Privacy-preserving Context-aware Recommender Systems: Analysis and New Solutions

Qiang Tang and Jun Wang

APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
{qiang.tang, jun.wang}@uni.lu

April 9, 2015

Abstract. Nowadays, recommender systems have become an indispensable part of our daily life and provide personalized services for almost everything. However, nothing is for free – such systems have also upset the society with severe privacy concerns because they accumulate a lot of personal information in order to provide recommendations. In this work, we construct privacy-preserving recommendation protocols by incorporating cryptographic techniques and the inherent data characteristics in recommender systems. We first revisit the protocols by Jeckmans et al. at ESORICS 2013 and show a number of security and usability issues. Then, we propose two privacy-preserving protocols, which compute predicted ratings for a user based on inputs from both the user's friends and a set of randomly chosen strangers. A user has the flexibility to retrieve either a predicted rating for an unrated item or the Top-N unrated items. The proposed protocols prevent information leakage from both protocol executions and the protocol outputs: a somewhat homomorphic encryption scheme is used to make all computations run in encrypted form, and inputs from the randomly-chosen strangers guarantee that the inputs of a user's friends will not be compromised even if this user's outputs are leaked. Finally, we use the well-known MovieLens 100k dataset to evaluate the performances for different parameter sizes.

1 Introduction

As e-commerce websites began to develop, users were finding it very difficult to make the most appropriate choices from the immense variety of items (products and services) that these websites were offering. 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. As such, the availability of choices, instead of producing a benefit, started to decrease users' well-being. Eventually, this need led to the development of recommender systems (or, recommendation systems). Informally, recommender systems are a subclass of information filtering systems that seek to predict the 'rating' or 'preference' that a user would give to an item (e.g. music, book, or movie) they had not yet considered, using a model built from the characteristics of items and/or users. Today, recommender systems play an important role in highly rated commercial websites such as Amazon, Facebook, Netflix, Yahoo, and YouTube. Netflix even awarded a million dollars prize to the team that first succeeded in improving substantially the performance of its recommender system. Besides these well-known examples, recommender systems can also be found in every corner of our daily life.

In order to compute recommendations, the service provider needs to collect a lot of personal data from its users, e.g. ratings, transaction history, and location. 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. Privacy issues in recommender systems have been surveyed in [4, 17, 29]. 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). Weinsberg et al. showed that what has been rated by a user can already breach his privacy [34]. 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 [16]. Recently Calandrino et al. [7] 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.

1.1 State-of-the-Art

Broadly speaking, existing privacy-protection solutions for recommender systems can be divided into two categories. One category is cryptographic solutions, which heavily rely on cryptographic primitives (e.g. homomorphic encryption, zero knowledge proof, threshold encryption, commitment, private information retrieval, and a variety of two-party or multi-party cryptographic protocols). For example, the solutions from [2, 8, 9, 12–15, 20, 24, 25, 28, 32, 36] fall into this category. More specifically, the solutions from [2, 8, 9, 12, 15, 20, 32] focus on distributed setting where every individual user is expected to participate in the recommendation computation, while those from [13, 14, 24, 25, 28, 36] focus on partitioned dataset, where several organizations wish to compute recommendations for their own users by joining their private dataset. These solutions typically assume semi-honest attackers and apply existing cryptographic primitives to secure the procedures in standard recommender protocols. This approach has two advantages: rigorous security guarantee in the sense of secure computation (namely, every user only learns the recommendation results and the server learns nothing) can be achieved, and there is no degradation in accuracy. The disadvantage lies in the fact that these solutions are all computation-intensive so that they become impractical when user/item populations get large.

The other category is data obfuscation based solutions, which mainly rely on adding noise to the original data or computation results to achieve privacy. The solutions from [5, 19, 21–23, 26, 27, 31, 35] fall into this category. 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. With respect to privacy guarantees, an exception is the differential privacy based approach from [19] which does provide mathematically sound privacy notions. However, cryptographic primitives are required for all users to generate the accumulated data subjects (e.g. sums and covariance matrix).

1.2 Our Contribution

While most privacy-preserving solutions focus on recommender systems which only take into account users' ratings as inputs, Jeckmans, Peter, and Hartel [14] moved a step further to propose privacy-preserving recommendation protocols for context-aware recommender systems, which include social relationships as part of the inputs to compute recommendations. Generally the protocols are referred to as the JPH protocols, and more specifically they are referred to as JPH online protocol and JPH offline protocol respectively. Interestingly, the JPH protocols make use of the recent advances in somewhat homomorphic encryption schemes [6]. In this paper, our contribution is three-fold.

Firstly, we analyze the JPH protocols and identify a number of security issues. The first issue is that the somewhat homomorphic encryption scheme has been used in very naive way in the JPH online protocol, so that the semi-honest server can learn which user has rated which items and the actual ratings. The second issue is that a user's private information may be leaked through his friend's output in the JPH online protocol. A privacy breach occurs when the attacker has learned a predicted rating for the friend. The third issue is that the encrypted division method used in the JPH protocols may leak unnecessary information about a user's friends. The fourth issue is that a group of users can mount key recovery attacks against the server to recover its private key in the JPH offline protocol, thus these users can decrypt everything protected by the private key. The fifth issue with the JPH protocols is that in each

protocol execution predicted ratings are required to be computed for all items, namely not only for unrated items but also already-rated items. This incurs not only computational overhead but also unnecessary information leakage.

Secondly, we revise the prediction computation formula from [14] by incorporating inputs from both friends and strangers. This change not only aligns the formula with standard recommender algorithms [18] but also enables us to avoid the cold start problem of the JPH protocols. Security wise, it helps us prevent potential information leakages through the outputs of friends. We then propose two privacy preserving protocols. One enables a user to check whether a specific unrated item might be of his interest, and the other returns the Top-N unrated items. Therefore, we provide more flexible choices for users to discover their interests in practice. Both protocols are secure against envisioned threats in our threat model.

Thirdly, we analyze accuracy performances of the new protocols, and show that for some parameters the accuracy is even better than some other well-known recommendation protocols, e.g. those from [18].

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we present preliminaries on notation and building blocks. In Section 3, we demonstrate the security and usability issues with the JPH protocols. In Section 4, we propose our new formulation and trust assumptions for recommender systems. In Section 5, we present two protocols for single prediction and Top-N recommendations respectively. In Section 6, we present security and accuracy analysis for the proposed protocols. In Section 7, we conclude the paper.

2 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 $\boldsymbol{X}$, to denote a vector. Given two vector $\boldsymbol{X}$ and $\boldsymbol{Y}$, we use $\boldsymbol{X} \cdot \boldsymbol{Y}$ to denote their inner product.
- In a recommender system, the item set is denoted by a vector $\mathbf{B} = (1, 2, \cdots, b, \cdots, |\mathbf{B}|)$, and a user x's ratings are denoted by a vector $\mathbf{R}_x = (r_{x,1}, \cdots, r_{x,b}, \cdots, r_{x,|\mathbf{B}|})$. 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. With respect to $\mathbf{R}_x$, a binary vector $\mathbf{Q}_x = (q_{x,1}, \cdots, q_{x,b}, \cdots, q_{x,|\mathbf{B}|})$ is defined as follows: $q_{x,b} = 1$ iff $r_{x,b} \neq 0$ for every $1 \leq b \leq |\mathbf{B}|$.

2.1 Brakerski-Vaikuntanathan SWHE Scheme

Let λ be the security parameter. The Brakerski-Vaikuntanathan public-key SWHE scheme [6] is parameterized by two primes $q, t \in \text{poly}(\lambda) \in \mathbb{N}$ where $t < q$, a degree n polynomial $f(x) \in \mathbb{Z}[x]$, two error distributions χ and χ' over the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$. The message space is $\mathcal{M} = \mathbf{R}_t = \mathbb{Z}_t[x]/\langle f(x)\rangle$. An additional parameter is $D \in \mathbb{N}$, namely the maximal degree of homomorphism allowed (and to the maximal ciphertext length). The parameters $n, f, q, t, \chi, \chi', D$ are public.

- $\mathsf{Keygen}(\lambda)$: (1) sample $s, e_0 \leftarrow \chi$ and $a_0 \in R_q$; (2) compute $\mathbf{s} = (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$; (3) output $SK = \mathbf{s}$ and $PK = (a_0, b_0 = a_0 s + te_0)$.
- $\mathsf{Enc}(PK, m)$: (1) sample $v, e' \leftarrow \chi$ and $e'' \leftarrow \chi'$; (2) compute $c_0 = b_0 v + te'' + m$, $c_1 = -(a_0 v + te')$; (3) output $\mathbf{c} = (c_0, c_1)$.
- $\mathsf{Dec}(SK, \mathbf{c} = (c_0, \ldots, c_D) \in R_q^{D+1})$: output $m = (\mathbf{c} \cdot \mathbf{s} \bmod q) \bmod t$.

Since the scheme is somewhat homomorphic, it provides an evaluation algorithm Eval, which can multiply and add messages based on their ciphertexts only. For simplicity, we show how Eval works when the ciphertexts are freshly generated. Let $\mathbf{c}_\alpha = (c_{\alpha 0}, c_{\alpha 1})$ and $\mathbf{c}_\beta = (c_{\beta 0}, c_{\beta 1})$. Note that the multiplication operation will add an additional element for the ciphertext. This

is why the Dec algorithm generally assumes the ciphertext to be a vector of $D+1$ elements (if the ciphertext has less elements, simply pad 0s).

$$\mathsf{Eval}(+, \mathbf{c}_\alpha, \mathbf{c}_\beta) = (c_{\alpha 0} + c_{\beta 0}, c_{\alpha 1} + c_{\beta 1}).\ \mathsf{Eval}(\cdot, \mathbf{c}_\alpha, \mathbf{c}_\beta) = (c_{\alpha 0} \cdot c_{\beta 0}, c_{\alpha 0} \cdot c_{\beta 1} + c_{\alpha 1} \cdot c_{\beta 0}, c_{\alpha 1} \cdot c_{\beta 1}).$$

When the evaluations are done to a ciphertext and a plaintext message, there is a simpler form for the evaluation algorithm, denoted as Eval^*. This has been used in [14].

$$\mathsf{Eval}^*(+, \mathbf{c}_\alpha, m') = (c_{\alpha 0} + m', c_{\alpha 1}).\ \mathsf{Eval}^*(\cdot, \mathbf{c}_\alpha, m') = (c_{\alpha 0} \cdot m', c_{\alpha 1} \cdot m').$$

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 $\sum_{1 \le i \le N} [m_i]_u$ to denote the result of sequentially applying $\mathsf{Eval}(+, ,)$ to the cipheretxts.

3 Analysis of JPH Protocols

Let the active user, who wants to receive new recommendations, be denoted as user u. Let the friends of user u be denoted by $\mathbf{F}_u$. Every friend $f \in \mathbf{F}_u$ and user u assigns each other weights $w_{f,u}, w_{u,f}$ respectively, and these values can be regarded as the perceived importance to each other. Then, the predicted rating for an unrated item $b \in \mathbf{B}$ for user u is computed as follows.

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

In the JPH protocols [14], Jeckmans et al. did not explicitly explain their notation $[x]_u + y$ and $[x]_u \cdot y$. We assume these operations are as $[x]_u + y = \mathsf{Eval}^*(+, [x]_u, y)$ and $[x]_u \cdot y = \mathsf{Eval}^*(\cdot, [x]_u, y)$. In any case, this assumption only affects the *Insecurity against Semi-honest Server* issue for the JPH online protocol. All other issues still exist even if this assumption is not true.

3.1 JPH Online Protocol

In the online scenario, the recommendation protocol is executed between the active user u, the server, and user u's friends. In the *initialization phase*, user u generates a public/private key pair for the Brakerski-Vaikuntanathan SWHE scheme, and all his friends and the server obtain a valid copy of his public key. The protocol runs in two stages as described in Fig. 1.

1. In the first stage, user u sends his encrypted weights to all his friends, who will then add their parts to the weights and prepare their contributions for computing the final predication as required by Equation (1).
2. In the second stage, user u and the server run some sort of two-party protocol so that the server can compute Equation (1) without learning user u's ratings and letting user u know his friends's contributions.

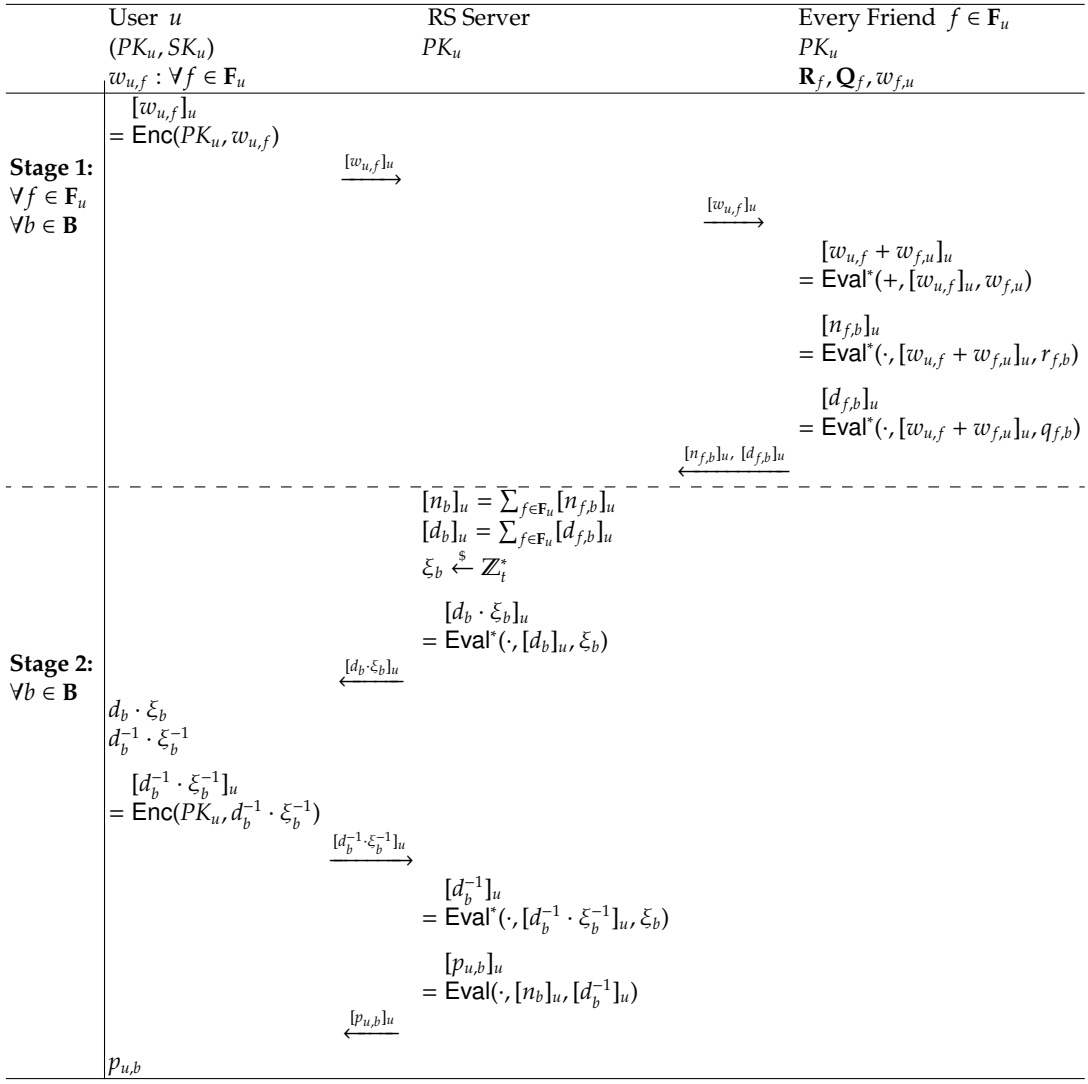

Fig. 1. JPH Online Protocol

It is worth noting that the friends only need to be involved in the first stage. Despite the security analysis in the original paper, we observe the following security issues.

- *Hidden assumption*. Jeckmans et al. [14] did not mention any assumption on the communication channel between users and the server. In fact, if the communication channel between any user f and the server does not provide confidentiality, then user u can obtain $[n_{f,b}]_u$, $[d_{f,b}]_u$ by passive eavesdropping. Then, user u can trivially recover $q_{f,b}$ and $r_{f,b}$.
- *Insecurity against Semi-honest Server*. With $[d_{f,b}]_u$, the server can trivially recover $q_{f,b}$, i.e. if $[d_{f,b}]_u = 0$ then $q_{f,b} = 0$; otherwise $q_{f,b} = 1$. After recovering $q_{f,b}$, the server can trivially recover $r_{f,b} = \frac{[n_{f,b}]_u}{[d_{f,b}]_u}$. The root of the problem is the homomorphic operations have been done in the naive way, with $\mathsf{Eval}^*(\cdot,,)$ and $\mathsf{Eval}^*(+,,)$.
- *Encrypted Division Problems*. The first concern is that it may not be able to determine the predicted rating $p_{u,b}$. As a toy example, let $t = 7$. In this case, both $\lfloor \frac{2}{3} \rceil = 1$ and $\lfloor \frac{3}{1} \rceil = 3$ link to the index $2 \cdot 3^{-1} = 3 \cdot 1^{-1} = 3 \mod 7$. If $p_{u,b} = 3$, then user u will not be able to determine whether the predicted rating is $\lfloor \frac{2}{3} \rceil = 1$ or $\lfloor \frac{3}{1} \rceil = 3$. The second concern is

that the representation of $p_{u,b}$ in the protocol may leak more information than the to-be predicted value $\lfloor \frac{n_b}{d_b} \rceil$. As an example, $\lfloor \frac{2}{3} \rceil = \lfloor \frac{3}{4} \rceil = 1$. Clearly, giving $2 \cdot 3^{-1}$ or $3 \cdot 4^{-1}$ leaks more information than the to-be predicted value 1.

– *Potential Information Leakage through Friends*. For user u, his friends may not be friends with each other. For example, it may happen that some friend $f \in \mathbf{F}_u$ is not a friend of any other user from $\mathbf{F}_u$. Suppose that the users $\mathbf{F}_u \backslash f$ have learned the the value $p_{u,b}$ or some approximation of it (this is realistic as they are friends of user u). Then, they may be able to infer whether user f has rated the item b and the actual rating.

Besides the above security issues, there are some usability issues with the protocol as well. One issue is that, at the time of protocol execution, maybe only a few friends are online. In this case, the predicted rating may not be very accurate. It can also happen that $p_{u,b}$ cannot be computed, because none of user u's friends has rated item b. This is the typical cold start problem in recommender systems [1]. The other issue is that the predicted rating needs to computed for every $b \in \mathbf{B}$ even if user u has already rated this item. Otherwise, user u may leak information to the server, e.g. which items have been rated. This not only leaks unnecessary information to user u, but also makes it very inefficient when user u only wants a prediction for a certain unrated item.

3.2 JPH Offline Protocol

In the offline scenario, the friends $\mathbf{F}_u$ need to delegate their data to the server to enable user u to run the recommendation protocol when they are offline. Inevitably, this leads to a more complex *initialization phase*. In this phase, both user u and the server generate their own public/private key pair for the Brakerski-Vaikuntanathan SWHE scheme and they hold a copy of the valid public key of each other. Moreover, every friend $f \in \mathbf{F}_u$ needs to pre-process $\mathbf{R}_f$, $\mathbf{Q}_f$, and $w_{f,u}$. The rating vector $\mathbf{R}_f$ is additively split into two sets $\mathbf{S}_f$ and $\mathbf{T}_f$. The splitting for every rating $r_{f,b}$ is straightforward, namely choose $r \xleftarrow{\$} \mathbb{Z}_t^*$ and set $s_{f,b} = r$ and $t_{f,b} = r_{f,b} - r \mod t$. Similarly, the weight $w_{f,u}$ is split into $x_{f,u}$ and $y_{f,u}$. It is assumed that $\mathbf{T}_f$ and $\mathbf{Q}_f$ will be delivered to user u through proxy re-encryption schemes.

Running between user u and the server, the two-stage protocol is described in Fig. 2.

1. In the first stage, user u encrypt the partially-combined weights under PK_u and sends the ciphertexts to the server, which can then compute the encrypted weights between user u and all his friends. The server encrypts the random weight shares from user u's friends under PK_s and sends the ciphertexts to user u, who can then compute the encrypted weights under the server's public key.
2. In the second stage, user u and the server run some sort of two-party protocol so that the server can compute Equation (1) without learning user u's ratings and letting user u know his friends' contributions.

This protocol has exactly the same *encrypted division*, *potential information leakage through friends* and usability issues, as stated in Section 3.1. In addition, we have the following new concerns.

– *Explicit Information Disclosure*. It is assumed that the $\mathbf{Q}_f$ values for all $f \in \mathbf{F}_u$ are obtained by user u in clear. This is a direct violation of these users' privacy because it has shown that leaking what has been rated by a user can breach his privacy [34].
– *Key Recovery Attacks against the Server*. Chenal and Tang [10] have shown that given a certain number of decryption oracle queries an attacker can recover the private key of the Brakerski-Vaikuntanathan SWHE scheme. We show that user u can manipulate the protocol and recover the server's private key SK_u. Before the attack, user u sets up a fake account u' and a set of fake friends $\mathbf{F}_{u'}$ (e.g. through Sybil attacks [11]). The key recovery attack relies on multiple executions of the protocol, and it works as follows in each execution.
  1. User u' chooses a carefully-chosen ciphertext c and replaces $[z_b + \xi_{1,b}]_s$ with c. He also sets $\xi_{1,b} = 0$ for $[-\xi_{1,b}]_u$.

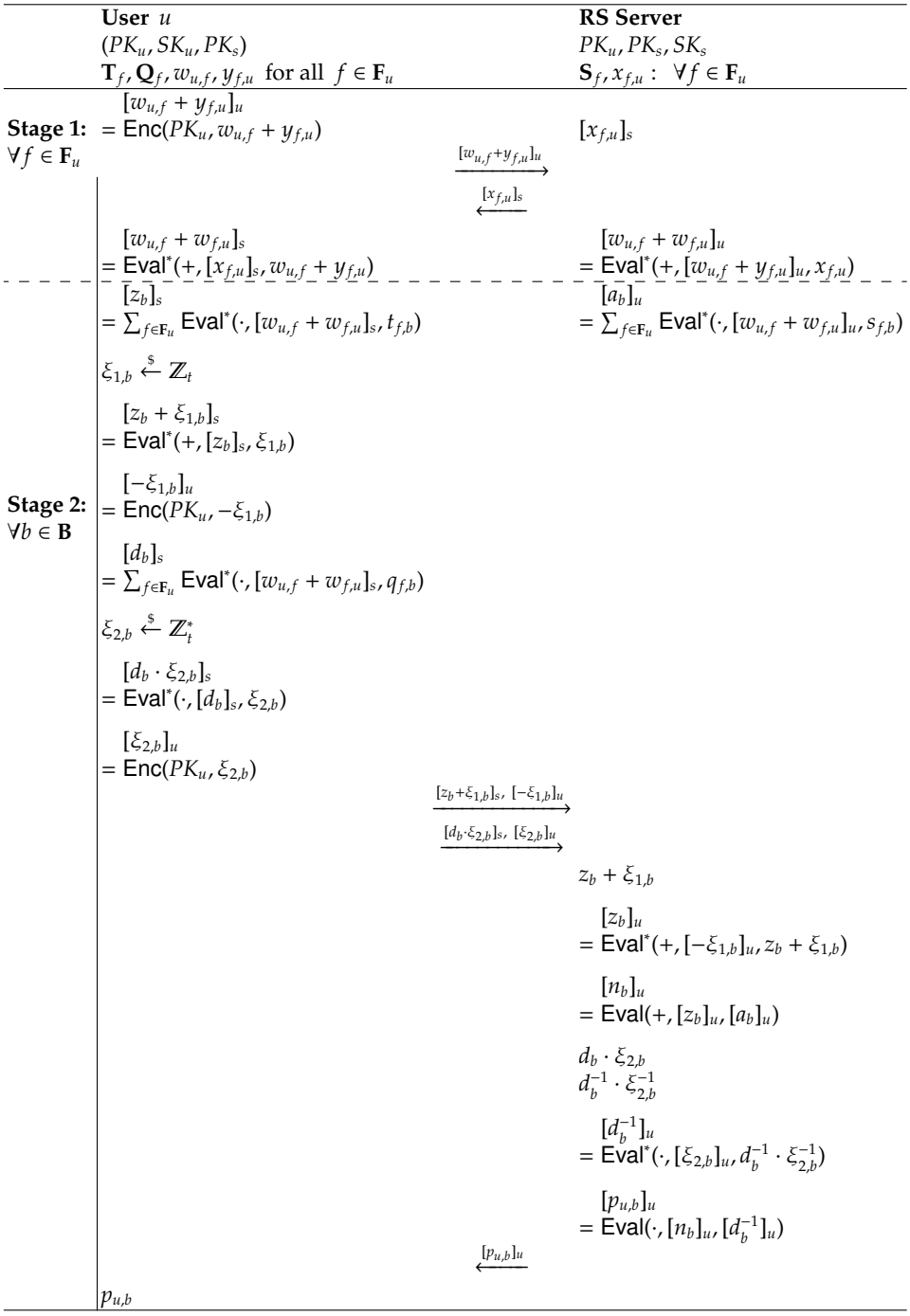

Fig. 2. JPH Offline Protocol

2. When receiving $[p_{u',b}]_{u'}$, user u' can recover the constant in $\mathsf{Dec}(SK_s, c)$ because he knows a_b and d_b^{-1} (note that user u' forged all his friends $F_{u'}$).

It is straightforward to verify that, if c is chosen according to the specifics in [10] then user u' (and user u) can recover SK_s in a polynomial number of executions. With SK_s, user u can recover the weights from his real friends in $\mathbf{F}_u$ and then infer their ratings. It is worth stressing that this attack does not violate the semi-honest assumption in [14].

If a user f has changed his rating, then he needs to update every element in $\mathbf{S}_f$. Otherwise, the server will know which item has been newly rated. Moreover, $\mathbf{T}_f, \mathbf{Q}_f$ need to be delivered to all f's friends through proxy re-encryption schemes. Suppose that, in a certain period, a subset $\mathbf{F}'_u$ of users have updated their ratings. If the server somehow learns that user u gets a high recommendation for an item b', then it can conclude that some users from $\mathbf{F}'_u$ have recently rated b'.

Remark 1. As to the aforementioned key recovery attacks concern, there is one subtlety. In the original Brakerski-Vaikuntanathan SWHE scheme, the plaintext is $\mathbb{Z}_t[x]$, while the recommendation protocol only requires the plaintext space to be $\mathbb{Z}_t$. It can be argued that the described attack can be prevented if the server performs a validity check when decrypting $[z_b + \xi_{1,b}]_s$: if the decrypted value does not fall into $\mathbb{Z}_t$ then output a decryption error. This will invalidate our attack (and the attack from [10]), but it does guarantee there is no another attack. We leave a further investigation of this issue as a future work.

4 New Formulation of Recommender System

In this section, we provide a new approach to compute recommendations and define fine-grained trust assumptions among the participants in a recommender system.

4.1 Computing Predicted Ratings

In our solution, we compute the predicted rating for user u based on inputs from both his friends and some strangers for both accuracy and security reasons. In reality friends like and consume similar items, but it might happen 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 3, we have shown that the private information of user u's friends might be leaked through user u's outputs. This is because the outputs are computed solely based on the inputs of user u's friends. It is reasonable to believe that, by taking into account some randomly chosen strangers, we will mitigate both problems.

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 Perason/Cosine similarity-based collaborative filtering schemes [18]. Let the stranger set be $\mathbf{T}_u$, the predicted value $p^*_{u,b}$ for an unrated item b is computed as follows. Note that $\overline{r_t}$ is user t's average rating, namely $\lceil \frac{\sum_{i \in \mathbf{B}} r_{t,i}}{\sum_{i \in \mathbf{B}} q_{t,i}} \rfloor$.

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

When factoring in the inputs from the friends, we make two changes to Equation (1) from Section 3.1. 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 very subjective matter for u only. Jeckmans et al. averaged the weights for the purpose of limiting information leakage [14]. 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 an unrated item b is computed as follows. Note that $\overline{r_f}$ is user f's average rating, namely $\lceil \frac{\sum_{i \in \mathbf{B}} r_{f,i}}{\sum_{i \in \mathbf{B}} q_{f,i}} \rfloor$.

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

In practice, the similarity between friends means that they tend to prefer to similar items. However, this does not implies 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 normally designed for dealing with integers, we rephrase the formula as follows, where α, β are two integers.

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

4.2 Threat Model

As to communication, we assume all communications are mediated by the RS server and the communication channels are integrity and confidentiality protected. Instead of making a general semi-honest assumption on all participants, we distinguish the following.

1. *Threat from semi-honest RS server.* In the view of all users, the RS server will follow the protocol specification but it may try to infer their private information from openly collected transaction records.
2. *Threat from a semi-honest friend.* In the view of a user, none of his friends will collude with the RS server or another party to breach his privacy. We believe 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. Nevertheless, we still need to consider possible privacy threats in two scenarios.
   - In the view of $f \in \mathbf{F}_u$, user u may attempt to learn his private information when running the recommendation protocol. In the view of user u, his friend $f \in \mathbf{F}_u$ may also try to infer his information as well.
   - In the view of $f \in \mathbf{F}_u$, user u's output (e.g. a new rated item and predicted rating value) may be leaked. If another party obtains such auxiliary information, then user f's private information may be at risk. For example, the *Potential Information Leakage through Friends* security issue in Section 3.1 falls into this scenario.
3. *Threat from strangers.* We consider the following two scenarios.
   - In the view of user u and his friends, a stranger may try to learn their private information.
   - In the view of a stranger, who is involved in the protocol execution of user u, user u may try to learn his private information.

5 New Privacy-Preserving Recommender Protocols

In this section, we propose two privacy-preserving protocols: 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. Both protocols share the same *initialization phase*.

In the initialization phase, user u generates a public/private key pair (PK_u, SK_u) for the Brakerski-Vaikuntanathan SWHE scheme and sends PK_u to the server. For the purpose of enabling strangers to validate his public key, user u asks his friends to certify his public key and puts the certification information on the server. In addition, user u 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. Besides the user-specific parameters, the global system parameters should also be established in the initialization phase. Such parameters should include α, β which determine how a predicated rating value for user u is generated based on the inputs of friends and strangers, and they should also include the minimal sizes of friend set $\mathbf{F}_u$ and stranger set $\mathbf{T}_u$.

Before describing the new protocols, it is worth stressing our *design principle* first. We require that all inputs (namely, $q_{f,b}$, $q_{t,b}$, $r_{f,b}$, $r_{t,b}$ for all $f \in \mathbf{F}_u$ and $t \in \mathbf{T}_u$) are freshly encrypted locally in each computation step. This seemingly unnecessary procedure makes the protocols immune to the vulnerabilities of the JPH online protocol (where $r_{f,b}$ and $q_{f,b}$ are directly multiplied to other ciphertexts). Moreover, instead of giving user u the predicted ratings, we only return a yes/no answer in the single prediction protocol and the Top-N list in the Top-N protocol. This reduces potential information leakages.

5.1 Recommendation Protocol for Single Prediction

When user u wants to figure out whether the predicted rating for an unrated item b is above a certain threshold τ in his mind, he initiates the protocol in Fig. 3. In more details, the protocol runs in three stages.

1. In the first stage, 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. The server first sends PK_u to some randomly chosen strangers who are the friends of user u's friends in the system[1]. Such a user t can then validate PK_u by checking whether their mutual friends have certified PK_u. After the server has successfully found a viable stranger set $\mathbf{T}_u$, it forwards $[\mathbf{I}_b]_u$ to every user in $\mathbf{T}_u$. With PK_u and $(\mathbf{R}_t, \mathbf{Q}_t)$, user t can compute the following based on the homomorphic properties. For notation purpose, assume $[\mathbf{I}_b]_u = ([\mathbf{I}_b^{(1)}]_u, \cdots, [\mathbf{I}_b^{(|B|)}]_u)$

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

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

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

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

2. In the second stage, 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, which then forwards $[w_{u,f}]_u$ and $[\mathbf{I}_b]_u$ to user f. With PK_u, $[\mathbf{I}_b]_u$, $[w_{u,f}]_u$ and $(\mathbf{R}_f, \mathbf{Q}_f)$, user f can compute the following.

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

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

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

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

3. In the third stage, user u sends his encrypted average rating $[\overline{r_u}]_u = \mathsf{Enc}(PK_u, \overline{r_u})$ to the server. The server first computes $[n_T]_u$, $[d_T]_u$, $[n_F]_u$, $[d_F]_u$ as shown in Fig. 3, and then compute $[X]_u$, $[Y]_u$ as follows.

$$temp_1 = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [d_F]_u, [\overline{r_u}]_u), [d_T]_u), \mathsf{Enc}(PK_u, \alpha + \beta))$$

[1] Due to the small world phenomenon, the population of friends of friends can already be very large (see [3] and http://sysomos.com/insidetwitter/sixdegrees/). Therefore, such kind of strangers can provide enough randomness for security. Generally, we can allow the server to choose the strangers randomly. Then, the server needs to provide a friendship path for the stranger to validate the public key PK_u and we need to assume trust somehow propagates through the chain of friends.

$$temp_2 = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [n_T]_u, [d_F]_u), \mathsf{Enc}(PK_u, \beta))$$
$$temp_3 = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [n_F]_u, [d_T]_u), \mathsf{Enc}(PK_u, \alpha))$$

$[X]_u = \mathsf{Eval}(+, \mathsf{Eval}(+, temp_1, temp_2), temp_3)$, $[Y]_u = \mathsf{Eval}(\cdot, \mathsf{Eval}(\cdot, [d_F]_u, [d_T]_u), \mathsf{Enc}(PK_u, \alpha+\beta))$

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.

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

Due to the fact that all values are encrypted under PK_u, user u needs to run a comparison protocol COM with the server to learn whether $p_{u,b} \geq \tau$. Since X, Y, τ 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 $X \geq \tau \cdot Y$. Many such protocols exist, and that by Veugen [33] is the most efficient one.

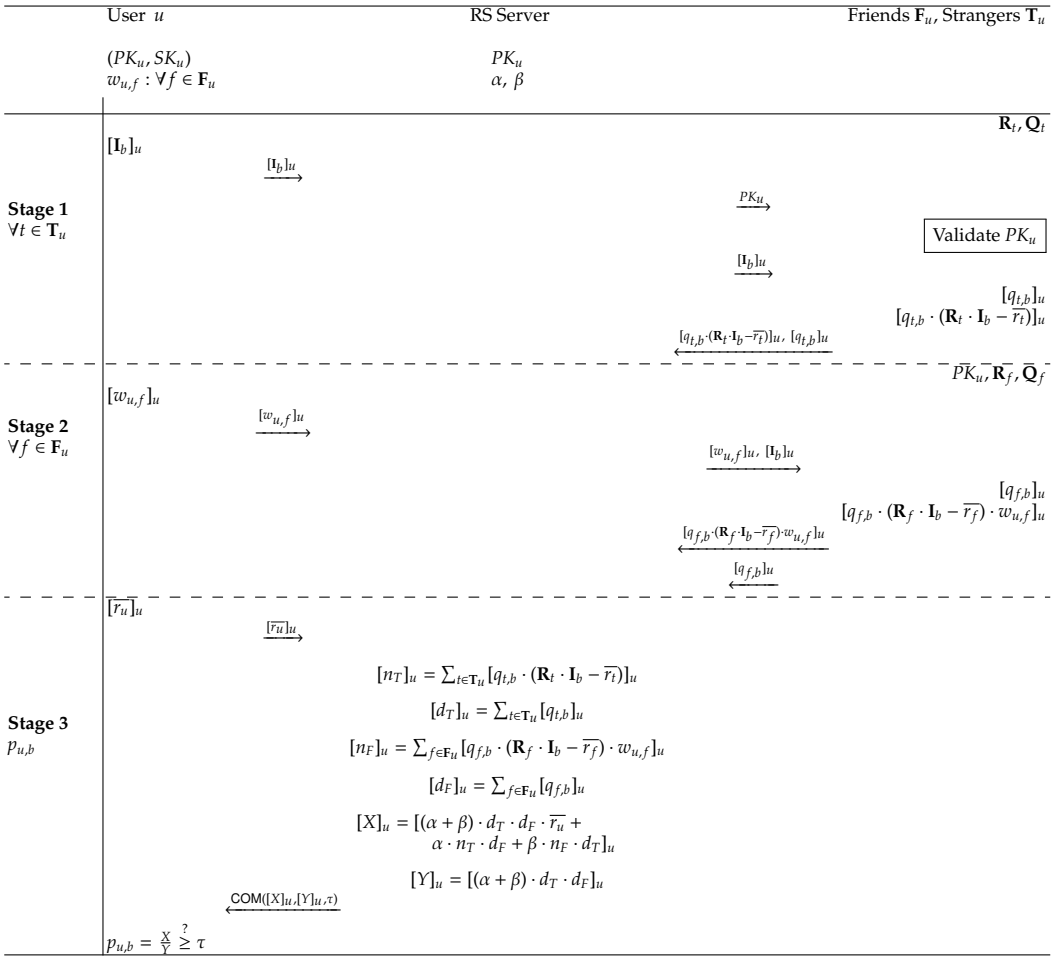

Fig. 3. Single Prediction Protocol

5.2 Recommendation Protocol for Top-N Items

When the active user u wants to figure out Top-N unrated items, he initiates the protocol in Fig. 4. In more details, the protocol runs in three stages.

1. In the first stage, the server sends PK_u to some randomly chosen strangers who can then validate PK_u as in the previous protocol. Suppose that the server has successfully found $\mathbf{T}_u$. 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 $[q_{t,b}]_u = \mathsf{Enc}(PK_u, q_{t,b})$ for every $1 \le b \le |\mathbf{B}|$. All encrypted values are sent back to the server.
2. In the second stage, to every friend $f \in \mathbf{F}_u$, user u sends the encrypted weight $[w_{u,f}]_u = \mathsf{Enc}(PK_u, w_{u,f})$. With PK_u, $[w_{u,f}]_u$ and $(\mathbf{R}_f, \mathbf{Q}_f)$, user f can compute $[q_{f,b}]_u$ and
$$[q_{f,b} \cdot (r_{f,b} - \overline{r_f}) \cdot w_{u,f}]_u = \mathsf{Eval}(\cdot, \mathsf{Enc}(PK_u, q_{f,b} \cdot (r_{f,b} - \overline{r_f})), [w_{u,f}]_u)$$
for every $1 \le b \le |\mathbf{B}|$. All encrypted values are sent back to the server.
3. In the third stage, user u generates two matrices $\mathbf{M}_X, \mathbf{M}_Y$ as follows: (1) generate a $|\mathbf{B}| \times |\mathbf{B}|$ 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.

$$\begin{bmatrix} 1 & 0 & \cdots 0 \\ 0 & 1 & \cdots 0 \\ \cdots & \cdots & \cdots \\ 0 & 0 & \cdots 1 \end{bmatrix} \xrightarrow[\text{permutation}]{\text{column}} \mathbf{M}_Y = \begin{bmatrix} 0 & 1 & \cdots 0 \\ 0 & 0 & \cdots 1 \\ \cdots & \cdots & \cdots \\ 1 & 0 & \cdots 0 \end{bmatrix} \xrightarrow[\text{rated items}]{\text{zeroing}} \mathbf{M}_X = \begin{bmatrix} 0 & 1 & \cdots 0 \\ 0 & 0 & \cdots 0 \\ \cdots & \cdots & \cdots \\ 1 & 0 & \cdots 0 \end{bmatrix}$$

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.

(a) 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 $1 \le b \le |\mathbf{B}|$ as shown in Fig. 4, in the same way as in the previous protocol. Referring to Equation (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.
$$\begin{aligned} p_{u,b} &= \frac{\beta}{\alpha+\beta} \cdot \frac{n_{T,b}}{d_{T,b}} + \frac{\alpha}{\alpha+\beta} \cdot \frac{n_{F,b}}{d_{F,b}} \\ &= \frac{\beta \cdot n_{T,b} \cdot d_{F,b} + \alpha \cdot n_{F,b} \cdot d_{T,b}}{(\alpha+\beta) \cdot d_{T,b} \cdot d_{F,b}} \\ &= \frac{X_b}{Y_b} \end{aligned}$$

(b) The server permutes the ciphertexts vector $(([X_1]_u, [Y_1]_u), ([X_2]_u, [Y_2]_u), \cdots, ([X_{|\mathbf{B}|}]_u, [Y_{|\mathbf{B}|}]_u))$ in an oblivious manner as follows.
$$([U_1]_u, [U_2]_u, \cdots, [U_{|\mathbf{B}|}]_u) = [\mathbf{M}_X]_u \cdot ([X_1]_u, [X_2]_u, \cdots, [X_{|\mathbf{B}|}]_u)^T$$
$$([V_1]_u, [V_2]_u, \cdots, [V_{|\mathbf{B}|}]_u) = [\mathbf{M}_Y]_u \cdot ([Y_1]_u, [Y_2]_u, \cdots, [Y_{|\mathbf{B}|}]_u)^T$$
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 $\mathsf{Eval}(\cdot,,)$ 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 $U_i = 0$ because the element 1 in b-th column has been set to 0.

(c) Based on some COM protocol, e.g. that used in the previous protocol, the server ranks $\frac{U_i}{V_i}$ $(1 \le i \le |\mathbf{B}|)$ in the encrypted form using any standard ranking algorithm, where comparisons are done interactively with user u through the encrypted integer comparison protocol COM.

(d) 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.

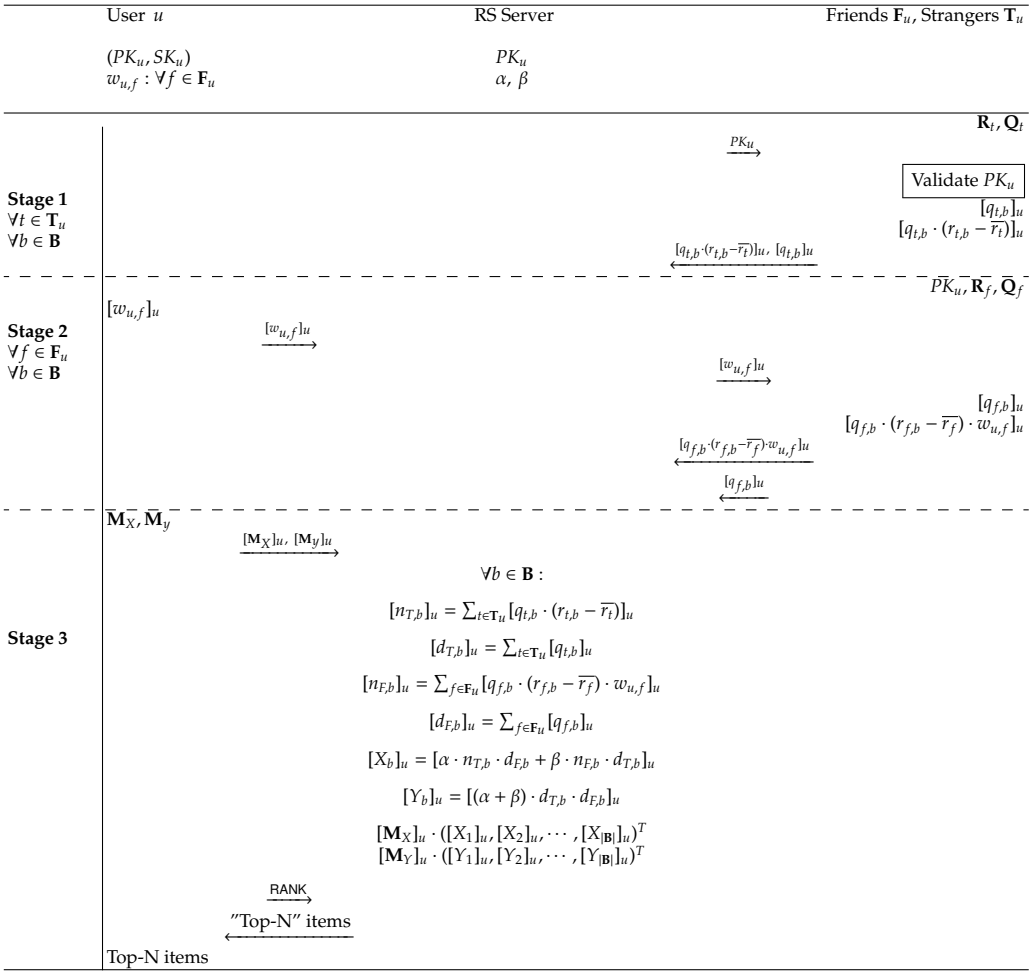

Fig. 4. Top-N Protocol

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.

6 Evaluating the Proposed Protocols

From the analysis in Section 3.2, the JPH offline protocol has a number of security and usability issues. In particular, on-the-fly updating personal information at the server and for his friends is a very tedious task in practice. By incorporating inputs from strangers, our protocols can generate recommendations when many friends are offline. The experiment results, shown in next subsection, show that we can approximately achieve the functionality of JPH offline protocol but without incurring these issues.

6.1 Parameters and Performances

The selection of the global parameters α, β and the sizes of $\mathbf{F}_u$ and $\mathbf{T}_u$ can affect the security, in particular when considering the threat from a semi-trusted friend. 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 outputs of user u. This will in turn make information reference attacks easier (for user u to infer the inputs of his friends). However, if $\frac{\alpha}{\alpha+\beta}$ gets smaller and $\mathbf{T}_u$ gets larger, then we will lose the motivation of explicitly distinguishing friends and strangers in computing recommendations, namely the accuracy of recommendations may get worse. How to choose these parameters will depend on the application scenarios and the overall distributions of users' ratings.

In order to get some rough idea about how these parameters influence the accuracy of recommendation results. We choose the MovieLens 100k dataset [2] 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[3]. For different parameters, the Mean Average Error (MAE) [30] of the proposed protocols is shown in Table 1. Note that lower MAE implies more accurate recommendations.

$(\lvert\mathbf{F}_u\rvert, \lvert\mathbf{T}_u\rvert)$ \ $\frac{\alpha}{\alpha+\beta}$	0.5	0.6	0.7	0.8	0.9	1.0
(10, 10)	0.8222	0.8168	0.8158	0.8193	0.8265	0.8444
(20, 10)	0.8075	0.7996	0.7964	0.8009	0.8077	0.8194
(30, 10)	0.8033	0.7896	0.7819	0.7849	0.7871	0.7999
(40, 10)	0.7965	0.7801	0.7728	0.7695	0.7755	0.7814
(50, 10)	0.7909	0.7762	0.7666	0.7623	0.7623	0.7684
(60, 10)	0.7840	0.7722	0.7604	0.7552	0.7559	0.7588
(70, 10)	0.7824	0.7685	0.7538	0.7483	0.7460	0.7506
(80, 10)	0.7818	0.7643	0.7540	0.7432	0.7390	0.7432
(90, 10)	0.7785	0.7598	0.7494	0.7419	0.7356	0.7378
(100, 10)	0.7757	0.7559	0.7448	0.7390	0.7338	0.7312

Table 1. MAE of Experiments

From the numbers, it is clear that the more friends are involved the more accurate recommendation results user u will obtain (i.e. the MAE is lower). There is also a trend that the MAE becomes smaller when the contribution factor $\frac{\alpha}{\alpha+\beta}$ becomes larger. According to the accuracy results by Lemire and Maclachlan (in Table 1 of [18] where the values are MAE divided by 4), their smallest MAE is $0.752 = 0.188 \times 4$. From the above Table 1, we can easily get lower MAE when $|F_u| \geq 70$ by adjusting $\frac{\alpha}{\alpha+\beta}$.

With respect to the computational complexity of the proposed protocols, we simply count the number of different computations required. For the single prediction protocol, the numbers are listed in Table 2. There is the additional cost of COM protocol between user u and the server. For the Top-N protocol, the numbers are listed in Table 3. There is the additional cost of RANK protocol between user u and the server. One interesting fact is that, in order to hide the item b, the single prediction protocol requires more computations from the strangers than the Top-N protocol. We leave a further investigation of the computational performances as a future work.

[2] http://grouplens.org/datasets/movielens/

[3] The algorithms are here: https://github.com/thelakers/moophs/blob/master/201504-pp-context-aware-keyfunctions.py. This might not be very realistic, as the user's subjective weights may be different from the similarity calculated based on rating data. Later, we will perform a field test with really users to validate this.

$(\lvert\mathbf{F}_u\rvert, \lvert\mathbf{T}_u\rvert)$ \ $\frac{\alpha}{\alpha+\beta}$	Enc	Eval(+,,)	Eval(·,,)
Friend	1+3$\lvert B\rvert$	3$\lvert B\rvert - 2$	3$\lvert B\rvert + 2$
Stranger	1+3$\lvert B\rvert$	3$\lvert B\rvert - 2$	3$\lvert B\rvert + 1$
Server	3	$2\lvert T_u\rvert + 2\lvert F_u\rvert - 2$	9
User u	$\lvert B\rvert + \lvert F_u\rvert + 1$	0	0

Table 2. (Partial) Complexity of Single Prediction Protocol

$(\lvert\mathbf{F}_u\rvert, \lvert\mathbf{T}_u\rvert)$ \ $\frac{\alpha}{\alpha+\beta}$	Enc	Eval(+,,)	Eval(·,,)
Friend	2$\lvert B\rvert$	0	$\lvert B\rvert$
Stranger	2$\lvert B\rvert$	0	0
Server	3	$2\lvert T_u\rvert+2\lvert F_u\rvert+\lvert B\rvert^2-\lvert B\rvert-2$	$\lvert B\rvert^2 + 6$
User u	$2\lvert B\rvert^2 + \lvert F_u\rvert$	0	0

Table 3. (Partial) Complexity of Top-N Protocol

6.2 Security Analysis

Informally, the protocols are secure due to two facts: (1) all inputs are first freshly encrypted and then used in the computations; (2) all computations (e.g. computing predictions and ranking) done by the server and other users are in the encrypted form. As to the single prediction protocol in Section 5.1, we have the following arguments.

1. *Threat from semi-honest RS server.* Given the COM protocol is secure (namely, the server does not learn anything in the process). Then the server learns nothing about any user's private input information, e.g. $b, \tau, \mathbf{R}_u, \mathbf{Q}_u, \mathbf{R}_f, \mathbf{Q}_f, w_{u,f}, \mathbf{R}_t, \mathbf{Q}_t$ for all f and t, because every element is freshly encrypted in the computation and all left computations are done homomorphically. Moreover, the server learns nothing about $p_{u,b} \overset{?}{\geq} \tau$ based on the security of COM.
2. *Threat from a semi-honest friend.* We consider two scenarios.
   – Informally, a friend f's contribution to $p_{u,b}$ is protected by the inputs from users $\mathbf{F}_u \backslash f$ and the strangers $\mathbf{T}_u$. Given a randomly chosen unrated item for user u and a randomly chosen friend $f \in \mathbf{F}_u$, we perform a simple experiment to show how f's input influences the predicted rating. We set $\frac{\alpha}{\alpha+\beta} = 0.8$ and the $(\lvert\mathbf{F}_u\rvert, \lvert\mathbf{T}_u\rvert) = (30, 10)$ in all tests, and choose strangers randomly in every test.

Rating Value \ Tests	Test 1	Test 2	Test 3	Test 4	Test 5
With f's input (r)	4.0351	3.7165	3.9125	3.9125	4.0667
Without f's input (r')	3.7014	4.0343	3.9125	3.9698	3.9556
$r - r'$	0.3337	-0.3178	0.0000	-0.0573	0.1111

Table 4. Influence of a Single Friend

The results in 4 imply that a friend f's contribution to user u's output is obfuscated by the inputs from the stranger set. Simply from the output of user u, it is hard to infer user f's input. Furthermore, it should be clear that the larger the friend set is the less information of a single friend will be inferred. More test results appear in Appendix I. Due to the encryption, the friends learn nothing about user u.

– For similar reasons, it will be hard for $\mathbf{F}_u \backslash f$ to infer user f's data even if they learned user u's output at the end of a protocol execution.

3. *Threat from strangers.* We consider the following two scenarios.
   – In the view of strangers, all values are encrypted under user u's public key, so that they will not be able to derive any information about the inputs and outputs of user u and his friends.
   – For the strangers involved in a protocol execution, it does not leak much information for several reasons. Firstly, user u does not know which stranger is involved in the protocol execution. Secondly, the inputs of a group strangers are blended in the output to user u. We perform a simple experiment to show how strangers' inputs influence the predicted ratings for user u. We set $\frac{\alpha}{\alpha+\beta} = 0.8$ and the $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (30, 10)$. Table 5 shows the rating differences for 5 unrated items, depending on whether a stranger is involved in the computational or not. It is clear that very little information about a stranger can be inferred from user u's outputs.

Unrated Items / Strangers	Item_1	Item_2	Item_3	Item_4	Item_5
Stranger_1	0.0	-0.0825	0.0	0.0	0.0
Stranger_2	0.0	0.0	0.0	0.0	0.0
Stranger_3	0.0	0.0211	0.0	0.0	0.0
Stranger_4	0.0913,	0.0	0.0	0.0	0.0
Stranger_5	0.0	0.0134	0.0702	0.0	0.1375

Table 5. Influence of Strangers

Thirdly, the strangers are independently chosen in different protocol executions, so that it is difficult to leverage on the accumulated information.

The same analysis applies to the Top-N protocol in Section 5.2. As to user u's outputs, the matrices $[\mathbf{M}_X]_u, [\mathbf{M}_Y]_u$ randomly permuted the predictions so that the ranking does not leak any information about the Top-N items.

7 Conclusion

Recommender systems are complex in the sense that many users are involved and contributing to the outputs of each other. The privacy challenge is big because it is difficult to reach a realistic security model with efficient privacy-preserving protocols. This work, motivated by [14], tried to propose a realistic security model by leveraging on the similarity and trust between friends in digital communities. Compared to [14], we went a step further by introducing randomly selected strangers into the play and make it possible to protect users' privacy even if their friends' outputs are compromised. Moreover, we adjusted the recommendation formula and achieve better accuracy than some other well-known recommender protocols [18]. Following our work, many interesting topics remain open. One is to test our protocols on real dataset. Another is to implement the protocols and see how realistic the computational performances are. Another is to adjust the recommendation formula to reflect more advanced algorithms, such as Matrix Factorizations [20], which however will have different requirements on the involved user population. Another is to investigate stronger security models, e.g. assuming a malicious RS server. Yet another topic is to formally investigate the information leakages from the outputs. Our methodology, namely introducing randomly selected strangers, has some similarity with the differential privacy based approach [19]. A detailed comparative study will be very useful to understand their connections.

Acknowledgements

The authors are supported by a CORE (junior track) grant from the National Research Fund, Luxembourg.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
2. 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.
3. 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.
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. A. Bilge and H. Polat. A scalable privacy-preserving recommendation scheme via bisecting k-means clustering. *Information Processing & Management*, 49(4):912 – 927, 2013.
6. 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.
7. 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.
8. J. F. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.
9. J. F. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, 2002.
10. M. Chenal and Q. Tang. On key recovery attacks against existing somewhat homomorphic encryption schemes. In *Progress in Cryptology – LATINCRYPT 2014*, page to appear, 2014.
11. J. R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002*, volume 2429 of *LNCS*, pages 251–260. Springer, 2002.
12. Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk. Efficiently computing private recommendations. In *International Conference on Acoustic, Speech and Signal Processing*, 2011.
13. S. Han, W. K. Ng, and P. S. Yu. Privacy-preserving singular value decomposition. In Y. E. Ioannidis, D. L. Lee, and R. T. Ng, editors, *Proceedings of the 25th International Conference on Data Engineering*, pages 1267–1270. IEEE, 2009.
14. 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.
15. A. Jeckmans, Q. Tang, and P. Hartel. Privacy-preserving collaborative filtering based on horizontally partitioned dataset. In *2012 International Symposium on Security in Collaboration Technologies and Systems (CTS 2012)*, pages 439–446, 2012.
16. M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy. In *The Tenth ACM SIGMOD International Conference on Knowledge Discovery and Data Mining*, pages 599–604. ACM, 2004.
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. 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.
19. 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.
20. 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.
21. R. Parameswaran. *A robust data obfuscation approach for privacy preserving collaborative filtering*. PhD thesis, Georgia Institute of Technology, 2006.

22. 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.
23. H. Polat and W. Du. Privacy-preserving collaborative filtering. *International journal of electronic commerce*, 9:9–36, 2005.
24. H. Polat and W. Du. Privacy-preserving collaborative filtering on vertically partitioned data. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 3721 of *LNCS*, pages 651–658, 2005.
25. H. Polat and W. Du. Privacy-preserving top-n recommendation on horizontally partitioned data. In *2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005)*, pages 725–731. IEEE Computer Society, 2005.
26. H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC)*, pages 791–795. ACM, 2005.
27. H. Polat and W. Du. Achieving private recommendations using randomized response techniques. In *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006*, pages 637–646. Springer, 2006.
28. H. Polat and W. Du. Privacy-preserving top-N recommendation on distributed data. *J. Am. Soc. Inf. Sci. Technol.*, 59:1093–1108, 2008.
29. N. Ramakrishnan, B.J. Keller, B.J. Mirza, and A. Y. Grama. Privacy risks in recommender systems. *Internet Computing, IEEE*, 5:54–63, 2001.
30. G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer, 2011.
31. 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.
32. 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.
33. T. Veugen. Comparing encrypted data. http://bioinformatics.tudelft.nl/sites/default/files/Comparing2011.
34. 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.
35. I. Yakut and H. Polat. Arbitrarily distributed data-based recommendations with privacy. *Data & Knowledge Engineering*, 72(0):239 – 256, 2012.
36. J. Zhan, C. Hsieh, I. Wang, T. Hsu, C. Liau, and D. Wang. Privacy-preserving collaborative recommender systems. *Trans. Sys. Man Cyber Part C*, 40:472–476, 2010.

Appendix I: More Performance Tests

For every friend f_i of user u, randomly choose an unrated item, we compute the predicted rating difference with and without f's input. We set $\frac{\alpha}{\alpha+\beta} = 0.8$ and the $(|\mathbf{F}_u|, |\mathbf{T}_u|) = (30, 10)$ in all tests, but the strangers are chosen randomly in every test.

Tests / Rating Differences	Test 1	Test 2	Test 3	Test 4	Test 5
W.r.t f_1	-0.0616	0.0549	0.0593	-0.2166	0.036
W.r.t f_2	-0.0057	-0.2945	0.0	0.0	0.0993
W.r.t f_3	0.07080	0.0	0.2037	-0.0351	-0.1788
W.r.t f_4	-0.1658	-0.024	0.0124	-0.0558	0.0325
W.r.t f_5	-0.2477	-0.004	-0.0671	0.0302	0.2273

Table 6. Influence of Friends