Privacy-preserving Friendship-based Recommender Systems

Qiang Tang and Jun Wang

Abstract—Today, recommender systems are playing an indispensable role in our daily life. However, nothing is for free – such systems have also upset the society with severe privacy concerns. In this paper, we first revisit the concept of computing recommendations based on inputs from both a user’s friends and a set of randomly chosen strangers. We propose two security models to formalize information leakages in recommender systems. We then clarify two protocols by Tang and Wang at ESORICS 2015, analyse their security in our security models, and investigate their performances according newly-constructed Twitter datasets and MovieLens 100k dataset. Our experiments show that the single prediction protocol is efficient and can be considered practical in reality. We finally propose a new decentralized single prediction protocol and compare it to the centralized (clarified) protocol.

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 bookstore 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.

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. Unsurprisingly, the abundance of recommender systems has intrigued a lot of privacy concerns. In practice, advanced recommender systems collect more personal information (e.g. context information such as location and social surroundings) than ratings, and they inevitably cause more severe privacy concerns. Privacy issues in recommender systems have been surveyed in [6], [28], [47]. The most widely-recognized privacy concern is about the fact the service provider has full access to all users’ inputs (e.g. which items are rated and the corresponding ratings). The other less well-known yet equally serious privacy concern is that the outputs from a recommender system can also lead to privacy breaches against innocent users. Ten years ago, Kantarcioglu, Jin and Clifton expressed this concern for general data mining services [27]. Recently, Calandrino et al. [12] showed 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. In practice, advanced recommender systems collect more personal information (e.g. context information such as location and social surroundings) than ratings, and they inevitably cause more severe privacy concerns. Existing privacy-protection solutions can be generally divided into two categories.

• The cryptographic solutions (e.g. [1], [13], [14], [17], [26], [37], [51], [54]) 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. Moreover, many solutions (e.g., [37, 54]) introduce additional semi-trusted servers which are difficult to be instantiated in reality. It is worth noting that solutions for partitioned datasets [24], [25], [42], [43], [46], [59] could suffer from similar problems.

- The data-obfuscation solutions (e.g., [8], [36], [39], [40], [41], [44], [45], [50], [58]) 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. [60] showed how to recover perturbed ratings in the solutions from [40], [44]. With respect to privacy guarantees, an exception is the differential privacy based approach from [5], [33], [36] 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).

### System Robustness

In practice, recommender systems are often open systems. It means that there is normally no (strong) authentication for the end users and their inputs to the system. This leaves an open door for attackers to manipulate the system’s outputs to suit their needs (e.g. to promote their own products or to cause disruption in the service). Such robustness issues can in turn cause the recommender system to become unreliable and untrustworthy, resulting in user dissatisfaction. Lam and Riedl [30] investigated the concept of shilling attacks, where a malicious company can lie to the recommender system to have their products recommended more often than those from their competitors.

O’Mahony [38] proposed a framework for robustness in the presence of (maliciously injected) noises. Following [30], [38], a large number of following-up works have been devoted to formulating robustness properties, detecting attacks, and designing countermeasures. Massa and Avesani [34], [35] proposed the concept of trust-aware collaborative filtering recommender systems. In their solutions, trust metrics between users are proposed and taken into account when computing recommendations. By putting more weights on the “trusted” friends, attacks can be mitigated to some extent. Lam et al. [29] gave a discussion between data privacy and robustness properties. Seminario and Wilson [49] investigated the tradeoff between accuracy and robustness properties. The authors in [3, 23] gave very comprehensive survey on the robustness issues.

### B. Our Contribution

Friendships between friends are very valuable information for recommender systems. In this paper, we demonstrate that we can leverage such information to build accurate and efficient recommender systems with strong data privacy and robustness properties. However, potential privacy concerns between friends make the application of such information quite tricky. For a user Alice, revealing her inputs to her friend might be more uncomfortable than to a stranger. This requires us to carefully craft the security model and precisely define the security guarantees. We improve on the literature and have the following contributions.

- We revisit the concept of friendship-based recommender system from [52], where the recommendations for a user Alice are computed mainly based on the inputs from Alice’s friends. One notable feature of the concept is that inputs from some randomly-chosen strangers are also used to compute recommendations for Alice, for the purpose of protecting the privacy of Alice’s friends and mitigate the cold boot problem of recommender systems. We newly formalize several security properties for two scenarios. In the average-case scenario, we assume both friends and the service provider are semi-honest. In the worst-case scenario, we assume friends may also be compromised.
- We clarify the single prediction and Top-n protocols from [52] and show that they achieve provable security in our security models. We perform comprehensive analysis on the performances of the proposed protocols. Experiments are carried out on both standard dataset with simulated friendships and newly-constructed Twitter datasets with real friendships. The construction and analysis of Twitter datasets are of independent interests for the community. Our experimental results have validated the hypothesis that we can build accurate and efficient privacy-preserving recommender systems based on friendships.
- We propose a new decentralized single prediction protocol, in which we get rid of the service provider and base the security solely on the semi-honest assumption among friends. We further show the tradeoffs between centralized and decentralized solutions.

### 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 revisit the friendship-based recommender system structure from [52] and present the new security models. In Section IV, we revisit the single prediction and Top-n protocols from [52]. In Section V, we construct and analyse two Twitter datasets. In Section VI, we present security and accuracy analysis for the proposed protocols. In Section VII, we present a
decentralized privacy-preserving single prediction protocol. In Section VIII, we conclude the paper.

II. Preliminary

When $X$ is a set, $x \sim X$ means that $x$ is chosen from $X$ uniformly at random, and $|X|$ means the size of $X$. If $\chi$ is a distribution, then $s \sim \chi$ means that $s$ is sampled according to $\chi$. 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 $\mathbf{B} = (1, 2, \cdots, b, \cdots, M)$, 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.

| User 1 ($R_1$) | Item 1 | \cdots | Item b | \cdots | Item M |
|---------------|--------|--------|--------|--------|
| $r_{1,1}$     | \cdots | $r_{1,b}$ | \cdots | $r_{1,M}$ |
| User 2 ($R_2$) | Item 1 | \cdots | Item b | \cdots | Item M |
| $r_{2,1}$     | \cdots | $r_{2,b}$ | \cdots | $r_{2,M}$ |
| \vdots        | \vdots | \vdots | \vdots | \vdots |
| User N ($R_N$) | Item 1 | \cdots | Item b | \cdots | Item M |
| $r_{N,1}$     | \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.

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

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 $\bar{r}_x$ to denote user $x$’s average rating, namely $\frac{1}{\sum_{b=1}^{M} q_{x,b}} \sum_{b=1}^{M} r_{x,b}$.

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

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

where $\Gamma$ 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 [19], many somewhat homomorphic encryption (SWHE) schemes have been proposed (e.g. the BV scheme [11], BGV scheme [10], FV scheme [18], YASHE scheme [9]). A SWHE scheme can be described by four algorithms ($\text{Keygen}$, $\text{Enc}$, $\text{Dec}$, $\text{Eval}$), where the $\text{Eval}$ algorithm can only be executed for a limited number of times.

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

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, $\text{Enc}(PK_u, m)$ represents the probabilistic output of running $\text{Enc}$ for the message $m$. When $m$ is a vector of messages, we use $\text{Enc}(PK_u, \mathbf{m})$ to denote the vector of ciphertexts, where encryption is done for each element independently. We use the notation $\sum_{1 \leq b \leq N} [m_b]_u$ to denote the result of sequentially applying $\text{Eval}(\cdot, \cdot)$ to the ciphertexts.

III. Tailored Recommender Algorithms

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. A visualization of the system structure is shown in Fig. 1.

![Fig. 1: System Structure in the View of User u](image-url)

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. In practice, such a service provider may also provide additional inputs (e.g. users’ behavioral information) to the recommender algorithms for more accurate recommendations. From the security perspective, the service provider enables us to build efficient privacy-preserving recommender systems. Regardless all the benefits, we can replace this third party with a Secure Multi-party Computation (SMC) protocol, and we show such an example in Section VII.
In the following, we first describe a tailored approach to compute recommendations, and then present our security definitions.

### A. Computing Predicted Ratings

Researchers design recommender systems by leveraging existing social relationships of end users, e.g., the trust-aware protocols [34, 35] and the JPH protocols from [25]. Given a user \( u \), let his friend set be \( F_u \). The prediction of JPH protocols is computed as follows, where \( w_{u,f} \) and \( w_{f,u} \) are the weights that users \( u \) and \( f \) assign to each other.

\[
\begin{align*}
    p_{u,b} &= \frac{\sum_{f \in F_u} q_{f,b} \cdot r_{f,b} \cdot \left( \frac{w_{u,f} + w_{f,u}}{2} \right)}{\sum_{f \in F_u} q_{f,b} \cdot \left( w_{u,f} + w_{f,u} \right)} \\
       &= \frac{\sum_{f \in F_u} r_{f,b} \cdot (w_{u,f} + w_{f,u})}{\sum_{f \in F_u} q_{f,b} \cdot (w_{u,f} + w_{f,u})} \\
&= \frac{\sum_{f \in F_u} r_{f,b} \cdot (w_{u,f} + w_{f,u})}{\sum_{f \in F_u} q_{f,b} \cdot (w_{u,f} + w_{f,u})} \tag{1}
\end{align*}
\]

In [25], the authors only discussed the security properties of their solutions without touching upon the performance. 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 predicted value from Equation (1) may not be very accurate (cold start problem). In Section VI, we back up this argument with two experiments. Besides the potential performance issues, the JPH protocols may incur privacy concerns due to the fact that recommendations are computed solely based on the friends’ inputs. In [52], Tang and Wang have shown that the private information of user \( u \)’s friends might be leaked through user \( u \)’s outputs.

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 [32]. Let the stranger set be \( T_u \) the predicted value \( p^*_{u,b} \) for an unrated item \( b \) is computed as follows.

\[
\begin{align*}
    p^*_{u,b} &= \bar{r}_u + \frac{\sum_{t \in T_u} q_{t,b} \cdot (r_{t,b} - \bar{r}_t)}{\sum_{t \in T_u} q_{t,b}} \\
&= \bar{r}_u + \frac{\sum_{t \in T_u} q_{t,b} \cdot (r_{t,b} - \bar{r}_t)}{\sum_{t \in T_u} q_{t,b}} \tag{2}
\end{align*}
\]

In practice, the similarity between friends means that they tend to prefer to 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 \( \bar{r}_u \) reflects the user’s rating style and \( \frac{\sum_{t \in T_u} q_{t,b} \cdot (r_{t,b} - \bar{r}_t)}{\sum_{t \in T_u} q_{t,b}} \) 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 \( 0 \leq \rho \leq 1 \). Due to the fact that cryptographic primitives are normally designed for dealing with integers, we rephrase the formula as follows, where \( \alpha, \beta \) are two integers.

\[
\begin{align*}
    p_{u,b} &= \frac{\beta}{\alpha + \beta} \cdot p^*_{u,b} + \frac{\alpha}{\alpha + \beta} \cdot p_{u,b} \tag{4}
\end{align*}
\]

It is worth noting that the prediction \( p_{u,b} \) is not deterministic because we assume the stranger set \( T_u \) is randomly chosen for the computation.

### B. 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 III-C.

Let the users be indexed by an integer \( x \geq 1 \). We assume user \( x \) has a public/private key pair \( (PK_x, SK_x) \) and a rating vector \( R_x \), 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 RSProtocol and let it output \( \text{Prediction}(R_{u,f}, R_f, \forall f \in F_u, R_f, \forall t \in T_u) \) to user \( u \) and output nothing to others. Note that the service provider will randomly set \( T_u \) in the execution of RSProtocol.

As a standard cryptographic practice, every security property is modeled as a game between a challenger \( C \) and an attacker \( A \). In the game, the challenger \( C \)
In the view of a stranger from $F_u$, user $u$ may try to learn his private information (even colluding with $u$’s friends in $F_u$) by acting maliciously. The attack game is shown in Fig. 4, and the detailed description follows.

1) The challenger generates the key pairs $(PK_u, SK_u)$ for all $u$, and also generates $(PK_v, SK_v)$.
2) The attacker generates the rating matrices $R_v$ for all $x$, the social graph $SG$, and selects a user $u$ and his friend set $F_u$.
3) The challenger generates a random bit $b$, and gives $SK_v$ for all $x \notin F_u$ except for $SK_u$ and $SK_v$ to the attacker.
4) The challenger and the attacker run the RSProtocol protocol. The challenger simulates user $u$ from $F_u$ and the service provider, and the attacker simulates the rest. In the protocol execution, if $b = 0$ the challenger uses the original rating matrices $R_u$ and $R_f$ for all $f \in F_u$ otherwise it uses some 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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$

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. 3, and the detailed description follows.

1) $(PK_u, SK_u) \forall x, PK_u, SK_u \xrightarrow{\$} C$
2) $(u, F_u, R_v \forall x, SG) \xrightarrow{\$} \mathcal{A}$
3) $b \xleftarrow{\$} [0, 1]; \mathcal{A} \leftarrow SK_u \forall x \notin F_u(x \neq u \land x \neq s)$
4) $RSProtocol(R_v, SK_v, R_f, SK_f) \forall f \in \begin{cases} F_u, SK_u, b; & SK_v \forall x \notin F_u(x \neq u \land x \neq s), R_v \forall x) \\ b' \leftarrow \mathcal{A} \end{cases}$
In the view of user $u$, with respect to the robustness of a recommender system, we only consider the scenario where a user tries to provide ill-formed inputs. The only threat is from the randomly-chosen strangers. Informally, the robustness property can be evaluated by the probability that a stranger’s cheating is detected by the service provider. Referring to the formula (2) in Section III-A, a stranger $t'$’s input is $q_{t,b} \cdot (r_{t,b} - \overline{r}_t)$ and $q_{t,b}$ (in the encrypted form). Formally, robustness is defined as: for any $t$, if the input is not in either of the following forms then it can be detected with overwhelming probability:

- $q_{t,b} \cdot (r_{t,b} - \overline{r}_t) = 0$ and $q_{t,b} = 0$.
- $-5 \leq q_{t,b} \cdot (r_{t,b} - \overline{r}_t) \leq 5$ and $q_{t,b} = 1$.

Threat from a semi-honest friend. In the view of user $u$, we assume that none of his friends will collude with the recommender server or another party to breach his privacy and 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. 5, and the detailed description follows.

1) (PK$_x$, SK$_x$) $\forall x$, PK$_u$, SK$_u$ $\leftarrow$ C
2) ($u$, F$_u$, $f^t \in$ F$_u$, R$_t$, $\forall x$, SG) $\leftarrow$ $\mathcal{A}$
3) $b \leftarrow$ [0, 1]; $\mathcal{A} \leftarrow$ SK$_{f^t}$
4) RSPProtocol($R_v$, SK$_v$, $\forall x \neq f^t$, $b$; SK$_{f^t}$, R$_v$, $\forall x$)
5) $b' \leftarrow$ $\mathcal{A}$

Fig. 5: Security against any Semi-honest Friend

1) The challenger generates the key pairs (PK$_x$, SK$_x$) for all $x$, and also generates (PK$_u$, SK$_u$).
2) The attacker generates the rating matrices $R_x$ for all $x$, the social graph SG, and selects a user $u$ and his friend set $F_u$. The attacker also chooses $f^t \in F_u$.
3) The challenger generates a random bit $b$, and gives SK$_u$ and SK$_f$ for all $f \in F_u$ except for SK$_{f^t}$ to the attacker.
4) The challenger and the attacker run the RSPProtocol protocol. The attacker simulates user $u$ and users from $F_u$ except for $f^t$, and the challenger simulates the rest. In the protocol execution, the challenger first samples the stranger set $T_u$, and then proceeds as follows.
- If $b = 0$ the challenger uses the original rating matrices $R_x$ for all $t \in T_u$.
- Otherwise it uses any $R_x^t$ for all $t \in T_u$ such that Prediction($R_u$, $R_f$, $\forall f \in F_u$, $\forall t \in T_u$) = Prediction($R_u$, $R_f^t$, $\forall f \in F_u$, $\forall t \in T_u$).
5) At the end, the attacker outputs a bit $b'$.

Robustness. With respect to the robustness of a recommender system, we only consider the scenario where a user tries to provide ill-formed rating values. For instance, such a user could set a rating value to be 1000 instead of 0 - 5. Other types of attacks against robustness (e.g. profile injection) can be addressed by existing countermeasures that have been proposed in other works.

Under our assumptions, user $u$’s friends will follow the protocol but may try to infer $f^t$’s input. The attack game is shown in Fig. 6, and the detailed description follows.

In the view of $f^t \in F_u$, user $u$ and $u$’s other friends will follow the protocol but may try to infer $f^t$’s input. The attack game is shown in Fig. 6, and the detailed description follows.

![Fig. 6: Security for a Friend against User $u$ and other Friends](image-url)
C. 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. 7. This can be regarded as a combined version of the games in Fig. 2, 3, 5, by assuming a malicious attacker.

1) The challenger generates the key pairs \((PK_u, SK_u)\) for all \( x \), and also generates \((PK_x, SK_x)\).
2) The attacker generates the rating matrices \( R_x \) for all \( x \), the social graph \( SG \), and selects a user \( u \) and his friend set \( F_u \).
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 \( u \) and the attacker simulates the rest. In the protocol execution, the attacker first samples the stranger set \( T_u \) which should include \( t^* \), and then proceeds as follows.
   - If \( b = 0 \) the challenger uses the original rating matrix \( R^*_u \).
   - Otherwise it uses any \( R^*_u \) such that \( \text{Prediction}(R^*_u, R_f) \forall f \in F_u, R_f \forall t \in T_u = \text{Prediction}(R^*_u, R_f) \forall f \in F_u, R_f \forall t \in T_u(t \neq t^*), R^*_f) \).
5) At the end, the attacker outputs a bit \( b' \).

In the view of a friend \( f^* \in F_u \), 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. 9, and it is an enhanced version of the attack game in Fig. 6 in the malicious model.

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

Robustness. In the worst case, both a stranger and a friend may provide ill-formed rating values. Formally,
robustness is defined as follows: for any user \( x \) which is in \( F_u \) or \( T_u \), if the input is not in either of the following forms then it can be detected with overwhelming probability.

- \( q_{x,b} \cdot (r_{x,b} - \bar{r}_x) = 0 \) and \( q_{x,b} = 0 \).
- \( -5 \leq q_{x,b} \cdot (r_{x,b} - \bar{r}_x) \leq 5 \) and \( q_{x,b} = 1 \).

IV. Friendship-based Recommender Protocols

With respect to the tailored recommender algorithms in Section III, the global system parameters should be established in advance. Such parameters should include \( \alpha, \beta \) 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 \( T_u \).

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 \( R_u \) his social graph, and assigns a weight \( w_{u,f} \) to each of his friend \( f \in F_u \). All other users perform the same operations in this phase.

Next, we describe two protocols from [52]: one for the active user to learn the predicted rating for an unrated item, and the other is for the active user to learn Top-n unrated items.

A. Single Prediction Protocol

When user \( u \) wants to test whether the predicted rating for an unrated item \( b \) is above a certain threshold \( \tau \) (an integer) in his mind, he initiates the protocol in Fig. 10. 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 \( I_b \), which only has 1 for the \( b \)-th element, and sends the ciphertext \([I_b]_u = \text{Enc}(PK_u, I_b)\) to the server. Let’s assume \([I_b]_u = (I_b^{(1)}_u, \ldots, I_b^{(M)}_u)\).

b) The server first sends \( PK_u \) to some randomly chosen strangers, and see whether they want to participate in the computation.

c) After the server has successfully found a viable stranger set \( T_u \), it forwards \([I_b]_u \) to every user in \( T_u \).

d) With \( PK_u \) and \((R_f, Q_f)\), every user \( t \) from \( T_u \) can compute the following based on the homomorphic properties.

\[
[q_{b,t}]_u = \sum_{1 \leq i \leq M} \text{Eval}('r, \text{Enc}(PK_u, q_{b,t}), [I_b^{(i)}]_u)
\]

\[
[R_f \cdot I_b]_u = \sum_{1 \leq i \leq M} \text{Eval}('r, \text{Enc}(PK_u, r_{b,t}), [I_b^{(i)}]_u)
\]

\[
[q_{b,t} \cdot (R_f \cdot I_b - \bar{r}_b)]_u = \text{Eval}('r, [q_{b,t}]_u, \text{Eval}(+, [R_f \cdot I_b]_u, \text{Enc}(PK_u, -\bar{r}_b)))
\]

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

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

b) The server sends \([w_{u,f}]_u \) and \([I_b]_u \) to user \( f \).

c) With \( PK_u \) \([I_b]_u, [w_{u,f}]_u \) and \((R_f, Q_f)\), user \( f \) can compute the following.

\[
[q_{f,b}]_u = \sum_{1 \leq i \leq M} \text{Eval}('r, \text{Enc}(PK_u, q_{f,b}), [I_b^{(i)}]_u)
\]

\[
[R_f \cdot I_b]_u = \sum_{1 \leq i \leq M} \text{Eval}('r, \text{Enc}(PK_u, r_{f,b}), [I_b^{(i)}]_u)
\]

\[
[q_{f,b} \cdot (R_f \cdot I_b - \bar{r}_b) \cdot w_{u,f}]_u = \text{Eval}('r, \text{Eval}(+, [q_{f,b}]_u, [w_{u,f}]_u), \text{Eval}(+, [R_f \cdot I_b]_u, \text{Enc}(PK_u, -\bar{r}_b)))
\]

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

a) User \( u \) sends his encrypted average rating \([\bar{r}_u]_u = \text{Enc}(PK_u, \bar{r}_u)\) to the server.

b) The server first computes \([n_f]_u, [d_f]_u, [n_f]_u, [d_f]_u\) as shown in Fig. 10, and then computes \([X]_u, [Y]_u\) as follows.

\[
temp_1 = \text{Eval}(, \text{Eval}(, [d_f]_u, [\bar{r}_u]_u), [d_f]_u, \text{Enc}(PK_u, \alpha + \beta))
\]

\[
temp_2 = \text{Eval}(, \text{Eval}(, [n_f]_u, [d_f]_u), \text{Enc}(PK_u, \beta))
\]

\[
temp_3 = \text{Eval}(, \text{Eval}(, [n_f]_u, [d_f]_u), \text{Enc}(PK_u, \alpha))
\]

\[
[X]_u = \text{Eval}(+, \text{Eval}(+, \text{temp}_1, \text{temp}_2), \text{temp}_3)
\]

\[
[Y]_u = \text{Eval}(, \text{Eval}(, [d_f]_u, [d_f]_u), \text{Enc}(PK_u, \alpha + \beta))
\]

Referring to Equations (2) and (3), we have \( p^\ast_{u,b} = \bar{r}_u + \frac{n_f}{d_f} \) and \( p^\ast_{u,b} = \bar{r}_u + \frac{n_f}{d_f} \). The ultimate prediction \( p_{u,b} \) can be denoted as follows.

\[
p_{u,b} = \frac{\beta \cdot p^\ast_{u,b} + \alpha \cdot \bar{r}_u}{\alpha + \beta} \cdot p^\ast_{u,b} - \frac{(\alpha + \beta) \cdot d_f \cdot d_f \cdot \bar{r}_u + \beta \cdot n_f \cdot d_f + \alpha \cdot n_f \cdot d_f}{(\alpha + \beta) \cdot d_f \cdot d_f} = \frac{X}{Y}
\]

c) User \( u \) runs a comparison protocol COM with the server to learn whether \( p_{u,b} \geq \tau \). Since \( X, Y, \tau \) are integers, COM is indeed an encrypted integer comparison protocol: where user \( u \) holds the private key \( sk_u \) and \( \tau \), the server holds \([X]_u, [Y]_u\), and the protocol outputs a bit to user \( u \) indicating whether \( X \geq \tau \cdot Y \). Many such protocols exist, we present an implementation in Section VI-D based on a protocol by Veugen [53].
ing zero knowledge proofs. In more detail, at the end of Stage 1 and Stage 2, we simplify the computation of \( X \), \( \beta \), and \( \alpha \).

When the active user \( u \) wants to figure out Top-n unrated items, he initiates the protocol in Fig. 11. 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 \( T_u \).

b) With \( PK_u \) and \( (R_t, Q_t) \), user \( t \in T_u \) can compute \( [q_{f,b} \cdot (r_{f,b} - \overline{T}_f)]_b = \text{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{T}_f)) \) and \( [q_{f,b}]_b = \text{Enc}(PK_u, q_{f,b}) \) 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 \( f \in F_u \), user \( u \) sends the encrypted weight \([w_{u,f}]_u = \text{Enc}(PK_u, w_{u,f})\).

b) With \( PK_u \), \([w_{u,f}]_u \) and \( (R_f, Q_f) \), user \( f \) can compute \([q_{f,b}]_u \) and

\[
[q_{f,b} \cdot (r_{f,b} - \overline{T}_f)]_u = \text{Eval}(\cdot, \text{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{T}_f)), [w_{u,f}]_u)
\]

for every \( 1 \leq b \leq M \). All encrypted values are sent back to the server.

B. Top-n Protocol

When the active user \( u \) wants to figure out Top-n unrated items, he initiates the protocol in Fig. 11. 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 \( T_u \).

b) With \( PK_u \) and \( (R_t, Q_t) \), user \( t \in T_u \) can compute \( [q_{f,b} \cdot (r_{f,b} - \overline{T}_f)]_b = \text{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{T}_f)) \) and \( [q_{f,b}]_b = \text{Enc}(PK_u, q_{f,b}) \) 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 \( f \in F_u \), user \( u \) sends the encrypted weight \([w_{u,f}]_u = \text{Enc}(PK_u, w_{u,f})\).

b) With \( PK_u \), \([w_{u,f}]_u \) and \( (R_f, Q_f) \), user \( f \) can compute \([q_{f,b}]_u \) and

\[
[q_{f,b} \cdot (r_{f,b} - \overline{T}_f)]_u = \text{Eval}(\cdot, \text{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{T}_f)), [w_{u,f}]_u)
\]

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 $M_X, M_Y$ as follows: (1) generate a $M \times M$ identity matrix; (2) randomly permute the columns to obtain $M_Y$; (3) to obtain $M_X$, for every $b$, if item $b$ has been rated then replace the element 1 in $b$-th column with 0.

\[
\begin{bmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1
\end{bmatrix} \rightarrow M_Y =
\begin{bmatrix}
0 & 1 & \ldots & 0 \\
0 & 0 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \ldots & 0
\end{bmatrix}
\]

\[
\rightarrow M_X =
\begin{bmatrix}
0 & 1 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \ldots & 0
\end{bmatrix}
\]

User $u$ encrypts the matrices (element by element) and sends $[M_X]_u, [M_Y]_u$ 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. 11, in the same way as in the previous protocol in Fig. 10. Referring to Formula (4), we see that $\bar{r}_u$ 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.

\[
p_{u,b} = \frac{\beta}{\alpha + \beta} n_{T,b} + \frac{\alpha}{\alpha + \beta} n_{F,b} = \frac{\beta \cdot n_{T,b} + \alpha \cdot n_{F,b} \cdot d_{T,b}}{(\alpha + \beta) \cdot d_{F,b}} = \frac{X_b}{Y_b}
\]

ii) The server permutes the ciphertext vector $[(X_1)_u, (Y_1)_u, (X_2)_u, (Y_2)_u, \ldots, (X_M)_u, (Y_M)_u]$ in an oblivious manner as follows.

\[
([U_1]_u, [U_2]_u, \ldots, [U_M]_u) = [M_X]_u \cdot ([X_1]_u, [X_2]_u, \ldots, [X_M]_u)^T
\]

\[
([V_1]_u, [V_2]_u, \ldots, [V_M]_u) = [M_Y]_u \cdot ([Y_1]_u, [Y_2]_u, \ldots, [Y_M]_u)^T
\]

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 $\text{Eval}(\cdot, \cdot)$ and the addition is done with $\text{Eval}(+, \cdot)$. 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 $u \frac{i}{T}$ ($1 \leq i \leq |B|$) in the encrypted form. One straightforward way of constructing the RANK protocol is to combine an encrypted integer comparison protocol $\text{COM}$ and any standard sorting algorithm. The $\text{COM}$ protocol has slightly different semantics from that in the previous protocol in Section IV-A: 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. Regardless the difference, similar to the implementation from Section VI-D, it is straightforward to get such a protocol based on that of Veugen [53].

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.

The usage of matrix $M_X$ 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.

Compared to the protocol from [52], 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 [52]; (3) we correct two errors in the computation of $[X_b]_u$ and $[d_{F,b}]_u$ in Fig. 4 in [52].

V. Construction and Analysis of Twitter Datasets

In this section, we construct new datasets based on the MovieTweetings dataset [15] (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. The making of a new Twitter dataset is straightforward, namely users $x$ and $y$ can be considered as friends without considering the original tweets.

All the original figures and source codes of the experiments are uploaded to Github 1.

A. Dataset Construction

MovieTweetings consists of ratings on movies that are extracted from tweets [15]. Such tweets originate from the social rating widget available in IMDb apps. The tweets are well structured in the following form:

“I rated The Matrix 9/10
http://www.imdb.com/title/tt0133093/#IMDb"
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 friend and each friend 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 three new datasets which contain friendship information.

- In the first dataset, each user has at least 1 friend and each friend has at least 1 rating. We call it the FMT dataset.
- The second is a subset of the FMT dataset, where each user has at least 5 friends and each friend has at least 5 ratings. We call it the 5-FMT dataset.
- The third is a subset of the 5-FMT dataset, where each user has at least 10 friends and each friend has at least 10 ratings. We call it the 10-FMT dataset.

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 is incomplete. We summarize some basic information of these datasets in the following table.

### Table II: Basic Facts

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Friends</th>
<th>Strangers</th>
<th>Min Ratings/User</th>
<th>Ave. Ratings/User</th>
<th>Max Ratings/User</th>
<th>Friends/Strangers</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>20156</td>
<td>35456</td>
<td>1</td>
<td>12</td>
<td>856</td>
<td>10</td>
</tr>
<tr>
<td>FMT</td>
<td>211954</td>
<td>359908</td>
<td>1</td>
<td>18</td>
<td>856</td>
<td>10</td>
</tr>
<tr>
<td>5-FMT</td>
<td>40987</td>
<td>40987</td>
<td>1</td>
<td>5</td>
<td>856</td>
<td>10</td>
</tr>
<tr>
<td>10-FMT</td>
<td>20316</td>
<td>20316</td>
<td>1</td>
<td>5</td>
<td>856</td>
<td>10</td>
</tr>
</tbody>
</table>

B. Social Graph Characteristics

Due to the fact that friendship links are very sparse in the FMT dataset, we will mainly use the 5-FMT and 10-FMT datasets in our analysis. We summarize some simple facts in Table II. Note that, besides friends, we also count the number of friends of friends (FoFs).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Min Friend Num</th>
<th>Avg Friend Num</th>
<th>Max Friend Num</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-FMT Dataset (FoFs)</td>
<td>5</td>
<td>20</td>
<td>149</td>
</tr>
<tr>
<td>5-FMT Dataset (Friends)</td>
<td>5</td>
<td>20</td>
<td>524</td>
</tr>
<tr>
<td>10-FMT Dataset (Friends)</td>
<td>10</td>
<td>27</td>
<td>109</td>
</tr>
<tr>
<td>10-FMT Dataset (FoFs)</td>
<td>10</td>
<td>203</td>
<td>329</td>
</tr>
</tbody>
</table>
To get more fine-grained statistics on the friendship links, we present four histograms in Fig. 12, 13, 14, 15. The “Friend Num” value represents the number of friends that a user can have, and the “Frequency” value represents how many users have “Friend Num” friends (FoFs).

We then map the 5-FMT and 10-FMT datasets into directed graphs, in Fig. 16 and 17 respectively. In the graphs, 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. To get a precise idea on the connectivity property among users, we summarize the degree of separations in two histograms, in Fig. 18 and 19 respectively. If users $x$ and $y$ have $t$ degree of separations, it means the length of the shortest path from user $x$ to user $y$ is $t$. It is interesting to see that the users are well connected.
C. Friends Similarity Characteristics

In order to test the folklore that friends share more similarities than strangers, we compute the Cosine similarities between users in the 5-FMT and 10-FMT datasets and plot them in Fig. 20 and 21. Based on the fact that most ▶ dots are distributed above □ dots, we can conclude that friends typically share more similarities than strangers.

VI. Analysis of the Proposed Protocols

In this section, we investigate the recommendation accuracy of the Formula (4) from Section III-A and complexities of the protocols from Section IV-A and IV-B.

A. Recommendation Accuracy: MovieLens Dataset

We choose the MovieLens 100k dataset [48] and 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 neighborhood as the friends, and randomly choose a certain number of users from the rest as strangers. It is worth stressing that we assume friendship to be bilateral.

For different parameters, the MAE values of the proposed protocols are shown in Table III. The column denotes the possible values of \( \frac{\alpha}{\alpha + \beta} \) and the row denotes the possible values of \( (|F_u|, |T_u|) \). Note that lower MAE implies more accurate recommendations.
With respect to the JPH protocols, we carry out the experiment as follows.

1) Calculate the Cosine similarity matrix.
2) Choose a user $u$, compute $w_{u,f}$ and $w_{f,u}$ as follows. If $f$ is $u$’s friend, set $w_{u,f}$ to be the Cosine similarity between $u$ and $f$. If $u$ is not a friend of $f$, then set $w_{f,u} = 0$.
3) Compute predictions according to Formula (1) in Section III.
4) 5-fold cross validation, namely run Steps 2 and 3 five times and take the average.

The MAE is $1.9626 = \frac{1}{5} \times 1.9381 + 1.8867 + 1.9264 + 1.8872$. In Step 2, we always set $w_{f,u} = 0$ when $u$ is not a friend of $f$. This may seems unfair for the JPH protocols. If we always set $w_{f,u} = w_{u,f}$, then the MAE is $1.2037 = \frac{1}{5} \times 1.3563 + 1.2450 + 1.1049 + 1.3718 + 1.1426$. Even in this case their accuracy is much worse than ours.

### C. Asymptotic Computational Complexities

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

#### TABLE VI: Complexity of Single Prediction Protocol

<table>
<thead>
<tr>
<th></th>
<th>Enc</th>
<th>Eval($\ast$,..)</th>
<th>Eval(,...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friend</td>
<td>$2M + 1$</td>
<td>$2M - 1$</td>
<td>$2M + 2$</td>
</tr>
<tr>
<td>Stranger</td>
<td>$2M + 1$</td>
<td>$2M - 1$</td>
<td>$2M + 1$</td>
</tr>
<tr>
<td>Server</td>
<td>$4$</td>
<td>$2</td>
<td>T_u, u] + 2</td>
</tr>
<tr>
<td>User $u$</td>
<td>$M +</td>
<td>F_u</td>
<td>+ 1$</td>
</tr>
</tbody>
</table>

#### TABLE VII: Complexity of Top-n Protocol

### D. Estimated Timing Information

To get some idea on the real-world performance, we estimate the running time of the single prediction protocol. The running time of the Top-n protocol can be easily obtained based on this estimation.

We adopt the MovieLens 100k dataset where $M = 1700$ and set $(|F_u|, |T_u|) = (10, 10)$. We use an implementation of the Brakerski scheme from [57], which has the timing...
cost for \( \text{Eval}(+,...) \) (6.64 \( \mu \)s), \( \text{Eval}(.,...) \) with key switching (2.66 ms), \( \text{Enc} \) (0.11 \( \mu \)s), \( \text{Dec} \) (0.35 ms), based on an Intel(R) Core(TM) i7-5600U CPU 2.60GHz. The estimated time information for SWHE-related operations is shown in Table VIII.

<table>
<thead>
<tr>
<th>Time (second)</th>
<th>Friend</th>
<th>Stranger</th>
<th>Server</th>
<th>User u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>0.05</td>
<td>0.0002</td>
</tr>
</tbody>
</table>

**TABLE VIII: Timing Numbers**

Next, we describe a COM protocol shown in Fig. 22 based on that of Veugen [53] and evaluate its performance. In addition to the SWHE scheme, the protocol also relies Goldwasser-Micali scheme [20]. In both schemes, we set the bit-length of the prime number to be 512. If \( x \) stand for plaintext, \( [x]_\mu \) denotes a ciphertext under the SWHE, \( [x]_\tau \) denotes a Paillier ciphertext, \( [x]_{GM} \) denotes a Goldwasser-Micali ciphertext. The expression \( a?b : c \) means that if \( a \) is true, execute \( b \) operation, otherwise execute \( c \). We let \( a_t \) and \( b_t \) stand for the least significant \( t \)-th bit of \( a \) and \( b \). We let \( \text{len} \) denotes the bit-length of \( Y_t \), which is about 20 for the Movielens dataset. To achieve a 70-bits statistical security for randomizing the \( X,Y \) values, we use a set \( \mathcal{R} = [0,1]^100 \). At the end of the protocol, \( t = 1 \) implies \( \frac{x}{\tau} \approx \tau \).

We implement the Goldwasser-Micali scheme, which has the timing cost for \( \text{Enc} \) (1.5 \( \mu \)s), \( \text{Dec} \) (4.5 \( \mu \)s), 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.

**E. Security Analysis of the Proposed Protocols**

In the proposed 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 [52], [25]. Referring to the security properties from Section III-B and III-C, these properties are achieved based on the following assumptions and facts: (1) semi-honest assumptions on friends and the service provider; (2) the semantic security of the underlying SWHE scheme; (3) the security of the comparison protocol COM; (4) the fact that everything is encrypted under user \( u \)'s public key and only the final results are returned to user \( u \) in the encrypted form. We skip the formal proofs because the reductions are straightforward.

In our security models, we do not consider the potential information leakages from the output of a recommender system. Intuitively, it is related to the global parameters \( \alpha, \beta \) and the sizes of \( F_L \) and \( T_U \). If \( \frac{\alpha}{\beta} \) gets larger or the size of \( T_U \) 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. We try to shed lights on this aspect via the following analysis.

**Additional analysis for \( f \in F_U \).** Informally, a friend \( f \)'s contribution to \( p_{u,f} \) is protected by the inputs from users \( F_L \setminus f \) and strangers \( T_U \). We perform an experiment to show how a single friend influences the prediction rating. For illustration purpose, we use the Movielens 100k dataset, and set \( \frac{\alpha}{\beta} = 0.8 \) and the \( (|F_L|,|T_U|) \) is \((10,10)\).

In the experiment, we run 5-fold cross validation 50 times. As shown in Table II, in 10-FMT dataset, users have 27 friends in average. Thus this experiment can cover all the friends of most users. 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 \( P_0 \).
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 \( P_1 \).
4) Compute the prediction difference vector as \( P_0 - P_1 \). With all the prediction difference vectors \( P_0 - P_1 \) in the experiment, we plot the frequency of all difference values in Fig. 23.

---

**Fig. 22: COM Protocol**
It is clear that most friends have very small influence on user $u$’s output. Informally, we can argue that a friend’s input is often obfuscated by the inputs from others and inference should be difficult solely based on the output. For similar reasons, it will be hard for $F_u \setminus f$ to infer user $f$’s data even if they learned user $u$’s output at the end of a protocol execution.

Additional analysis for $t \in T_u$. 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 $t$.

We perform a similar experiment to what we have done for testing a single friend’s influence on the output of user $u$. The difference is that the operations on friends and strangers are swaped. The experiment results are plotted in Fig. 24. In comparison to Fig. 23, it is clear that a stranger is likely to have less influence on user $u$’s output than a friend.

As it shows in Fig. 23 and 24, there are some differences larger than 0.5. Precisely, the ratio is about 4% in Fig. 23, and it is about 10% in Fig. 24. The reason is that the 10-FMT is very sparse, and in our experiments, we set $(|F_u|, |T_u|) = (10, 10)$. It means that the neighborhood set is very small and a single user’s influence could be significant. The influence can be rapidly decreased if we increase the size of $F_u$ and $T_u$.

VII. Decentralized Prediction Generation

For simplicity, we assume that users are uniquely identified in the recommender system, and they share their social graph in public. 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 $R_u$, his social graph, and assigns a weight $w_{u,f}$ to each of his friend $f \in F_u$. 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. 25. Due to the small world phenomenon, the population of FoFs can already be very large (see [2]). In our 10-FMT dataset in Section V, the FoFs of a user cover more than half of the whole population (see Fig. 19).

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 IV-A.

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 $f^* \in T_u$. We further require that the every $f \in F_u$ should have at least one friend in $T_u$.

2) User $u$ generates a binary vector $I_u$, which only has 1 for the $b$-th element, and broadcasts $[I_u]_u =
However, it also has the following disadvantages.

3) With \( PK_u \), \( \{I_u\}_u \) and \( \{R_i, Q_i\}_i \), user \( f \) can compute the \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f) \cdot w_{u,f}]_u \) in exactly the same way as in Section IV-A. User \( f \) then sends \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f) \cdot w_{u,f}]_u \) to one of his friends in \( T_u \). He also forwards \( \{I_b\}_u \) and \( PK_u \) to the chosen friend.

4) For any \( t \in T_u \), he should receive \( \{I_b\}_u \) 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 \( \{R_i, Q_i\}_i \), every user \( t \) from \( T_u \) can compute \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) and \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) in exactly the same way as in Section IV-A.

   c) Suppose that user \( f \) has received \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \). He computes \( \sum_{f \in E_F} [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) and \( \sum_{f \in E_F} [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) to user \( t \).

   d) User \( f \) sends \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) and \( \sum_{f \in E_F} [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) to user \( t \).

5) User \( t' \) receives \( [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) and \( \sum_{f \in E_F} [q_{j,b}^r \cdot (R_f \cdot I_b - I_f)]_u \) from \( t \in T_u \). He then does the following.

   a) Compute \( [n_{j,b}^r, d_{j,b}^r, n_{j,b}^r, d_{j,b}^r]_u \) in exactly the same way as in Section IV-A.

   b) Run a comparison protocol COM with user \( u \) for the latter to learn whether \( \frac{n_{j,b}^r}{d_{j,b}^r} \geq \tau \).

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 IV-A, 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 friends and 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 on the service provider to blend their inputs any more, and the trust has been shifted to user user \( t' \).

Clearly, from the efficiency perspective, the centralized solution from Section IV-A 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. Conclusion**

In the paper, we first proposed two security models for recommender systems and then clarified the protocols from [52] and argued their security in the security models. The security properties are indeed fairly straightforward since these protocols heavily rely on SWHE schemes. What is non-trivial is that, through experiments, we showed that the information leakages from the output is indeed quite small. The idea of introducing randomly selected strangers to prevent information leakages from the output share some similarity with the differential privacy based approach [36], [55] and the differential identifiability approach [31]. A more rigorous comparison remains as an interesting future work, particularly in the line of the works from [7], [16]. With the Movielens 100k dataset and newly-constructed Twitter datasets, we demonstrated the accuracy and computational complexities for our protocols. It turned out that the single prediction protocol can be considered practical while the Top-n protocol is less efficient. In this paper, we also tried to shed some light on the tradeoffs between centralized and decentralized privacy-preserving solutions. It seems that a centralized solution is more realistic in practice.

**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 evaluation in Section VI-D and thank David Wu from Stanford University for answering our questions about the implementation of Brakerski scheme.

**References**


