Privacy-preserving Friendship-based Recommender Systems

Qiang Tang and Jun Wang

Abstract

Privacy-preserving recommender systems have been an active research topic for many years. However, until today, it is still a challenge to design an efficient solution without involving a fully trusted third party or multiple semi-trusted third parties. The key obstacle is the large underlying user populations (i.e. huge input size) in the systems. In this paper, we revisit the concept of friendship-based recommender systems, proposed by Jeckmans et al. and Tang and Wang. These solutions are very promising because recommendations are computed based on inputs from a very small subset of the overall user population (precisely, a user's friends and some randomly chosen strangers). We first clarify the single prediction protocol and Top-n protocol by Tang and Wang, by correcting some flaws and improving the efficiency of the single prediction protocol. We then design a decentralized single protocol by getting rid of the semi-honest service provider. In order to validate the designed protocols, we crawl Twitter and construct two datasets (FMT and 10-FMT) which are equipped with auxiliary friendship information. Based on 10-FMT and MovieLens 100k dataset with simulated friendships, we show that even if our protocols use a very small subset of the datasets, their accuracy can still be equal to or better than some baseline algorithm. Based on these datasets, we further demonstrate that the outputs of our protocols leak very small amount of information of the inputs, and the leakage decreases when the input size increases. We finally show that he single prediction protocol is quite efficient but the Top-n is not. However, we observe that the efficiency of the Top-n protocol can be dramatically improved if we slightly relax the desired security guarantee.

I. Introduction

Recommender system is one type of information filtering systems that seek to predict the preferences that users would give to an item (e.g. music, book, or movie) they have not yet considered, using a model built from the characteristics of items and/or users. It enables users to make the most appropriate choices from the immense variety of items that are available. Take an online book store as an example, going through the lengthy book catalogue not only wastes a lot of time but also frequently overwhelms users and leads them to make poor decisions. Without recommender systems, the availability of choices, instead of producing a benefit, may downgrade users' experiences. Today, recommender systems play an important role in every corner of our daily life. Two representative types of recommender systems are neighborhood-based and model-based. In a neighborhood-based recommender system, in order to predict user u's rating for an item i, the system first chooses a neighborhood for user u or the item i then computes the prediction based on data from the neighborhood. In a model-based recommender system, in order to predict user u's rating for an item i, the system first trains a model using all available data then computes the prediction based on the model.

In practice, 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 a double-edged sword. On one side users get better recommendations when they reveal more personal data, but on the flip side they sacrifice more privacy if they do so. For instance, Narayanan and Shmatikov [22] presented a robust de-anonymization attack against anonymized Netflix dataset. Weinsberg et al. [36] showed that what has been rated by a user can already help an attacker identify this user, namely the breach of data privacy may lead to the breach of anonymity. Calandrino et al. [9] proposed 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. More detailed discussions about privacy issues in recommender systems can be found in [4], [17], [25].

A. Related Work

Existing privacy-protection solutions can be generally divided into two categories. The cryptographic solutions (e.g. [1], [10], [23], [28], [32]) often aim at securing the procedure of underlying recommender protocols, namely they do not consider the information leakage in the outputs. In this category, a typical method is to employ somewhat homomorphic encryption scheme and let all computations be done in encrypted form. Unfortunately, this will incur intolerable complexities and make the solutions impractical. Even though in the neighborhood-based systems, predictions are computed based on a small subset of users'data, a secure solution must compute the neighborhood privately in the first place, and this often introduces a lot of complexity. Moreover, many solutions (e.g. [23], [32]) introduce additional semi-trusted servers which are difficult to be instantiated

in reality. The data-obfuscation solutions (e.g. [24], [27], [37]) rely on adding noise to the original data or computation results to protect users' inputs. These solutions usually do not incur complicated manipulations on the users' inputs, so that they are much more efficient. The drawback is that they often lack rigorous privacy guarantees and downgrade the recommendation accuracy to some extent. For instance, Zhang et al. [38] showed how to recover perturbed ratings in the solutions from [24]. With respect to privacy guarantees, an exception is the differential privacy based approach (e.g. [3], [20], [21]) which provide mathematically sound privacy notions. However, these solutions either require a trusted third party (trusted curator in term of differential privacy) or need cryptographic primitives for all users to generate the accumulated data subjects (e.g. sums and covariance matrix).

In order to improve the efficiency of privacy-preserving recommender systems, one typical approach is to design more efficient cryptographic tools. However, even if the speedup is significant in cryptographic sense, it often does not result in practical recommender systems. This is due to the large underlying user populations, which make model training and neighborhood selection unrealistic even with efficient cryptographic tools. Recently, Jeckmans et al. proposed an interesting solution direction in [16], where they proposed the concept of friendship-based recommender system and gave solutions based on somewhat homomorphic encryption schemes. The rationale behind their concept is the following.

- In order to avoid the computationally-cumbersome neighborhood selection step in neighborhood-based recommender systems, the solution leverages auxiliary social network information of the users. This significantly reduce the amount of data used in computing predictions.
- Trust is a very subtle issue. Friends may trust each other in the sense that their peers will not collude with a third party to leak their information. If a collusion is discovered, then their relationship can be broken. On the other hand, some information may be sensitive among friends, but not with strangers. For instance, if a user has watched a porn movie, then disclosing this to his friends may make him embarrassed, but disclosing it to a stranger may not cause any harm. This motivates the adoption of homomorphic encryption to secure the computation.

Later, Tang and Wang [29] pointed out some security flaws in the protocol from [16] and gave improved solutions. All solutions from [16], [29] rely on the hypothesis that friends share more similar tastes than with strangers without any validation. Moreover, some proper analysis is missing for security and implementation results are missing to demonstrate the efficiency.

B. Our Contribution

In this paper, we revisit the concept of friendship-based recommender system and the protocols from [16], [29]. Generally speaking, we validate the hypothesis made in [16], [29] and comprehensively analyse existing and newly designed protocols. In more detail, our contribution lies in three aspects.

- We construct and analyze two Twitter datasets (i.e. FMT and 10-FMT) by defining friendship based on the *following* activities in Twitter. Based on 10-FMT, we experimentally validate the hypothesis that friends are more similar in the movie rating behaviors. Besides serving for evaluating our recommender protocols, the datasets are of independent interests for the community.
- We clarify the single prediction and Top-n protocols from [29] and correct some flaws in their specifications. The clarified single prediction protocol is more efficient than the original one. We further propose a new decentralized single prediction protocol, by getting rid of the service provider and basing its security solely on the semi-honest assumption among friends.
- We analyze the performances of the protocols. Firstly, we evaluate the recommendation accuracy of our protocols and show that they can achieve better accuracy than some baseline algorithm. Secondly, we analyse the security of our protocols. Since the security of protocol executions is straightforwardly guaranteed by the underlying homomorphic encryption scheme, we focus on the information leakage in algorithm outputs. Thirdly, regarding computational complexity, we provide both asymptotic and implementation results for the protocols. We show that the single prediction protocol is very efficient while the Top-n protocol is not. We further discuss two relaxations for the top-n protocol and show that they are quite efficient.

C. Organization

The rest of this paper is organized as follows. In Section II, we present preliminaries on notation and building blocks. In Section III, we recap the rating prediction algorithms from [16] and [29]. In Section IV, we define security models. In Section V, we construct and analyse two Tweeter datasets. In Section VI, we clarify the single prediction and Top-n protocols from [29]. In Section VII, we present a decentralized privacy-preserving single prediction protocol. In Section VIII, we analyze the accuracy properties of the protocols. In Section IX, we analyze the security properties of the protocols. In Section X, we analyze the complexity properties of the protocols. In Section XI, we conclude the paper.

II. Preliminary

When X is a set, $x \xleftarrow{\$} X$ means that x is chosen from X uniformly at random, and $|X|$ means the size of X. If χ is a distribution, then $s \leftarrow \chi$ means that s is sampled according to χ. We use bold letter, such as $\mathbf{X}$, to denote a vector. Given two vector $\mathbf{X}$ and $\mathbf{Y}$, we use $\mathbf{X} \cdot \mathbf{Y}$ to denote their inner product. We use $\| \mathbf{X} \|$ to denote the Euclidean length of $\mathbf{X}$.

In a recommender system, the item set is denoted by $\mathbf{I} = (1, 2, \cdots, b, \cdots, M)$, and a user x's ratings are denoted by a vector $\mathbf{R}_x = (r_{x,1}, \cdots, r_{x,b}, \cdots, r_{x,M})$. 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. The ratings are often organized in a rating matrix, as shown in Table I. The functionality of a recommender system is to predict the unrated $r_{x,i}$ values.

	Item 1	$\cdots$	Item b	$\cdots$	Item M
User 1 ($\mathbf{R}_1$)	$r_{1,1}$	$\cdots$	$r_{1,b}$	$\cdots$	$r_{1,M}$
User 2 ($\mathbf{R}_2$)	$r_{2,1}$	$\cdots$	$r_{2,b}$	$\cdots$	$r_{2,M}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
User N ($\mathbf{R}_N$)	$r_{N,1}$	$\cdots$	$r_{N,b}$	$\cdots$	$r_{N,M}$

TABLE I: Rating Matrix

Given two rating vectors $\mathbf{R}_x$ and $\mathbf{R}_y$ from users x and y, their Cosine similarity is computed as follows.

$$sim(x, y) = \frac{\mathbf{R}_x \cdot \mathbf{R}_y}{\| \mathbf{R}_x \| \cdot \| \mathbf{R}_y \|}$$

With respect to $\mathbf{R}_x$, a binary vector $\mathbf{Q}_x = (q_{x,1}, \cdots, q_{x,b}, \cdots, q_{x,M})$ is defined as follows: $q_{x,b} = 1$ iff $r_{x,b} \neq 0$ for every $1 \leq b \leq M$. Basically, $\mathbf{Q}_x$ indicates which items have been rated by user x. We further use $\overline{r_x}$ to denote user x's average rating, namely $\lceil \frac{\sum_{i \in \mathbf{B}} r_{x,i}}{\sum_{i \in \mathbf{B}} q_{x,i}} \rfloor$.

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

$$MAE = \frac{1}{|\Gamma|} \sum_{\hat{r}_{u,i} \in \Gamma} |\hat{r}_{u,i} - r_{u,i}|,$$

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 MAE implies more accurate recommendations.

A. Somewhat Homomorphic Encryption

Since the breakthrough work of Gentry [14], many somewhat homomorphic encryption (SWHE) schemes have been proposed (e.g. the BV scheme [8], BGV scheme [7], YASHE scheme [6]). A SWHE scheme can be described by four algorithms ($\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$), where the Eval algorithm can only be executed for a limited number of times.

- $\mathsf{Keygen}(\lambda)$: this algorithm outputs a public/private key pair (PK, SK).
- $\mathsf{Enc}(PK, m)$: this algorithm outputs a ciphertext c.
- $\mathsf{Dec}(SK, c)$: this algorithm outputs a plaintext m or an error $\perp$.
- $\mathsf{Eval}(\mathsf{op}, \mathbf{c}_\alpha, \mathbf{c}_\beta)$: suppose $\mathbf{c}_\alpha$ is a ciphertext of α and $\mathbf{c}_\beta$ is a ciphertext of β, this algorithm outputs a ciphertext for the plaintext $\alpha \, \mathsf{op} \, \beta$. The operator op is either addition + or multiplication $\cdot$.

Throughout the paper, given a public/private 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. In comparison, $\mathsf{Enc}(PK_u, m)$ represents the probabilistic output of running Enc for the message m. When $\mathbf{m}$ is a vector of messages, we use $\mathsf{Enc}(PK_u, \mathbf{m})$ to denote the vector of ciphertexts, where encryption is done for each element independently. We use the notation $\sum_{1 \leq i \leq N} [m_i]_u$ to denote the result of sequentially applying $\mathsf{Eval}(+, ,)$ to the cipheretxts.

III. Tailored Prediction Algorithms

In this section, we briefly review the prediction algorithms from [16], [29]. We refer to the algorithm from [16] as the JPH algorithm, and the algorithm from [29] as the TW algorithm.

A. JPH Prediction Algorithm

Given a user u, let his friend set be $\mathbf{F}_u$. The JPH prediction algorithm is defined as follows, where $w_{u,f}$ and $w_{f,u}$ are the weights that users u and f assign to each other.

$$\begin{aligned} p_{u,b} &= \frac{\sum_{f \in \mathbf{F}_u} q_{f,b} \cdot r_{f,b} \cdot (\frac{w_{u,f} + w_{f,u}}{2})}{\sum_{f \in \mathbf{F}_u} q_{f,b} \cdot (\frac{w_{u,f} + w_{f,u}}{2})} \\ &= \frac{\sum_{f \in \mathbf{F}_u} r_{f,b} \cdot (w_{u,f} + w_{f,u})}{\sum_{f \in \mathbf{F}_u} q_{f,b} \cdot (w_{u,f} + w_{f,u})} \end{aligned} \quad (1)$$

In [16], the authors only discussed the security properties of their solutions without touching upon the performances. In practice, friends share similar tastes may imply they also have rated similar items. Therefore, if user u has not rated item b then it is very likely that very few friends have rated the item b. If this happens, the predicated value from Equation (1) may not be very accurate (cold start problem). In Section VIII-C, we validate this argument with experimental results. In fact, this partially motivates the inclusion of strangers in TW prediction algorithms.

B. Revised TW Prediction Algorithm

Given an active user u, when factoring in the inputs from randomly chosen strangers, we will use the simple Bias From Mean (BFM) scheme for the purpose of simplicity. It is worth stressing that there are a lot of different choices for this task. Nevertheless, as to the accuracy, this scheme has similar performance to many other more sophisticated schemes, such as Slope One and Pearson/Cosine similarity-based collaborative

filtering schemes [19]. Let the stranger set be $\mathbf{T}_u$, the predicted value $p^*_{u,b}$ for item b is computed as follows.

$$p^*_{u,b} = \overline{r_u} + \frac{\frac{1}{2}\sum_{t\in\mathbf{T}_u} q_{t,b}\cdot(r_{t,b}-\overline{r_t})}{\sum_{t\in\mathbf{T}_u} q_{t,b}} \quad (2)$$

Note that we put a factor $\frac{1}{2}$ in the computation, which means that the weight of a stranger's contribution to the output is set to be $\frac{1}{2}$. In [29], this factor is set to 1 by default and implies that a stranger's weight to the output is higher than that of a friend. Even though in principle this factor can be flexibly set in the range of $[0,1]$ but we argue that $\frac{1}{2}$ is more appropriate than 1 because a friend should naturally contribute more to the output in a friendship-based recommender system.

When factoring in the inputs from the friends, we make two changes to Equation (1). One is to only take into account the weight value from user u. This makes more sense because how important a friend means to user u is a subjective matter for u only. The other is to compute the predication based on both u's average rating and the weighted rating deviations from his friends. Let the friend set be $\mathbf{F}_u$, the predicted value $p^{**}_{u,b}$ for item b is computed as follows.

$$p^{**}_{u,b} = \overline{r_u} + \frac{\sum_{f\in\mathbf{F}_u} q_{f,b}\cdot(r_{f,b}-\overline{r_f})\cdot w_{u,f}}{\sum_{f\in\mathbf{F}_u} q_{f,b}\cdot w_{u,f}} \quad (3)$$

In practice, the similarity between friends means that they tend to prefer similar items. However, this does not imply that they will assign very similar scores to the items. For example, a user Alice may be very mean and assign a score 3 to most of her favorite items while her friends may be very generous and assign a score 5 to their favorite items. Using the Equation (1), we will likely generate a score 5 for an unrated item for Alice, who may just rate a score 3 for the item even if she likes it. In this regard, Equation (3) is more appropriate because $\overline{r_u}$ reflects the user's rating style and $\frac{\sum_{f\in\mathbf{F}_u} q_{f,b}\cdot(r_{f,b}-\overline{r_f})\cdot w_{f,u}}{\sum_{f\in\mathbf{F}_u} q_{f,b}\cdot w_{f,u}}$ reflects the user's preference based on inputs from his friends.

Based on the inputs from the strangers and friends, a combined predicted value $p_{u,b}$ for an unrated item b can be computed as $p_{u,b} = \rho\cdot p^*_{u,b} + (1-\rho)\cdot p^{**}_{u,b}$ for some $0 \le \rho \le 1$. Due to the fact that cryptographic primitives are often designed for dealing with integers, we rephrase the formula as follows, where α, β are integers.

$$p_{u,b} = \frac{\beta}{\alpha+\beta}\cdot p^*_{u,b} + \frac{\alpha}{\alpha+\beta}\cdot p^{**}_{u,b} \quad (4)$$

It is worth noting that the prediction $p_{u,b}$ is not deterministic because we assume the stranger set $\mathbf{T}_u$ is randomly chosen for the computation.

C. An Additional Remark

In addition to solving the cold start problem in recommender systems, the other motivation to include the contributions from strangers is to reduce the information leakages of friends in the output for user u. We perform two experiments and put the results in Fig. 1 and 2, by following the same experiment procedure as in Section IX-A. It shows that when the number of friends is small, each of them may have quite big influence in the output (particularly compared to the experiment results in Section IX-A).

IV. Security Models

A. Basic Security Model

In the basic security model, we assume the service provider is semi-honest, which means it will follow the protocol specification and does not participate in the protocol as a user. Moreover, a user trusts his friends to be semi-honest. As to communication channel among users, we assume all communications are protected with respect to integrity and confidentiality (with forward secrecy). In practice, any user or the service provider can be compromised, so that it is important to investigate the security guarantee in such situations. We refine the security properties accordingly in Section IV-B.

Let the users be indexed by an integer $x \ge 1$. We assume user x has a public/private key pair (PK_x, SK_x) and a rating vector $\mathbf{R}_x$, the service provider has (PK_s, SK_s). We further assume the social graph (denoted as SG) among the users to be public information. We abstractly denote the recommender protocol as $\mathsf{RSProtocol}$ and let it output $\mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t\ \forall t \in \mathbf{T}_u)$ to user u and output nothing to others. Note that the service provider will randomly set $\mathbf{T}_u$ in the execution of $\mathsf{RSProtocol}$.

As a standard cryptographic practice, every security property is modeled as a game between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$. In the game, the challenger $\mathcal{C}$ simulates the honest participants while the attacker plays the role of compromised or malicious participants, and the computation is indeed done via a two-party protocol between $\mathcal{C}$ and $\mathcal{A}$. We abstractly represent the protocol execution with $\mathsf{RSProtocol}(\Phi;\Psi)$, where Φ are the inputs from $\mathcal{C}$ and Ψ are the inputs from $\mathcal{A}$. In the definitions, we omit the input of public parameters for simplicity. In order to model the fact that the attacker may have a wide range of background information about the rating matrices, we let the attacker define the rating matrices $\mathbf{R}_x$ for all x. Before formally describing the definitions, we point out a clear distinction between them due to the fact that only the active user u will receive an output at the end of protocol execution.

- In Definitions from Fig. 3, 4, 6, the attacker models participants other than user u so that we want to enforce the concept that the attacker learns nothing about u's inputs.
- In Definitions from Fig. 5 and 7, the attacker models user u, therefore we can only guarantee that the attacker learns the information in the output but nothing else.

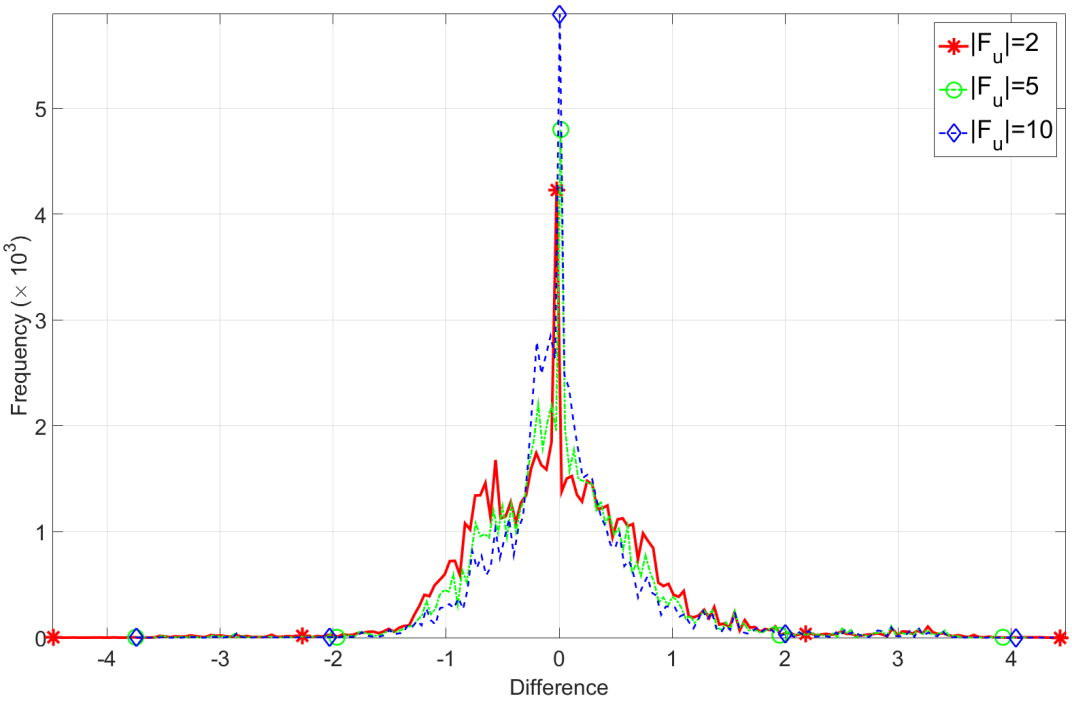

Fig. 1: Single Friend Influence (without involving any stranger) on 10-FMT Dataset. For clearly presenting those differences that fall out of the range $[-10^{-5}, 10^{-5}]$ and keeping the histogram structure, ≈99% of the differences that fall into range $[-10^{-5}, 10^{-5}]$ have been removed.

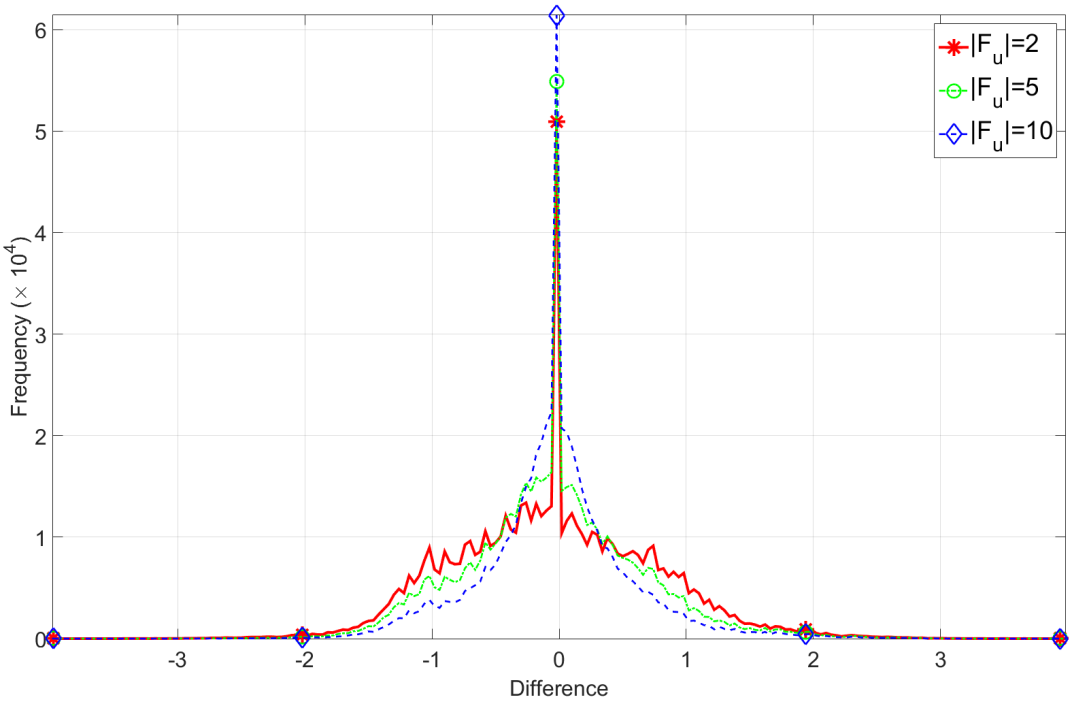

Fig. 2: Single Friend Influence (without involving any stranger) on MovieLens 100k Dataset. For clearly presenting those differences that fall out of the range $[-10^{-5}, 10^{-5}]$ and keeping the histogram structure, ≈99% of the differences that fall into range $[-10^{-5}, 10^{-5}]$ have been removed.

Semi-honest recommender service provider. In the view of all users, the service provider will follow the protocol specification but it may try to infer their private information from any collected transaction records. The attack game is shown in Fig. 3, and the detailed description follows.

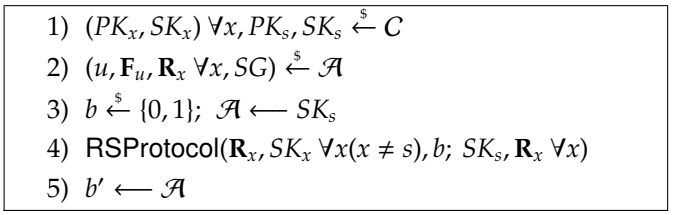

Fig. 3: Security against Semi-honest Service Provider

1) The challenger generates key pairs (PK_x, SK_x) for all x and (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x\ \forall x$, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$.
3) The challenger generates a random bit b and gives SK_s to $\mathcal{A}$.
4) The challenger and the attacker run the RSProtocol protocol, where the challenger simulates all users and the attacker simulates the service provider. In the protocol execution, if $b = 0$ the challenger uses the original rating matrices $\mathbf{R}_x$ for all x, otherwise it uses random matrices in the computation.
5) At the end, the attacker outputs a bit b'.

Malicious strangers. In the view of user u, the involved strangers in the protocol execution may try to learn his private information. *We assume the strangers are malicious, meaning that the attacker does not need to follow the protocol specification in the game.* The attack game is shown in Fig. 4, and the detailed description follows.

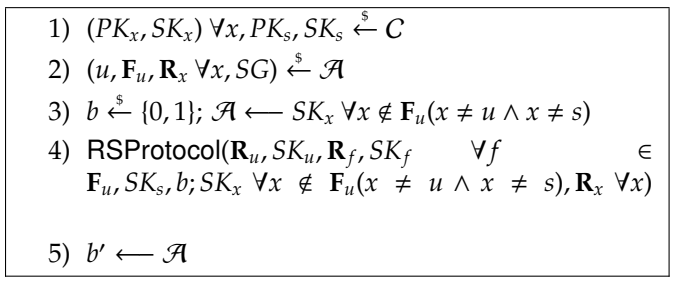

Fig. 4: Security against all Strangers

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$.
3) The challenger generates a random bit b, and gives SK_x for all $x \notin \mathbf{F}_u$ except for SK_u and SK_s to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The challenger simulates user u, users from $\mathbf{F}_u$ and the service provider, and the attacker simulates the rest. In the protocol execution, if $b = 0$ the challenger uses the original rating matrices $\mathbf{R}_u$ and $\mathbf{R}_f$ for all $f \in \mathbf{F}_u$, otherwise it uses some random matrices in the computation.
5) At the end, the attacker outputs a bit b'.

In the view of a stranger from $\mathbf{T}_u$, user u may try to learn his private information (even colluding with u's friends in $\mathbf{F}_u$) by acting maliciously. The attack game is shown in Fig. 5, and the detailed description follows.

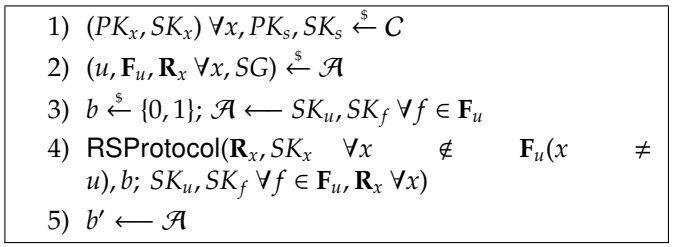

Fig. 5: Security for any Stranger against User u

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$.
3) The challenger generates a random bit b, and gives SK_u and SK_f for all $f \in \mathbf{F}_u$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The attacker simulates user u and users from $\mathbf{F}_u$, and the challenger simulates the rest. In the protocol execution, the challenger first samples the stranger set $\mathbf{T}_u$, and then proceeds as follows.
   - If $b = 0$ the challenger uses the original rating matrices $\mathbf{R}_t$ for all $t \in \mathbf{T}_u$.
   - Otherwise it uses any $\mathbf{R}_t^*$ for all $t \in \mathbf{T}_u$ such that $\mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t\ \forall t \in \mathbf{T}_u) = \mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t^*\ \forall t \in \mathbf{T}_u)$.
5) At the end, the attacker outputs a bit b'.

Threat from a semi-honest friend. In the view of user u, we assume that none of his friends will collude with the recommender service server or another party to breach his privacy and they will follow the protocol specification. It is reasonable to assume that the social norm deters such colluding attacks, and the deterrence comes from the fact that once such a collusion is known to the victim user then the friendship may be jeopardized. The attack game is shown in Fig. 6, and the detailed description follows.

1) $(PK_x, SK_x)\ \forall x, PK_s, SK_s \xleftarrow{\$} \mathcal{C}$
2) $(u, \mathbf{F}_u, f^\dagger \in \mathbf{F}_u, \mathbf{R}_x\ \forall x, SG) \xleftarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} \{0,1\}$; $\mathcal{A} \longleftarrow SK_{f^\dagger}$
4) $\mathsf{RSProtocol}(\mathbf{R}_x, SK_x\ \forall x(x \neq f^\dagger), b;\ SK_{f^\dagger}, \mathbf{R}_x\ \forall x)$
5) $b' \longleftarrow \mathcal{A}$

Fig. 6: Security against any Semi-honest Friend

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$. The attacker also chooses $f^\dagger \in \mathbf{F}_u$.
3) The challenger generates a random bit b, and gives $SK_{f^\dagger}$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol, where the attacker simulates user $f^\dagger$ and the challenger simulates the rest. In the protocol execution, if $b = 0$ the challenger uses the original rating matrices $\mathbf{R}_x$ for all x $(x \neq f^\dagger)$, otherwise it uses random matrices in the computation.
5) At the end, the attacker outputs a bit b'.

In the view of $f^\dagger \in \mathbf{F}_u$, user u and u's other friends will follow the protocol but may try to infer $f^\dagger$'s input. The attack game is shown in Fig. 7, and the detailed description follows.

1) $(PK_x, SK_x)\ \forall x, PK_s, SK_s \xleftarrow{\$} \mathcal{C}$
2) $(u, \mathbf{F}_u, f^\dagger \in \mathbf{F}_u, \mathbf{R}_x\ \forall x, SG) \xleftarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} \{0,1\}$; $\mathcal{A} \longleftarrow SK_u, SK_f\ \forall f \in \mathbf{F}_u(f \neq f^\dagger)$
4) $\mathsf{RSProtocol}(\mathbf{R}_{f^\dagger}, SK_{f^\dagger}, \mathbf{R}_x, SK_x\ \forall x \notin \mathbf{F}_u(x \neq u), b;\ SK_u, SK_f\ \forall f \in \mathbf{F}_u(f \neq f^\dagger), \mathbf{R}_x\ \forall x)$
5) $b' \longleftarrow \mathcal{A}$

Fig. 7: Security for a Friend against User u and other Friends

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$. The attacker also chooses $f^\dagger \in \mathbf{F}_u$.
3) The challenger generates a random bit b, and gives SK_u and SK_f for all $f \in \mathbf{F}_u$ except for $SK_{f^\dagger}$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The attacker simulates user u and users from $\mathbf{F}_u$ except for $f^\dagger$, and the challenger simulates the rest. In the protocol execution, the challenger first samples the stranger set $\mathbf{T}_u$, and then proceeds as follows.
   - If $b = 0$ the challenger uses the original rating matrices $\mathbf{R}_t$ for all $t \in \mathbf{T}_u$ and $\mathbf{R}_{f\dagger}$.
   - Otherwise it uses any $\mathbf{R}^*_t$ for all $t \in \mathbf{T}_u$ and $\mathbf{R}^*_{f^\dagger}$ such that $\mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f \forall f \in \mathbf{F}_u, \mathbf{R}_t \forall t \in \mathbf{T}_u) = \mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u(f \neq f^\dagger), \mathbf{R}^*_{f\dagger}, \mathbf{R}^*_t\ \forall t \in \mathbf{T}_u)$.
5) At the end, the attacker outputs a bit b'.

B. Worst-case Security Model

In the basic security model, the security definitions leverage on the assumption that friends and the service provider are semi-honest. Next, we relax this assumption and allow friends to be compromised.

For the active user u, the worst-case scenario is that all other parties collude and act maliciously. We can define the security property as shown in Fig. 8. This can be regarded as a combined version of the games in Fig. 3, 4, 6, by assuming a malicious attacker.

1) $(PK_x, SK_x)\ \forall x, PK_s, SK_s \xleftarrow{\$} \mathcal{C}$
2) $(u, \mathbf{F}_u, \mathbf{R}_x\ \forall x, SG) \xleftarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} \{0,1\}$; $\mathcal{A} \longleftarrow SK_x\ \forall x(x \neq u)$
4) $\mathsf{RSProtocol}(\mathbf{R}_u, SK_u, b;\ SK_x\ \forall x(x \neq u), \mathbf{R}_x\ \forall x)$
5) $b' \longleftarrow \mathcal{A}$

Fig. 8: Worst-case Security for User u

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$.
3) The challenger generates a random bit b, and gives SK_x for all x except for SK_u to the attacker.
4) The challenger and the attacker run the RSProtocol protocol, where the challenger simulates user u and the attacker simulates the rest. In the protocol execution, if $b = 0$ the challenger uses the original rating matrix $\mathbf{R}_u$, otherwise it uses some random matrix in the computation.
5) At the end, the attacker outputs a bit b'.

In the view of a stranger $t^\dagger \in \mathbf{T}_u$, the worst-case scenario is user u colludes with all other users, but not the service provider (otherwise it is impossible to get privacy anymore). The attack game is shown in Fig. 9, and it is an enhanced version of the attack game in Fig. 5 in the malicious model.

1) $(PK_x, SK_x)\ \forall x, PK_s, SK_s \xleftarrow{\$} C$
2) $(u, \mathbf{F}_u, t^\dagger, \mathbf{R}_x\ \forall x, SG) \xleftarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} \{0,1\}$; $\mathcal{A} \longleftarrow SK_x\ \forall x(x \neq t^\dagger, x \neq s)$
4) $\mathsf{RSProtocol}(\mathbf{R}_{t^\dagger}, SK_{t^\dagger}, SK_s, b;\ SK_x\ \forall x(x \neq t^\dagger, x \neq s), \mathbf{R}_x\ \forall x)$
5) $b' \longleftarrow \mathcal{A}$

Fig. 9: Worst-case Security for any Stranger

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$. The attacker also chooses $t^\dagger \notin \mathbf{F}_u$ and $t^\dagger \neq u$.
3) The challenger generates a random bit b, and gives SK_x for all x except for SK_s and $SK_{t^\dagger}$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The challenger simulates the service provider and user $t^\dagger$, and the challenger simulates the rest. In the protocol execution, the challenger first samples the stranger set $\mathbf{T}_u$ which should include $t^\dagger$, and then proceeds as follows.
   - If $b = 0$ the challenger uses the original rating matrix $\mathbf{R}_{t^\dagger}$.
   - Otherwise it uses any $\mathbf{R}^*_{t^\dagger}$ such that $\mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t\ \forall t \in \mathbf{T}_u) = \mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t\ \forall t \in \mathbf{T}_u(t \neq t^\dagger), \mathbf{R}^*_{t^\dagger})$.
5) At the end, the attacker outputs a bit b'.

In the view of a friend $f^\dagger \in \mathbf{F}_u$, the reasonable worst-case scenario is user u colludes with all other users, but not the service provider (otherwise it is impossible to get privacy anymore). The attack game is shown in Fig. 10, and it is an enhanced version of the attack game in Fig. 7 in the malicious model.

1) $(PK_x, SK_x)\ \forall x, PK_s, SK_s \xleftarrow{\$} C$
2) $(u, \mathbf{F}_u, f^\dagger \in \mathbf{F}_u, \mathbf{R}_x\ \forall x, SG) \xleftarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} \{0,1\}$; $\mathcal{A} \longleftarrow SK_x\ \forall x(x \neq f^\dagger, x \neq s)$
4) $\mathsf{RSProtocol}(\mathbf{R}_{f^\dagger}, SK_{f^\dagger}, SK_s, b;\ SK_x\ \forall x(x \neq f^\dagger, x \neq s), \mathbf{R}_x\ \forall x)$
5) $b' \longleftarrow \mathcal{A}$

Fig. 10: Worst-case Security for any Friend

1) The challenger generates the key pairs (PK_x, SK_x) for all x, and also generates (PK_s, SK_s).
2) The attacker generates the rating matrices $\mathbf{R}_x$ for all x, the social graph SG, and selects a user u and his friend set $\mathbf{F}_u$. The attacker also chooses $f^\dagger \in \mathbf{F}_u$.
3) The challenger generates a random bit b, and gives SK_x for all x except for SK_s and $SK_{f^\dagger}$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The challenger simulates the service provider and user $f^\dagger$, and the challenger simulates the rest. In the protocol execution, the challenger first samples the stranger set $\mathbf{T}_u$.
   - If $b = 0$ the challenger uses the original rating matrix $\mathbf{R}_{f^\dagger}$.
   - Otherwise it uses any $\mathbf{R}^*_{f^\dagger}$ such that $\mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u, \mathbf{R}_t\ \forall t \in \mathbf{T}_u) = \mathsf{Prediction}(\mathbf{R}_u, \mathbf{R}_f\ \forall f \in \mathbf{F}_u(f \neq f^\dagger), \mathbf{R}^*_{f^\dagger}, \mathbf{R}_t\ \forall t \in \mathbf{T}_u)$.
5) At the end, the attacker outputs a bit b'.

V. Construction and Analysis of Twitter Datasets

In this section, we construct two new datasets based on the MovieTweetings dataset [11] (abbreviated as MT dataset), which does not contain any friendship information. Based on the *"following"* activities in Twitter, we naturally introduce the concept of friendship as follows: if a user x follows user y then we say user x regards user y as a friend. Note that friendship is not guaranteed to be bi-directional, namely users x and y may not consider each other as friends at the same time.

A. Dataset Construction

MovieTweetings consists of ratings on movies that are extracted from tweets [11]. Such tweets originate from the social rating widget available in IMDb apps. To construct our new datasets, we use a snapshot of the MT dataset which contains 359908 ratings, 35456 users and 20156 items. Note that in this dataset each user has at least 1 rating. Since the MT dataset does not contain friendship information, we crawled the followees of each user ID recorded in it to create two new datasets with friendship information.

- In the FMT dataset, each user has at least 1 friend and each friend has at least 1 rating.
- In the 10-FMT dataset, each user has at least 10 friends and each friend has at least 10 ratings.

The specification for both datasets is in [34]. It is worth stressing that, in the new datasets, we only collect the Twitter users who have explicitly posted their movie ratings. In another word, the friend list of a user may be incomplete. The rating scale is regularized to [0,5]. We summarize some basic information of these datasets in the following table.

B. Hypothesis Validation

In order to test the folklore that friends share more similarities than strangers, we compute the Cosine similarities between users in the 10-FMT dataset and plot them in Fig. 11. Based on the fact that most ▶ dots are distributed above □ dots, we can conclude that friends typically share more similarities than strangers.

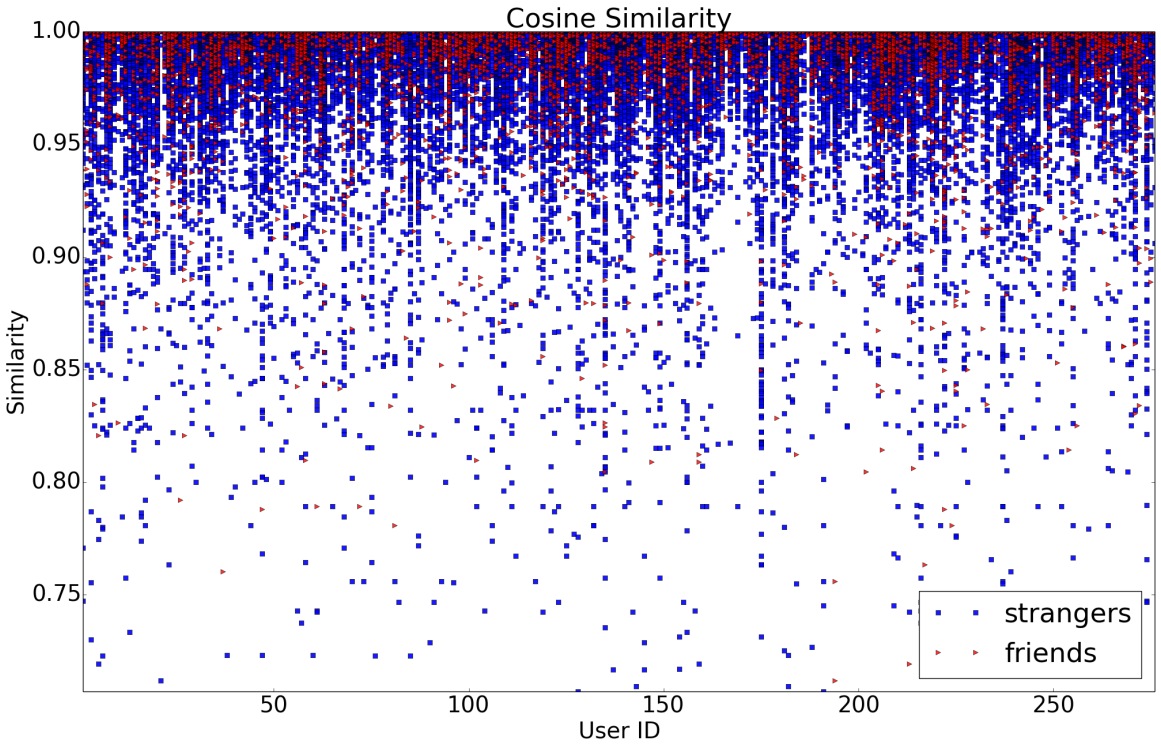

Fig. 11: Cosine Similarity of 10-FMT Dataset

	MT	FMT	10-FMT
Ratings Num	359908	211954	20316
Users Num	35456	17268	508
Items Num	20156	15682	3481
Matrix Density	0.050%	0.078%	1.15%
Min Ratings/User	1	1	10
Ave. Ratings/User	10	12	39
Max Ratings/User	856	856	448
Min Friends/User	–	1	10
Ave. Friends/User	–	6	26
Max Friends/User	–	282	109

C. Social Graph Characteristics

We summarize some simple facts of the 10-FMT dataset in Table II. Note that, besides friends, we also count the number of friends of friends (FoFs), which are used in designing the decentralized protocol in Section VII.

	Min Friend Num	Avg Friend Num	Max Friend Num
Friends	10	27	109
FoFs	10	203	329

TABLE II: Basic Facts

We then map the 10-FMT dataset into a directed graph in Fig. 12. In the graph, a node represents a user. If there is an arrow from user x to user y, then user x regards user y as a friend. It is clear that almost all users are connected in the social graph.

VI. Centralized Friendship-based Protocols

In this section, we first describe our centralized setting for recommender systems. Then, we describe two privacy-preserving protocols based on those from [29].

A. The Centralized Setting

A recommender system often has a large population of users, who may not know each other. Therefore, a common recommender service provider is always required to provide the communication platform for users to jointly run the recommender protocols. This leads us to assume a centralized setting for protocol design. We generally assume that there is a recommender service provider, which will maintain the social graph and mediate the executions of recommender protocols among users. The system structure is shown in Fig. 13.

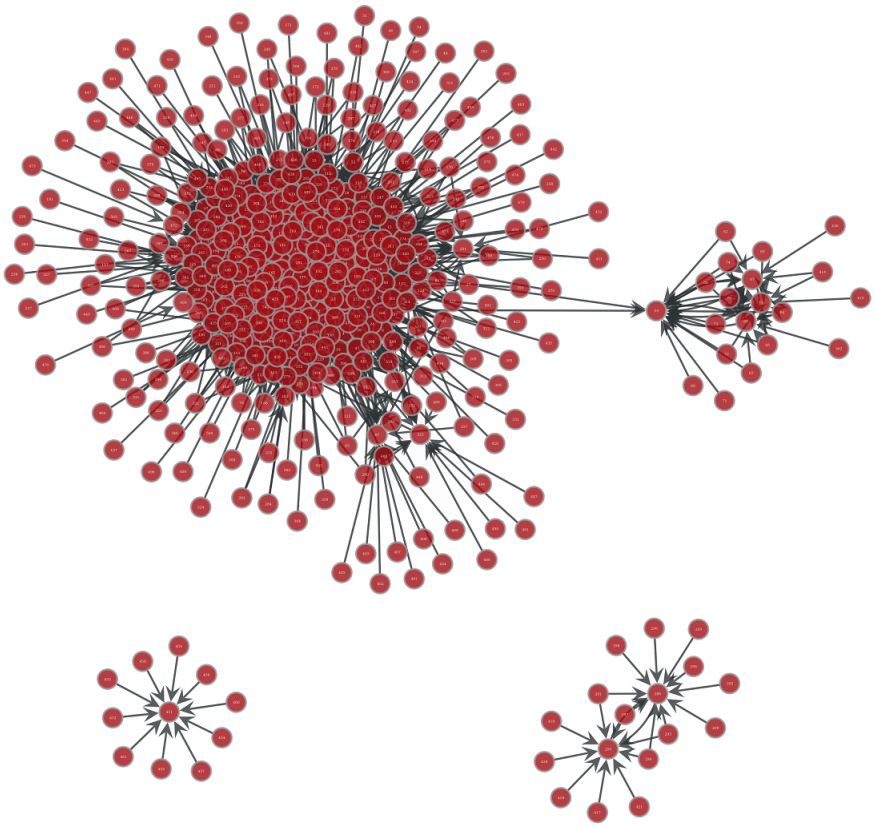

Fig. 12: Social Graph of 10-FMT Dataset

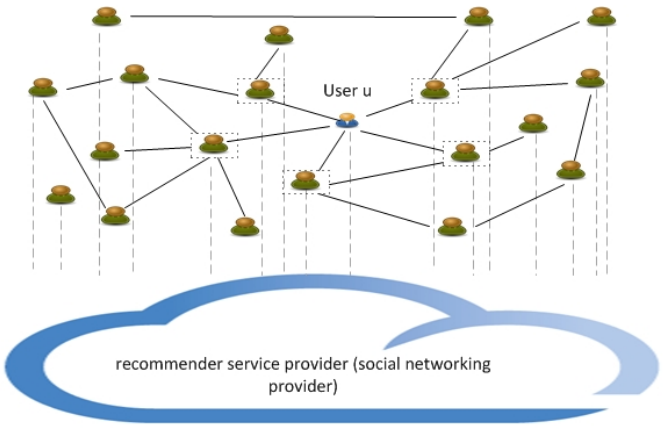

Fig. 13: System Structure in the View of User u

With respect to the tailored recommender algorithms in Section III, the global system parameters should be established in advance. Such parameters should include α, β which determine how a predicted rating value for user u is generated based on the inputs of friends and strangers, and they should also include the size of stranger set $\mathbf{T}_u$. In the initialization phase, user u generates his public/private key pair (PK_u, SK_u) for a SWHE scheme and sends PK_u to the server. We require that the SWHE scheme allows to encrypt negative integers. In addition, user u maintains a rating vector $\mathbf{R}_u$, his social graph, and assigns a weight $w_{u,f}$ to each of his friend $f \in \mathbf{F}_u$. All other users perform the same operations in this phase.

B. Centralized Single Prediction Protocol

When user u wants to test whether the predicted rating for an unrated item b is above a certain threshold τ (an integer) in his mind, he initiates the protocol in Fig. 14. Referring to the prediction algorithm from Section III-B, in stage 1 the service provider collects the inputs from the strangers in encrypted form according to Equation (2), while in stage 2 the service provider collects the inputs from the friends in encrypted form according to Equation (3). In stage 3, user u learns whether the prediction is above a threshold while the service provider learns nothing. In more detail, the protocol runs in three stages.

1) In the first stage, the participants interact as follows.
   a) User u generates a binary vector $\mathbf{I}_b$, which only has 1 for the b-th element, and sends the ciphertext $[\mathbf{I}_b]_u = \mathsf{Enc}(PK_u, \mathbf{I}_b)$ to the server. Let's assume $[\mathbf{I}_b]_u = ([\mathbf{I}_b^{(1)}]_u, \cdots, [\mathbf{I}_b^{(M)}]_u)$.
   b) The server first sends PK_u to some randomly chosen strangers, and see whether they want to participate in the computation.

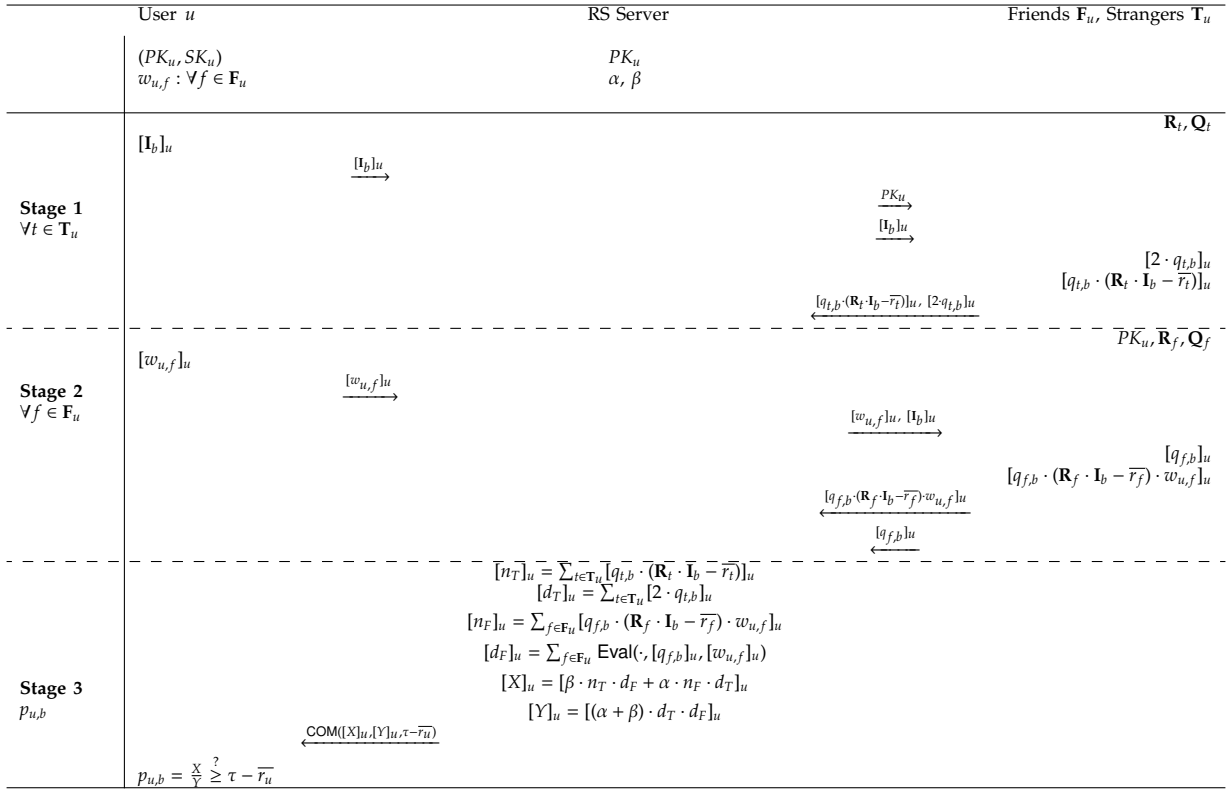

Fig. 14: Single Prediction Protocol

c) After the server has successfully found a viable stranger set $\mathbf{T}_u$, it forwards $[\mathbf{I}_b]_u$ to every user in $\mathbf{T}_u$.

d) With PK_u and $(\mathbf{R}_t, \mathbf{Q}_t)$, every user t from $\mathbf{T}_u$ can compute the following based on the homomorphic properties.

$$[2 \cdot q_{t,b}]_u = \sum_{1 \le i \le M} \mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, 2 \cdot q_{t,i}), [\mathbf{I}_b^{(i)}]_u)$$

$$[\mathbf{R}_t \cdot \mathbf{I}_b]_u = \sum_{1 \le i \le M} \mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, r_{t,i}), [\mathbf{I}_b^{(i)}]_u)$$

$$\begin{aligned}&[q_{t,b} \cdot (\mathbf{R}_t \cdot \mathbf{I}_b - \overline{r_t})]_u \\ =&\mathsf{Eval}(\cdot, [q_{t,b}]_u, \mathsf{Eval}(+, [\mathbf{R}_t \cdot \mathbf{I}_b]_u, \mathsf{Enc}(PK_u, -\overline{r_t})))\end{aligned}$$

2) In the second stage, the participants interact as follows.

a) For every friend $f \in \mathbf{F}_u$, user u sends the encrypted weight $[w_{u,f}]_u = \mathsf{Enc}(PK_u, w_{u,f})$ to the server.

b) The server sends $[w_{u,f}]_u$ and $[\mathbf{I}_b]_u$ to user f.

c) With PK_u, $[\mathbf{I}_b]_u$, $[w_{u,f}]_u$ and $(\mathbf{R}_f, \mathbf{Q}_f)$, user f can compute the following.

$$[q_{f,b}]_u = \sum_{1 \le i \le M} \mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, q_{f,i}), [\mathbf{I}_b^{(i)}]_u)$$

$$[\mathbf{R}_f \cdot \mathbf{I}_b]_u = \sum_{1 \le i \le M} \mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, r_{f,i}), [\mathbf{I}_b^{(i)}]_u)$$

$$\begin{aligned}[q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u =&\mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [q_{f,b}]_u, [w_{u,f}]_u), \\ &\mathsf{Eval}(+, [\mathbf{R}_f \cdot \mathbf{I}_b]_u, \mathsf{Enc}(PK_u, -\overline{r_f})))\end{aligned}$$

3) In the third stage, user u and the server interact as follows.

a) The server first computes $[n_T]_u$, $[d_T]_u$, $[n_F]_u$, $[d_F]_u$ as shown in Fig. 14, and then computes $[X]_u, [Y]_u$ as follows.

$$temp_1 = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [n_T]_u, [d_F]_u), \mathsf{Enc}(PK_u, \beta))$$

$$temp_2 = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [n_F]_u, [d_T]_u), \mathsf{Enc}(PK_u, \alpha))$$

$$[X]_u = \mathsf{Eval}(+, temp_1, temp_2)$$

$$[Y]_u = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [d_F]_u, [d_T]_u), \mathsf{Enc}(PK_u, \alpha+\beta))$$

Referring to Equations (2) and (3), we have $p^*_{u,b} = \overline{r_u} + \frac{n_T}{d_T}$ and $p^{**}_{u,b} = \overline{r_u} + \frac{n_F}{d_F}$. The ultimate prediction $p_{u,b}$ can be denoted as follows.

$$\begin{aligned}p_{u,b} &= \frac{\beta}{\alpha+\beta} \cdot p^*_{u,b} + \frac{\alpha}{\alpha+\beta} \cdot p^{**}_{u,b} \\ &= \overline{r_u} + \frac{\beta \cdot n_T \cdot d_F + \alpha \cdot n_F \cdot d_T}{(\alpha+\beta) \cdot d_T \cdot d_F} \\ &= \overline{r_u} + \frac{X}{Y}\end{aligned}$$

b) User u runs a comparison protocol COM with the server to learn whether $\frac{X}{Y} \ge \tau - \overline{r_u}$. Since

$X, Y, \tau - \overline{r_u}$ are integers, COM is indeed an encrypted integer comparison protocol: where user u holds the private key sk_u and τ, the server holds $[X]_u, [Y]_u$, and the protocol outputs a bit to user u indicating whether $X \geq (\tau - \overline{r_u}) \cdot Y$.

Compared to the protocol from [29], we have made the following changes: (1) given the fact that the service provider is semi-trusted, the strangers do not need to validate PK_u any more; (2) the strangers are chosen from the whole population while they are chosen from FoFs in [29]; (3) In Stage 1 and Stage 2, we simplify the computation of $[q_{t,b} \cdot (\mathbf{R}_t \cdot \mathbf{I}_b - \overline{r_t})]_u$ and $[q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$; (4) In stage 3, we do not require user u to send $[\overline{r_u}]_u$ to the server anymore, and this reduces the complexity from both sides; (5) we correct two errors in the computation of $[X]_u$ and $[d_F]_u$ in Fig. 3 from [29]. Note also that the prediction formula is also different from that in [29].

C. Centralized Top-n Protocol

When the active user u wants to figure out Top-n unrated items, he initiates the protocol in Fig. 15. This protocol shares the same design philosophy as that of single prediction protocol. The usage of matrix $\mathbf{M}_X$ in the random permutation of Stage 3 guarantees that the rated items will all appear in the end of the list after ranking. As a result, the rated items will not appear in the recommended Top-n items. In more detail, the protocol runs in three stages.

1) In the first stage, the participants interact as follows.
   a) The server sends PK_u to some randomly chosen strangers and see whether they want to participate in the computation. Suppose that the server has successfully found $\mathbf{T}_u$.
   b) With PK_u and $(\mathbf{R}_t, \mathbf{Q}_t)$, user $t \in \mathbf{T}_u$ can compute $[q_{t,b} \cdot (r_{t,b} - \overline{r_t})]_u = \mathsf{Enc}(PK_u, q_{t,b} \cdot (r_{t,b} - \overline{r_t}))$ and $[2 \cdot q_{t,b}]_u = \mathsf{Enc}(PK_u, 2 \cdot q_{t,b})$ for every $1 \leq b \leq M$. All encrypted values are sent back to the server.
2) In the second stage, the participants interact as follows.
   a) To every friend $f \in \mathbf{F}_u$, user u sends the encrypted weight $[w_{u,f}]_u = \mathsf{Enc}(PK_u, w_{u,f})$.
   b) With PK_u, $[w_{u,f}]_u$ and $(\mathbf{R}_f, \mathbf{Q}_f)$, user f can compute $[q_{f,b}]_u$ and
   $$\begin{aligned}&[q_{f,b} \cdot (r_{f,b} - \overline{r_f}) \cdot w_{u,f}]_u \\ =&\mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{r_f})), [w_{u,f}]_u)\end{aligned}$$
   for every $1 \leq b \leq M$. All encrypted values are sent back to the server.
3) In the third stage, user u and the server interact as follows.
   a) User u generates two matrices $\mathbf{M}_X, \mathbf{M}_Y$ as follows: (1) generate a $M \times M$ identity matrix; (2) randomly permute the columns to obtain $\mathbf{M}_Y$; (3) to obtain $\mathbf{M}_X$, for every b, if item b has been rated then replace the element 1 in b-th column with 0.
   $$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots\cdots & \cdots & \\ 0 & 0 & \cdots & 1 \end{bmatrix} \rightarrow \mathbf{M}_Y = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \cdots & \cdots\cdots & \cdots & \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$
   $$\rightarrow \mathbf{M}_X = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots\cdots & \cdots & \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$
   User u encrypts the matrices (element by element) and sends $[\mathbf{M}_X]_u, [\mathbf{M}_Y]_u$ to the server, which then proceeds as follows.
   i) The server first computes $[n_{T,b}]_u$, $[d_{T,b}]_u$, $[n_{F,b}]_u$, $[d_{F,b}]_u$, $[X_b]_u$, $[Y_b]_u$ for every $1 \leq b \leq M$ as shown in Fig. 15, in the same way as in the previous protocol in Fig. 14. Referring to Formula (4), we see that $\overline{r_u}$ appears in $p_{u,b}$ for every b. *For simplicity, we ignore this term when comparing the predictions for different unrated items.* With this simplification, the prediction $p_{u,b}$ can be denoted as follows.
   $$\begin{aligned} p_{u,b} &= \frac{\beta}{\alpha+\beta} \cdot \frac{n_{T,b}}{d_{T,b}} + \frac{\alpha}{\alpha+\beta} \cdot \frac{n_{F,b}}{d_{F,b}} \\ &= \frac{\beta \cdot n_{T,b} \cdot d_{F,b} + \alpha \cdot n_{F,b} \cdot d_{T,b}}{(\alpha+\beta) \cdot d_{T,b} \cdot d_{F,b}} \\ &= \frac{X_b}{Y_b} \end{aligned}$$
   ii) The server permutes the ciphertexts vector $(([X_1]_u, [Y_1]_u), ([X_2]_u, [Y_2]_u), \cdots, ([X_M]_u, [Y_M]_u))$ in an oblivious manner as follows.
   $$\begin{aligned}&([U_1]_u, [U_2]_u, \cdots, [U_M]_u) \\ =&[\mathbf{M}_X]_u \cdot ([X_1]_u, [X_2]_u, \cdots, [X_M]_u)^T\end{aligned}$$
   $$\begin{aligned}&([V_1]_u, [V_2]_u, \cdots, [V_M]_u) \\ =&[\mathbf{M}_Y]_u \cdot ([Y_1]_u, [Y_2]_u, \cdots, [Y_M]_u)^T\end{aligned}$$
   The multiplication between the ciphertext matrix and ciphertext vector is done in the standard way, except that the multiplication between two elements is done with $\mathsf{Eval}(\cdot,,)$ and the addition is done with $\mathsf{Eval}(+,,)$. Suppose item b has been rated before and $([X_b]_u, [Y_b]_u)$ is permuted to $([U_i]_u, [V_i]_u)$, then $U_i = 0$ since the element 1 in b-th column has been set to 0.
b) Based on some RANK protocol, the server sorts $\frac{U_i}{V_i}$ $(1 \leq i \leq |\mathbf{B}|)$ in the encrypted form. One straightforward way of constructing the RANK protocol is to combine an encrypted integer comparison protocol COM and any standard sorting algorithm. The COM protocol

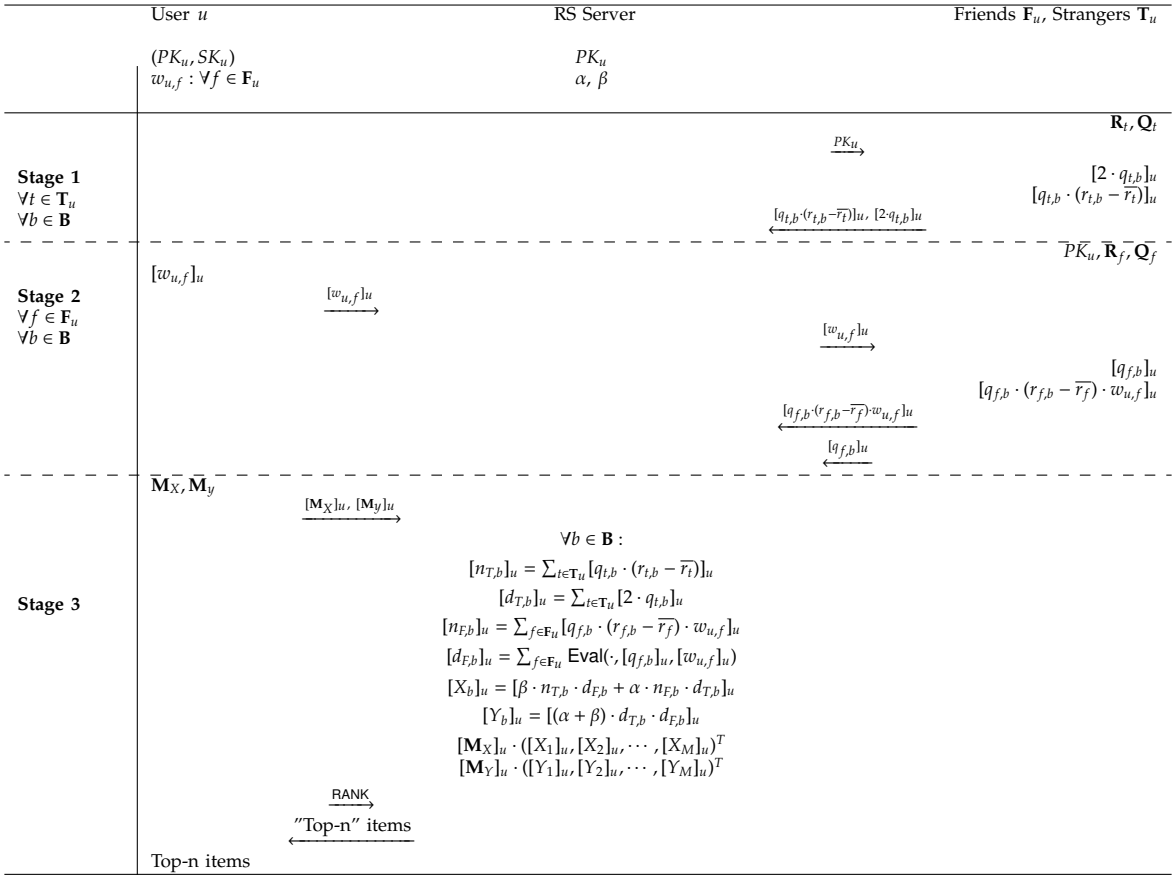

Fig. 15: Top-n Protocol

has slightly different semantics from that in the previous protocol in Section VI-B: user u has the private key and the service provider has two encrypted integers, at the end of the protocol the service provider learns the result.

c) After the ranking, the server sends the "Top-n" indexes (e.g. the permuted Top-n indexes) to user u, who can then recover the real Top-n indexes based on the permutation he has done.

Compared to the protocol from [29], we have made the following changes: (1) given the fact that the service provider is semi-trusted, the strangers does not need to validate PK_u any more; (2) the strangers are chosen from the whole population while they are chosen from FoFs in [29]; (3) we correct two errors in the computation of $[X_b]_u$ and $[d_{F,b}]_u$ in Fig. 4 from [29].

VII. Decentralized Friendship-based Protocol

In reality, *semi-honest* service provider is often viewed as a security weakness in protocol design. This motivates us to investigate privacy-preserving protocols in fully decentralized setting. Next, we first describe the setting and then present a decentralized single prediction protocol. Since we can extend the protocol to a Top-n variant in the same way as we have done in the centralized setting, we skip the details here.

A. The Decentralized Setting

For simplicity, we assume that users are uniquely identified in the recommender system, and they share their social graph with their friends. In the initialization phase, user u generates his public/private key pair (PK_u, SK_u) for a SWHE scheme. *For the purpose of enabling strangers to validate his public key, user u asks his friends to certify his public key and makes the certification information public as well.* In addition, user u maintains a rating vector $\mathbf{R}_u$, his social graph, and assigns a weight $w_{u,f}$ to each of his friend $f \in \mathbf{F}_u$. All other users perform the same operations in this phase.

Before going ahead, we want to point out that we choose a FoF as stranger in the following solution for the simplicity of description. In the view of user u, the topology is shown in Fig. 16. Due to the small world phenomenon, the population of FoFs can already be very large (see [2] and Section V-C).

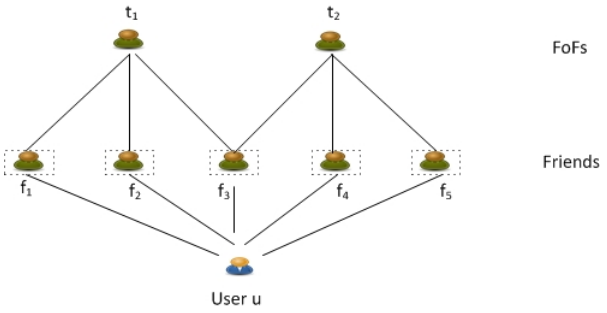

Fig. 16: Decentralized System Structure

B. Decentralized Single Prediction Protocol

Next, we describe a protocol for user u to check whether $p_{u,i} \geq \tau$ according to Formula (4) in Section III. It can be regarded as a decentralized version of the single prediction protocol from Section VI-B.

1) Based on the social graph (particularly his friend set F_u), user u chooses a stranger set T_u, consisting of his FoFs. He also chooses $t^* \in \mathsf{T}_u$. We further require that the every $f \in \mathsf{F}_u$ should have at least one friend in T_u.
2) User u generates a binary vector $\mathbf{I}_b$, which only has 1 for the b-th element, and broadcasts $[\mathbf{I}_b]_u = \mathsf{Enc}(PK_u, \mathbf{I}_b) = ([\mathbf{I}_b^{(1)}]_u, \cdots, [\mathbf{I}_b^{(M)}]_u)$ to his friends. He also sends $\mathsf{Enc}(PK_u, w_{u,f})$ to every user $f \in \mathsf{F}_u$.
3) With PK_u, $[\mathbf{I}_b]_u$, $[w_{u,f}]_u$ and $(\mathbf{R}_f, \mathbf{Q}_f)$, user f can compute the $[q_{f,b}]_u$ and $[q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$ in exactly the same way as in Section VI-B. User f then sends $[q_{f,b}]_u$ and $[q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$ to one of his friends in T_u. He also forwards $[\mathbf{I}_b]_u$ and PK_u to the chosen friend.
4) For any $t \in \mathsf{T}_u$, he should receive $[\mathbf{I}_b]_u$ and PK_u from at least one of his friend in F_u. If not, he can ask for such information from his friend. Then, he does the following.
   a) Validate PK_u.
   b) With PK_u and $(\mathbf{R}_t, \mathbf{Q}_t)$, every user t from $\mathbf{T}_u$ can compute $[q_{t,b}]_u$ and $[q_{t,b} \cdot (\mathbf{R}_t \cdot \mathbf{I}_b - \overline{r_t})]_u$ in exactly the same way as in Section VI-B.
   c) Suppose that user t has received $[q_{f,b}]_u$ and $[q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$ for $f \in \mathsf{F}_u^-$ where $\mathsf{F}_u^- \subseteq \mathsf{F}_u$. He computes $\sum_{f \in \mathbf{F}_u^-} [q_{f,b}]_u$ and $\sum_{f \in \mathbf{F}_u^-} [q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$.
   d) User t sends $[q_{t,b}]_u$, $[q_{t,b} \cdot (\mathbf{R}_t \cdot \mathbf{I}_b - \overline{r_t})]_u$, $\sum_{f \in \mathbf{F}_u^-} [q_{f,b}]_u$ and $\sum_{f \in \mathbf{F}_u^-} [q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$ to user t^*.
5) User t^* receives $[q_{t,b}]_u$, $[q_{t,b} \cdot (\mathbf{R}_t \cdot \mathbf{I}_b - \overline{r_t})]_u$, $\sum_{f \in \mathbf{F}_u^-} [q_{f,b}]_u$ and $\sum_{f \in \mathbf{F}_u^-} [q_{f,b} \cdot (\mathbf{R}_f \cdot \mathbf{I}_b - \overline{r_f}) \cdot w_{u,f}]_u$ from $t \in \mathbf{T}_u$. He then does the following.
   a) Compute $[n_T]_u, [d_T]_u, [n_F]_u, [d_F]_u$ and $[X]_u, [Y]_u$ in exactly the same way as in Section VI-B.
   b) Run a comparison protocol COM with user u for the latter to learn whether $\frac{X}{Y} \geq \tau - \overline{r_u}$.

C. Comparison to Centralized Protocol

In contrast to the centralized protocol from Section VI-B, the task of the semi-honest service provider is distributed to the "strangers", namely FoFs of user u. The overall computational complexity stays the same. The reason we have chosen the strangers to handle most of the computations is to reduce the complexity of the friends. In reality, the number of friends will be very limited, while the number of FoFs is much larger so that the chance a FoF is chosen is quite low.

If we assume trust can propagate through a chain of friends, then the strangers can be chosen more freely in the above solution. In comparison to the protocol from Section VI-B, this solution has the following advantages.

- The users do not need to semi-trust the service provider any more.
- User u can select the users (his friends and FoFs) to compute recommendations for himself. In order to do this, user u needs to maintain a social graph (at least his friends and FoFs).

However, it also has the following disadvantages.

- User u's FoFs need to perform more computations. Basically, the workload of the service provider has been shifted to them. This may become a heavy burden for the users.
- Users need to put more trust on their friends and FoFs, particularly on the user t^*. The users cannot leverage the service provider to blend their inputs anymore, and the trust has been shifted to user user t^*. In theory, this can be avoided by a secure multi-party computation protocol, but this will significantly increase the complexity.

Clearly, from the efficiency perspective, the centralized solution from Section VI-B is more realistic in practice. In order to reduce the trust on the service provider, we can (at least) add two layers of validations on its behaviors. One is that, before participating in the protocol execution, a stranger can ask the service provider to provide a chain of friends so that he can validate the public key PK_u. The other is that user u can ask the service provider to prove that it has performed the required operations honestly.

VIII. Accuracy Properties of the Proposed Protocols

In this section, we investigate the recommendation accuracy of the prediction algorithms from Section III, with respect to both centralized and decentralized settings where strangers are chosen differently therein. Because the 10-FMT dataset may be biased due to the fact that most of users don't post their movie ratings to Twitter, we also use MovieLens 100k dataset [26] with simulated friendships. Interestingly, the results align well in both datasets. In the experiments, we randomly split each data set into training set (80%) and testing set (20%). Note that in order to test all the users each time, instead of randomly splitting the original data sets in form of

triplets (user_id, item_id, rating), we randomly split each user's rating history into training set (80%) and testing set (20%). In each test, a user's friends are randomly selected from his friend-set, the strangers are also randomly chosen. The MAE values summarized in the following tables are the mean value of their corresponding 5-fold cross validation.

A. Accuracy in Centralized Setting

With respect to the 10-FMT dataset, the MAE of the revised TW algorithm from Section III-B is summarized in Table III. Due to the fact that a user has limited number of friends in the 10-FMT dataset, we only compute MAE up to 50 friends. The column denotes the possible values of $\frac{\alpha}{\alpha+\beta}$ and the row denotes the possible values of $(|\mathbf{F}_u|, |\mathbf{T}_u|)$, where strangers are randomly sampled. Lower MAE implies more accurate recommendations.

	0.5	0.6	0.7	0.8	0.9	1.0
(10, 10)	0.6178	0.6197	0.6192	0.6299	0.6362	0.6388
(20, 10)	0.6140	0.6168	0.6156	0.6204	0.6208	0.6291
(30, 10)	0.6076	0.6090	0.6094	0.6169	0.6234	0.6371
(40, 10)	0.6073	0.6066	0.6104	0.6150	0.6215	0.6300
(50, 10)	0.6066	0.6053	0.6095	0.6138	0.6199	0.6289

TABLE III: MAE on 10-FMT

With respect to the MovieLens 100k dataset, we define friends and strangers as follows. Given a user u, we first calculate the Cosine similarities with all other users and generate a neighborhood for user u. Then, we choose a certain number of users from the top-K_f most similar neighbors as the friends (In this paper, K_f = 250.), and randomly choose a certain number of users from the rest as strangers. The MAE of the revised TW algorithm from Section III-B is summarized in Table IV. According to the accuracy results by Lemire and Maclachlan (in Table 1 of [19] where the values are MAE divided by 4), their smallest MAE is $0.752 = 0.188 \times 4$. We can get similar or lower MAE when $|\mathbf{F}_u| \geq 70$ by adjusting $\frac{\alpha}{\alpha+\beta}$.

	0.5	0.6	0.7	0.8	0.9	1.0
(10, 10)	0.8195	0.8112	0.8074	0.8104	0.8157	0.8290
(20, 10)	0.8115	0.8002	0.7964	0.7937	0.8028	0.8086
(30, 10)	0.8046	0.7932	0.7866	0.7822	0.7874	0.7952
(40, 10)	0.8000	0.7852	0.7779	0.7739	0.7770	0.7834
(50, 10)	0.7943	0.7800	0.7693	0.7666	0.7658	0.7728
(60, 10)	0.7913	0.7757	0.7640	0.7593	0.7601	0.7636
(70, 10)	0.7888	0.7715	0.7601	0.7536	0.7530	0.7572
(80, 10)	0.7856	0.7682	0.7561	0.7482	0.7470	0.7484
(90, 10)	0.7830	0.7665	0.7527	0.7445	0.7424	0.7428
(100, 10)	0.7815	0.7626	0.7492	0.7398	0.7371	0.7386

TABLE IV: MAE on MovieLens 100k

From the numbers in Table III and Table IV, there is a general trend that MAE decreases when friends number increases. We plot some columns of both tables for a better illustration, shown in Fig 17.

When the numbers of friends and strangers are fixed, the contribution factor $\frac{\alpha}{\alpha+\beta}$ also plays a role in determining recommendation accuracy. We plot some rows of both tables for a better illustration, shown in Fig 18. The MAE decreases when $\frac{\alpha}{\alpha+\beta}$ increases (i.e. friends has more contribution) on the MovieLens 100k dataset, while the MAE slightly increases when $\frac{\alpha}{\alpha+\beta}$ grows higher than 0.6 on the 10-FMT dataset.

B. Accuracy in Decentralized Setting

For the decentralized setting, we compute the MAE on both datasets and present them in Table V and Table VI respectively. The MAE values are very close to those in Table III and Table IV, so that we can conclude that the recommendation accuracy is similar in both settings. It implies that sampling strangers from FoFs does not bring much accuracy gain with respect to both datasets.

	0.5	0.6	0.7	0.8	0.9	1.0
(10, 10)	0.6209	0.6168	0.6234	0.6289	0.6309	0.6465
(20, 10)	0.6135	0.6147	0.6164	0.6179	0.6275	0.6367
(30, 10)	0.6133	0.6085	0.6132	0.6188	0.6248	0.6309
(40, 10)	0.6124	0.6116	0.6110	0.6167	0.6233	0.6297
(50, 10)	0.6104	0.6084	0.6104	0.6147	0.6214	0.6301

TABLE V: MAE on 10-FMT

	0.5	0.6	0.7	0.8	0.9	1.0
(10, 10)	0.8181	0.8138	0.8132	0.8158	0.8188	0.8265
(20, 10)	0.8082	0.8034	0.7978	0.7994	0.8012	0.8123
(30, 10)	0.8026	0.7922	0.7879	0.7855	0.7885	0.7961
(40, 10)	0.7953	0.7862	0.7778	0.7763	0.7786	0.7826
(50, 10)	0.7917	0.7801	0.7726	0.7686	0.7688	0.7731
(60, 10)	0.7862	0.7747	0.7638	0.7625	0.7620	0.7664
(70, 10)	0.7854	0.7698	0.7604	0.7565	0.7532	0.7565
(80, 10)	0.7799	0.7663	0.7578	0.7502	0.7488	0.7489
(90, 10)	0.7781	0.7647	0.7524	0.7447	0.7407	0.7430
(100, 10)	0.7758	0.7603	0.7497	0.7406	0.7379	0.7377

TABLE VI: MAE on MovieLens 100k

C. Accuracy of JPH Prediction Algorithm

Since strangers are not considered in the JPH prediction algorithm from Section III-A, we compute the MAEs by only considering friends. For comparison, we assume all the friends are rational, and let $\frac{w_{u,f}+w_{f,u}}{2}$ equal to the Cosine similarity between user u and friend f. With respect to the 10-FMT and MovieLens 100k dataset, the MAE results are summarized in Table VII and VIII respectively. Clearly, their accuracy is much worse than our protocols which is mainly due to two reasons.

- JPH employs a very naive neighborhood-based method which can not capture users' rating preference. For example, some users prefer to give high ratings while some others lean to give low ratings.
- In reality, the data sets are very sparse and imbalanced. It may arise more serious cold-start problem to collaborative filtering techniques, including

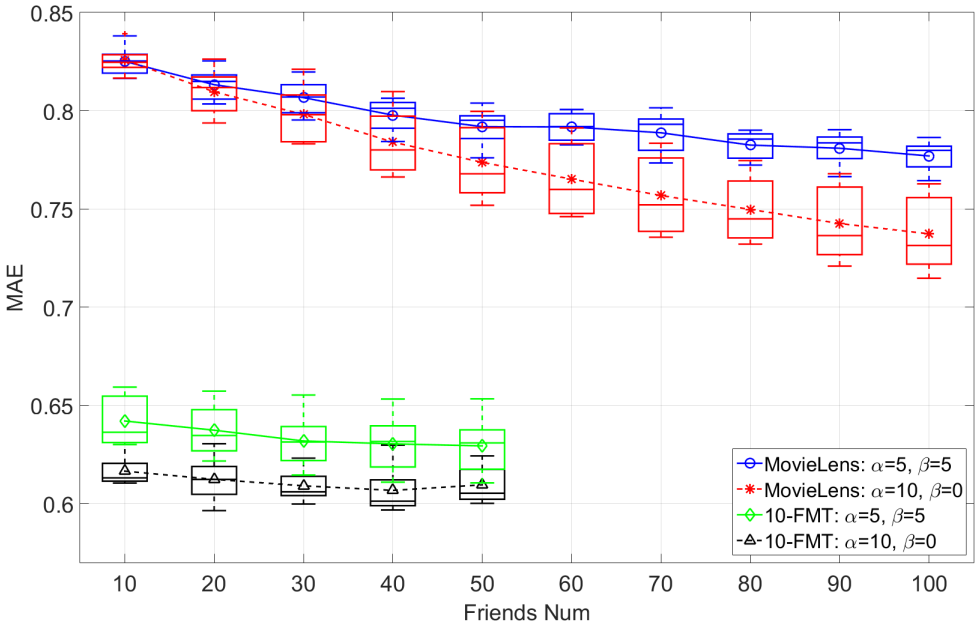

Fig. 17: MAE evaluation with different friends number.

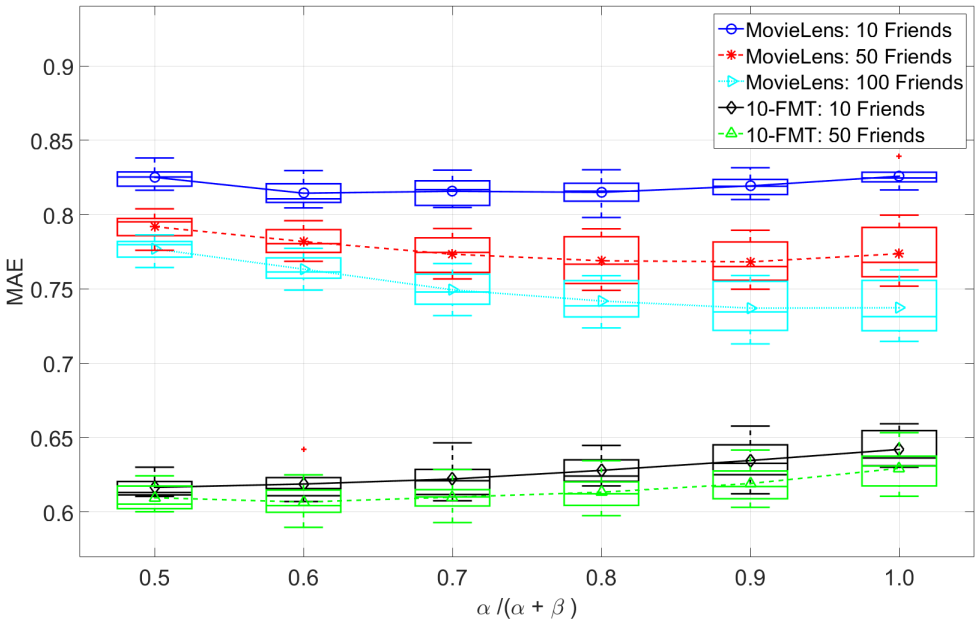

Fig. 18: MAE evaluation with different $\frac{\alpha}{\alpha+\beta}$.

neighborhood-based method, if only using friends' rating information for predication.

This validates our argument in Section III-A.

Friends Num	10	20	30	40	50
MAE	2.5961	2.2464	2.0690	1.9677	1.9072

TABLE VII: MAE of JPH on 10-FMT

Friends Num	20	40	60	80	100
MAE	1.9021	1.49451	1.2978	1.1825	1.1018

TABLE VIII: MAE of JPH on MoiveLens 100k

IX. Security Analysis of the Proposed Protocols

In the basic security model, the service provider is assumed to be semi-honest, which means it will follow

the protocol specification and does not participate in the protocol as a user. Moreover, a user trusts his friends to be semi-honest. As to communication channel among users, it is assumed that all communications are protected with respect to integrity and confidentiality (with forward secrecy). In the worst-case security model, it is assumed that some friends can be compromised.

The protocols from Section VI and VII are secure in both models based on the facts that all computations are done in the encrypted form under user u's public key and the comparison protocol is secure. It is worth noting that in these protocols the server does not need to generate any key pair for the SWHE scheme. As a result, the protocols are immune to key recovery attacks, in contrast to the JPH offline protocol [16], [29]. We skip the straightforward security reduction in this paper.

Next, we experimentally study the information leakages from recommendation outputs. We take the centralized protocols (where strangers are involved in the computation) as an example, and leave out the decentralized protocol which has similar results.

A. Inference from Outputs

In the security models [30], the potential information leakages from the output of a recommender system is not considered. Intuitively, this kind of leakage depends on the global parameters α, β and the sizes of $\mathbf{F}_u$ and $\mathbf{T}_u$. If $\frac{\alpha}{\alpha+\beta}$ gets larger or the size of $\mathbf{T}_u$ gets smaller, then the inputs from friends contribute more to the final output of user u. This will in turn make inference attacks easier against the friends but harder against the strangers. In the protocol design, we explicitly prevent user u from communicating with the strangers, therefore, user u will not trivially know whether a specific user t has been involved in the computation. The strangers are independently chosen in different protocol executions and the same stranger is unlikely to be involved in more than one executions, so that it is difficult for an attacker to leverage the accumulated information. Furthermore, we note the fact that there are many users in recommender systems but only 6 possible rating values for any item. This means that many users would give the same rating value $r_{t,b}$ for the item b. With respect to the single prediction protocol, even if $r_{t,b}$ is leaked, user u will not be able to link it to user t.

With respect the revised TW algorithm from Section III-B, a friend f's contribution to $p_{u,b}$ is protected by the inputs from users in $\mathbf{F}_u \backslash f$ and the strangers in $\mathbf{T}_u$. Similarly, a stranger t's contribution to $p_{u,b}$ is protected by the inputs from users in $\mathbf{F}_u$ and strangers in $\mathbf{T}_u \backslash t$. We perform some experiments to show how a single friend or stranger influences the predicted rating values. We use the both the 10-FMT and MovieLens 100k datasets, and set $\frac{\alpha}{\alpha+\beta} = 0.8$. For illustration purpose, we only consider two settings, namely $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (10, 10)$ and $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (30, 10)$.

Take the setting $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (10, 10)$ as an example, we perform the following experiment to test a friend's influence. In the experiment, we run 5-fold cross validation 50 times. In each 5-fold cross validation, we fix the friends of all users in the dataset by randomly selecting 11 friends for each user at the beginning, say each user has a fixed friend list L. Then for each user in the test set, the following procedure is carried out.

1) Randomly choose 10 strangers.
2) Randomly exclude 1 friend f_0 from the list L. Compute the predicted ratings of user u in the test set. Let the prediction vector be denoted as $\mathbf{P_0}$.
3) Randomly exclude 1 friend f_1 ($f_0 \neq f_1$) from the list L. Compute the predicted ratings of user u in the test set. Let the prediction vector be denoted as $\mathbf{P_1}$.
4) Compute the prediction difference vector as $\mathbf{P_0} - \mathbf{P_1}$.

Experiments for testing a stranger's influence and for the $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (30, 10)$ setting can be designed in a similar manner. After obtaining all prediction difference vectors $\mathbf{P_0} - \mathbf{P_1}$ in the experiments, we plot the frequency of all difference values in Fig. 19 for the 10-FMT dataset and in Fig. 20 for the MovieLens 100k dataset. From the figures, it is obvious that an individual's influence to the output is quite small. In particular, a friend's influence becomes smaller when the friend set becomes larger. Another observation is that a stranger's influence is much smaller than a friend, but it stays almost the same when the friend set becomes larger.

X. Computational Complexity Analysis

In this section, we investigate the computational complexities of the protocols from Section VI-B and VI-C. Since it is easy to infer the complexity of the decentralized protocol from the centralized one, we skip the details.

A. Asymptotic Analysis

With respect to the computational complexity of the proposed protocols, we first count the number of different computations required. For the single prediction protocol from Section VI-B, the numbers of SWHE-related operations are listed in Table IX. In addition, there is one comparison COM.

	Enc	Eval$(+,,)$	Eval$(\cdot,,)$
Friend	$2M+1$	$2M-1$	$2M+2$
Stranger	$2M+1$	$2M-1$	$2M+1$
Server	4	$2\|\mathbf{T}_u\|+2\|\mathbf{F}_u\|-3$	$\|\mathbf{F}_u\|+6$
User u	$M+\|\mathbf{F}_u\|$	0	0

TABLE IX: Complexity of Single Prediction Protocol

For the Top-n protocol from Section VI-C, the numbers of SWHE-related operations are listed in Table X. In addition, if we instantiate the RANK protocol with the well-known Heapsort algorithm, user u and the service provider needs to perform a COM protocol $O(M \log M)$ times.

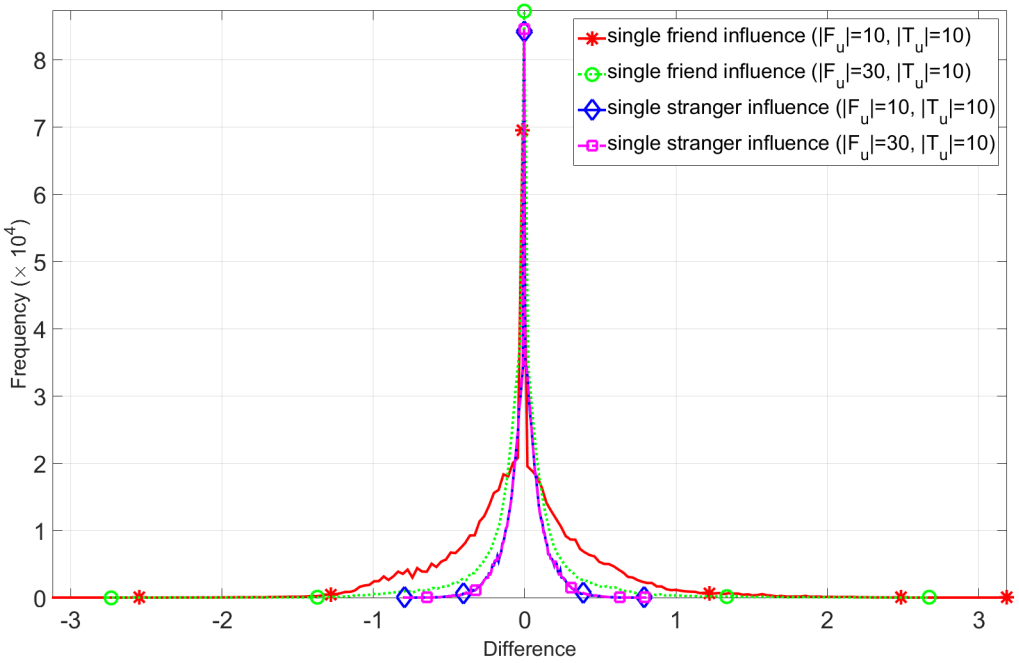

Fig. 19: Single Friend (Stranger) Influence on 10-FMT Dataset. Statistically, ≥87% and ≥91% differences fall into range $[-10^{-5}, 10^{-5}]$ w.r.t single friend influence testing and single stranger influence testing respectively. For clearly presenting those differences that fall out of range $[-10^{-5}, 10^{-5}]$ and keeping the histogram structure, ≈98% of the differences that fall into range $[-10^{-5}, 10^{-5}]$ have been removed.

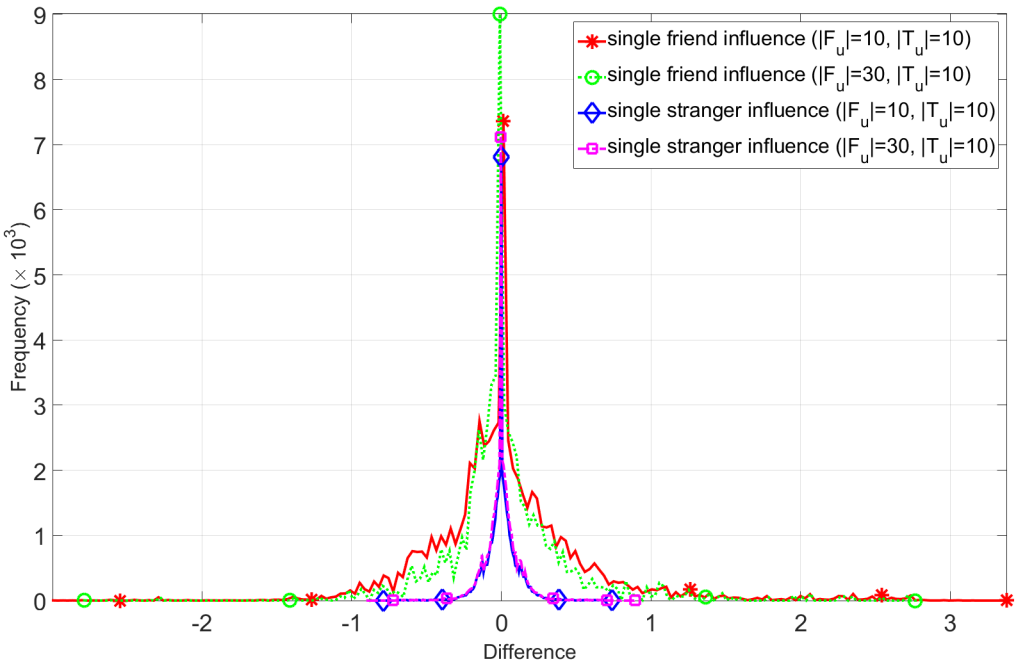

Fig. 20: Single Firend (Stranger) Influence on MovieLens 100k Dataset. Statistically, ≥89% and ≥81% differences fall into range $[-10^{-5}, 10^{-5}]$ w.r.t single friend influence testing and single stranger influence testing respectively. For clearly presenting those differences that fall out of range $[-10^{-5}, 10^{-5}]$ and keeping the histogram structure, ≈98% of the differences that fall into range $[-10^{-5}, 10^{-5}]$ are removed.

	Enc	Eval(+,,)	Eval(·,,)
Friend	$2M$	0	M
Stranger	$2M$	0	0
Server	3	$2(\|\mathbf{T}_u\|+\|\mathbf{F}_u\|+M)M-5M$	$(2M+6+\|\mathbf{F}_u\|)M$
User u	$2M^2+\|\mathbf{F}_u\|$	0	0

TABLE X: Complexity of Top-n Protocol

B. Implementation Results

We instantiate a COM protocol based on that of Veugen [31] and evaluate its performance. In addition to the SWHE scheme based on YASHE [6], the protocol also relies Goldwasser-Micali scheme [15]. In both schemes, we set the bit-length of the prime number to be 512. We implement the Goldwasser-Micali scheme, which has the timing cost for Enc (1.5 μs), Dec (4.5 μs), based on an Intel(R) Core(TM) i7-5600U CPU 2.60GHz. In executing the COM protocol, the computation time for the client and the server is roughly 0.45 ms and 2.82 ms respectively. We adopt the MovieLens 100k dataset where $M = 1682$ and set $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (70, 10)$. We use the Microsoft SEAL library [12] based on YASHE scheme. The timing information of the SEAL lib is Enc (42 ms), Dec (41 ms), Eval(·,,) (305 ms), Eval(+,,) (85 μs). The timing information of our protocols is shown in Table XI, and the source code is in [33].

	Friend	Stranger	Server	User u
Single	1.12	1.00	0.72	74.17
Top-n	141.22	140.55	1726446	236424

TABLE XI: Timing Numbers (Seconds)

Regardless the resource-constrained testing environment, it is clear that the Top-n protocol is very inefficient. The complexity mainly comes from the fact that we want to restrict user u to only learn the Top-n recommendations and prevent the server from learning any information. As such, there are two possible directions to relax the security guarantee and get better efficiency.

- One is to let user u learn more information (denoted as Relax-1 in Table XII). Referring to the protocol specification in Section VI-C, in stage 3, user u does not need to generate $\mathbf{M}_X, \mathbf{M}_Y$ and the server does not need to compute $([U_1]_u, [U_2]_u, \cdots, [U_M]_u)$ and $([V_1]_u, [V_2]_u, \cdots, [V_M]_u)$. There is no need to perform the ranking, the server just sends $[X_b]_u, [Y_b]_u$ for every $1 \le b \le M$ to user u, who can decrypt these ciphertexts and obtain the Top-n recommendations.
- The other is to let the server learn how many items user u has rated (denoted as Relax-2 in Table XII). In addition, we need to assume that the strangers will not collude with the server. Referring to the protocol specification in Section VI-C, in stage 1 and 2, user u generates a random permutation for the items in the item set and share the permutation information with the friends and strangers. In stage 3, user u does not need to generate $\mathbf{M}_X, \mathbf{M}_Y$ and the server does not need to compute $([U_1]_u, [U_2]_u, \cdots, [U_M]_u)$ and $([V_1]_u, [V_2]_u, \cdots, [V_M]_u)$. In stage 3, user u tells the server which items has been rated (the indices of these items have been permuted), and they interactively perform the ranking for the unrated items in the encrypted form as before.

	Friend	Stranger	Server	User u
Relax-1	141.22	140.55	1562	141.58
Relax-2	141.22	140.55	1610	10.46

TABLE XII: Timing Numbers (Seconds)

XI. Conclusion

In the paper, we have refined the protocols from [29] and proposed a new decentralized single prediction protocol. We have also provided detailed analysis to recommendation accuracy, inference attacks, and computational complexities. The idea of introducing randomly selected strangers to prevent information leakages from the output share some similarity with the differential privacy based approach [21], [35] and the differential identifiability approach [18]. A more rigorous comparison remains as an interesting future work, particularly in the line of the works from [5], [13]. With respect to accuracy analysis, it is an interesting future work to perform a study on an unbiased real-world dataset.

Acknowledgements

Both authors are supported by a CORE (junior track) grant from the National Research Fund, Luxembourg. Qiang Tang is also partially supported by an internal project from University of Luxembourg. The authors would like to thank Husen Wang from University of Luxembourg for conducting the complexity evaluation.

References

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

[2] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42, 2012.

[3] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 107–114. ACM, 2015.

[4] M. Beye, A. Jeckmans, Z. Erkin, Q. Tang, P. Hartel, and I. Lagendijk. *Social Media Retrieval*, chapter Privacy in Recommender systems, pages 263–281. Springer, 2013.

[5] R. Bhaskar, A. Bhowmick, V. Goyal, S. Laxman, and A. Thakurta. Noiseless database privacy. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 215–232. Springer, 2011.

[6] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.

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

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

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

[10] J. F. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.

[11] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Proceedings of Workshop on Crowdsourcing and Human Computation for Recommender Systems*, pages 84–89, 2013.

[12] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. https://www.microsoft.com/en-us/research/publication/manual-for-using-homomorphic-encryption-for-bioinformatics/.

[13] Y. Duan. Privacy without noise. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1517–1520. ACM, 2009.

[14] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[15] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM, 1982.

[16] A. Jeckmans, A. Peter, and P. H. Hartel. Efficient privacy-enhanced familiarity-based recommender system. In J. Crampton, S. Jajodia, and K. Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 400–417. Springer, 2013.

[17] S. K. Lam, D. Frankowski, and J. Riedl. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. In G. Muller, editor, *Emerging Trends in Information and Communication Security*, volume 3995 of *LNCS*, pages 14–29. Springer, 2006.

[18] J. Lee and C. Clifton. Differential identifiability. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1041–1049. ACM, 2012.

[19] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman, editors, *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, pages 471–475. SIAM, 2005.

[20] Z. Liu, Y. X. Wang, and A. Smola. Fast differentially private matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 171–178. ACM, 2015.

[21] F. McSherry and I. Mironov. 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*, pages 627–636, 2009.

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

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

[24] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 625–628, 2003.

[25] N. Ramakrishnan, B.J. Keller, B.J. Mirza, and A. Y. Grama. Privacy risks in recommender systems. *Internet Computing, IEEE*, 5:54–63, 2001.

[26] GroupLens Research. http://grouplens.org/datasets/movielens/.

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

[28] Q. Tang. Cryptographic framework for analyzing the privacy of recommender algorithms. In *2012 International Symposium on Security in Collaboration Technologies and Systems (CTS 2012)*, pages 455–462, 2012.

[29] Q. Tang and J. Wang. Privacy-preserving context-aware recommender systems: Analysis and new solutions. In G. Pernul, P. Y. A. Ryan, and E. R. Weippl, editors, *Computer Security - ESORICS 2015*, volume 9327 of *LNCS*, pages 101–119. Springer, 2015.

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

[31] T. Veugen. Comparing encrypted data. http://bioinformatics.tudelft.nl/sites/default/files/Comparing2011.

[32] T. Veugen, R. de Haan, R. Cramer, and F. Muller. 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.

[33] H. Wang. https://github.com/lux-jwang/Experiments/tree/master/code2016/tdsc/cryptonbm.

[34] J. Wang. https://github.com/lux-jwang/Experiments/tree/master/code2016.

[35] Y. Wang, S. E. Fienberg, and A. J. Smola. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37, pages 2493–2502, 2015.

[36] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft. Blurme: inferring and obfuscating user gender based on ratings. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12*, pages 195–202. ACM, 2012.

[37] I. Yakut and H. Polat. Arbitrarily distributed data-based recommendations with privacy. *Data & Knowledge Engineering*, 72(0):239 – 256, 2012.

[38] S. Zhang, J. Ford, and F. Makedon. Deriving private information from randomly perturbed ratings. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 59–69. SIAM, 2006.