes much more difficult in privacy-preserving systems (usually due to encryption). How to address these issues altogether remains as a challenging future work.

9. REFERENCES

[1] Aïmeur, E., Brassard, G., Fernandez, J.M., Onana, F.S.M.: Alambic: a privacy-preserving recommender system for electronic commerce. Int. J. Inf. Secur. 7, 307–334 (2008)

[2] Berlioz, A., Friedman, A., Kaafar, M.A., Boreli, R., Berkovsky, S.: Applying differential privacy to matrix factorization. In: Proceedings of the 9th ACM Conference on Recommender Systems. pp. 107–114 (2015)

[3] Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Cryptography and Coding, pp. 45–64. Springer (2013)

[4] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309–325 (2012)

[5] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Advances in Cryptology — CRYPTO 2011. pp. 505–524. Springer (2011)

[6] Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)

[7] Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: "you might also like: " privacy risks of collaborative filtering. In: 32nd IEEE Symposium on Security and Privacy, S&P 2011. pp. 231–246 (2011)

[8] Canny, J.F.: Collaborative filtering with privacy. In: IEEE Symposium on Security and Privacy. pp. 45–57 (2002)

[9] Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. computational complexity 2(2), 111–128 (1992)

[10] Dankar, F.K., Emam, K.E.: Practicing differential privacy in health care: A review. Trans. Data Privacy 6(1), 35–67 (2013)

[11] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 169–178 (2009)

[12] GroupLens: http://grouplens.org/datasets/movielens/1m/

[13] Guerraoui, R., Kermarrec, A., Patra, R., Taziki, M.: D2P: Distance-based Differential Privacy in Recommenders. Proc. VLDB Endow. 8(8), 862–873 (2015)

[14] Haeberlen, A., Pierce, B.C., Narayan, A.: Differential privacy under fire. In: Proceedings of the 20th USENIX Conference on Security. pp. 33–33 (2011)

[15] Jeckmans, A., Peter, A., Hartel, P.H.: Efficient privacy-enhanced familiarity-based recommender system. In: Computer Security - ESORICS 2013. LNCS, vol. 8134, pp. 400–417. Springer (2013)

[16] Kim, S., Kim, J., Koo, D., Kim, Y., Yoon, H., Shin, J.: Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. pp. 617–628 (2016)

[17] Library, M.S.: https://sealcrypto.codeplex.com/

[18] Liu, Z., Wang, Y.X., Smola, A.: Fast differentially private matrix factorization. In: Proceedings of the 9th ACM Conference on Recommender Systems. pp. 171–178. ACM (2015)

[19] McSherry, F., Mironov, I.: Differentially private recommender systems: building privacy into the Netflix prize contenders. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 627–636 (2009)

[20] Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: 2008 IEEE Symposium on Security and Privacy (S&P 2008). pp. 111–125 (2008)

[21] Nguyen, P., Stern, J.: The hardness of the hidden subset sum problem and its cryptographic implications. In: Annual International Cryptology Conference. pp. 31–46. Springer

[22] Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., Boneh, D.: Privacy-preserving matrix factorization. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. pp. 801–812 (2013)

[23] Shen, Y., Jin, H.: Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In: Proceedings of the 2014 IEEE International Conference on Data Mining. pp. 540–549 (2014)

[24] Shokri, R., Pedarsani, P., Theodorakopoulos, G., Hubaux, J.: Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In: Proceedings of the third ACM conference on Recommender systems (RecSys '09). pp. 157–164 (2009)

[25] Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1310–1321 (2015)

[26] Tang, Q., Wang, J.: Privacy-preserving context-aware recommender systems: Analysis and new solutions. In: Computer Security - ESORICS 2015. LNCS, vol. 9327, pp. 101–119. Springer (2015)

[27] Tang, Q., Wang, J.: Privacy-preserving friendship-based recommender systems. Cryptology ePrint Archive: Report 2015/1152 (2015)

[28] Veugen, T., de Haan, R., Cramer, R., Muller, F.: A framework for secure computations with two non-colluding servers and multiple clients, applied to recommendations. IEEE Transactions on Information Forensics and Security 10(3), 445–457 (2015)

[29] Weinsberg, U., Bhagat, S., Ioannidis, S., Taft, N.: Blurme: inferring and obfuscating user gender based on ratings. In: Sixth ACM Conference on Recommender Systems, RecSys '12. pp. 195–202 (2012)