Efficient Privacy-Enhanced Familiarity-Based Recommender System

Arjan Jeckmans, Andreas Peter, Pieter Hartel

Distributed and Embedded Security Group, University of Twente
{a.j.p.jeckmans,a.peter,pieter.hartel}@utwente.nl

Abstract. Recommender systems can help users to find interesting content, often based on similarity with other users. However, studies have shown that in some cases familiarity gives comparable results to similarity. Using familiarity has the added bonus of increasing privacy between users and utilizing a smaller dataset. In this paper, we propose an efficient privacy-enhanced recommender system that is based on familiarity. It is built on top of any given social network (without changing its behaviour) that already has information about the social relations between users. Using secure multi-party computation techniques and somewhat homomorphic encryption the privacy of the users can be ensured, assuming honest-but-curious participants. Two different solutions are given, one where all users are online, and another where most users are offline. Initial results on a prototype and a dataset of 50 familiar users and 1000 items show a recommendation time of four minutes for the solution with online users and of five minutes for the solution with offline users.

1 Introduction

Recommender systems can help users to find interesting content, for example a movie to watch, or books to buy. These recommender systems often rely on a large database of information from a lot of different users. With such a database the systems then recommend content based on similarity (agreement in rating behaviour) between users. However, studies [11, 12, 17, 25] have shown that for taste related domains, such as movies and books, familiarity (social closeness between users) gives comparable accuracy to using similarity. Familiarity captures how well users know each other (and thus their preferences). Using familiarity instead of similarity removes the information need from unknown users, thus increasing privacy between users. Since no information from unknown users is needed, a recommender system based on familiarity also works on a smaller dataset, leading to a higher efficiency. In this paper we focus on the generation of recommendations using only familiarity. We leave as future work, a recommender system that combines both similarity and familiarity.

As a pre-requisite for a familiarity-based recommender system, a familiarity network needs to be known to the recommendation provider. Since this familiarity information is already present in online social networks, we can leverage these networks to provide recommendations. Our aim is to build a recommendation

system on top of existing social networks (utilizing the familiarity relationship that is present), while preventing the social network from learning the users' taste preferences (not giving the social network any information that it does not have already).

While the general tastes (and possibly some specific tastes) of friends are known, the exact details of a friend's complete taste are usually not known. Revealing a specific taste to friends can be embarrassing [21] as it does not conform to the group norm, or to the societal norm as a whole. For example, if all friends of a person dislike 'The Hunger Games', but that person loves the book, if the friends find out this could be embarrassing. As such, the privacy of the user with regards to their taste needs to be protected from both friends (specific taste) and the online social network (general and specific taste).

To ensure the privacy of the users, we make use of secure multi-party computation and a somewhat homomorphic encryption scheme. The motivation for a somewhat homomorphic encryption scheme (we use [4]) is: 1) it allows us to do a (bounded) number of additions and at least one multiplication on encrypted data, and 2) the message space is pre-determined by public parameters and is the same across keypairs. The latter property allows for blinding values under one key and unblinding under another.

In constructing our solution, next to privacy, we focus on the efficiency of the solution. Our contribution is the following: First, we look at the privacy that the weighted average recommendation formula can give to the user and friends. We observe that weighted average based on user supplied weights does not provide enough privacy. Based on this, we propose an adjusted formula that offers more privacy. Second, utilizing this adjusted formula, we construct a protocol that computes the recommendation for a user, when all his friends are online. However, users are not guaranteed to be online in a social network. Third, as users can be offline, we also construct a protocol where the users friends are offline, and the user works together with the social network server to compute the recommendations. Not having to wait for all friends to have been online to do their part in the protocol increases the efficiency of the solution. Both protocols are secure, assuming honest-but-curious participants.

In this paper, we will use books as our running example for recommendations. The paper is structured as follows: Section 2 details the state of the art and related work. Section 3 gives the problem specification and details the adjusted recommendation formula. Section 4 outlines the cryptographic primitives that are used. Section 5 details the solution with online friends and the solution with offline friends. Section 6 analyzes the solutions, both in terms of privacy and efficiency. And Section 7 gives concluding remarks with regard to the solutions.

2 Related Work

In this section, we show related work in privacy protecting recommender systems that protect privacy through the use of cryptography and multi-party computation. In 2002, Canny [5] proposed using additive homomorphic encryption

to privately compute intermediate values of the collaborative filtering process. These intermediate values are made public and used in singular value decomposition and factor analysis, which leads to recommendations. However, the presented approach suffers from a heavy computational and communication overhead. Moreover, due to the nature of the used recommender system (singular value decomposition), users cannot input their familiarity information.

Hoens et al. [14] designed a privacy preserving recommender system for social networks that computes the weighted average rating for items. It gathers input from friends and friends of friends and onwards by first defining a group of users involved in the computation. Then a threshold homomorphic cryptosystem is set up. This cryptosystem, together with multi-party computation, is used to compute the weighted average. The weights are defined by the user for his friends, and by the friends for the friends of friends, and so on. Privacy is achieved through both cryptographic protocols as well as anonymity through multiple participants. The downsides of this solution are the requirement that users are online, the setup of a big group in advance, and the heavy computational load in the order of hours. Hoens et al. [15] designed a private recommender system for doctors, where patient ratings are aggregated. In this scenario, there is not a predefined group of patients and no weights are given to individual ratings or patients. Hoens et al. offer two solutions, one based on anonymized ratings, and one based on cryptography and multi-party computation. Again, the timing of the solution based on cryptography is in the order of hours.

Basu et al. [2] proposed a privacy preserving version of the slope one predictor, using a threshold additive homomorphic cryptosystem. In their scenario, different parties hold different parts of the data. In a social network setting, this means that each friend holds his own data. The parties pre-compute the deviation and cardinality matrices under encryption and make the cardinality matrix public. Then the prediction for a single item can be computed under encryption and all parties collaborate to decrypt the result. Their timing information, in the order of seconds, is based on a prediction for a single user and single book. This is after pre-computation in the order of hours. There is no support for offline users, nor for familiarity due to the way predictions are computed.

Erkin et al. [9] proposed a collaborative filtering algorithm based on additive homomorphic cryptosystems. This algorithm requires a second semi-trusted server to allow for users to be offline. However, in practical scenarios such a server is usually not available. The protocol of Erkin et al. does not give weights to the ratings. Runtime is in the order of minutes for a dataset of 1000 items and several thousand (variable) users.

Jeckmans et al. [16] proposed to use collaborative servers as a way to allow for offline users. A user can choose a trusted server, that will preserve the privacy on his behalf. The trusted server knows the user's ratings and thus the user has no privacy from this server. This trusted server can then collaborate with another server to increase the accuracy of the recommendations, without losing the privacy of its users. However, this is not a desirable solution for every scenario. In such a distributed setting, it becomes difficult for users to give weights

to friends, when friends are on different servers. The runtime of the protocol is in the order of minutes, and does not involve any user interaction, including the user for which the prediction is made.

3 The Problem Specification

We consider the following problem scenario: With an online social network as a basis, how can users use/share the taste information from/with their friends, without leading to undesired disclosure of specific tastes. The following subsections go into more detail about the entities and their relationship, the suggested method of using the taste information, and what undesired disclosure is.

3.1 Architecture

The system consists of three entities:

- the user, for whom a prediction has to be generated,
- the online social network, also denoted as the server, acting as a gateway to access the user's friends and assisting in the prediction computation, and
- the friends of the user, giving their opinions as input for the book predictions.

Because of the nature of online social networks, not all friends will be online when the request for book scores is made. Because the user is unlikely to want to wait until all friends have come online, the online social network acts as an intermediate for the user's friends (while not learning information about the friends' preferences). As such, we distinguish two scenarios; book recommendation when the user's friends are all online, and book recommendation when the user's friends are all offline. It is also possible that some friends of the user are online, while some are offline. For simplicity we take this third scenario to be equal to book recommendation when the user's friends are all offline.

3.2 Recommendation Formula

Before predictions can be made, the familiarity between users has to be captured. Towards this end, the user can score his friends on their familiarity (social closeness) and the expected overlap in reading habits. Scoring a friend essentially gives that friend a weight that determines how heavy his opinion counts towards a specific book recommendation. Based on the friends' ratings for books and the weight for each friend, the recommender system predicts a score for each book. This helps the user to select the next book to read.

A book prediction is denoted by $p_{u,b}$, for user $u, 1 \leq u \leq U$, of book $b, 1 \leq b \leq B$, where U is the total number of users and B is the total number of books. The recommendation formula is as follows:

$$p_{u,b} = \frac{\sum_{f=1}^{F_u} q_{f,b} \cdot r_{f,b} \cdot w_{u,f}}{\sum_{f=1}^{F_u} q_{f,b} \cdot w_{u,f}}, \tag{1}$$

where F_u is the number of friends of a user u, $q_{f,b}$ is 1 if friend f rated book b and 0 otherwise, $r_{f,b}$ the rating of friend f for book b, and $w_{u,f}$ the weight given by the user u to friend f. The indication, $q_{f,b}$, if a book b has been rated a friend f is either 0 or 1, $q_{f,b} \in \{0, 1\}$. The range of the prediction, $p_{u,b}$, is equal to the range of the ratings given to a book, $r_{f,b}$. For example, this range can be between 0 and 5 for a 0 to 5 star rating system. The weight given to a friend, $w_{u,f}$, can be in the range between 0 and 1 excluding 0, as 0 would indicate no friendship. This formula has been used in previous research in similarity-based [13], familiarity-based [11] and trust-based recommendation systems [26].

However, when looking at the inherent privacy this formula can give us, we notice two things:

1. Due to the fact that the user u learns the predictions $p_{u,b}$ and determines the weights $w_{u,f}$, with two prediction requests the user can learn which books are rated by one friend, i.e. learn $q_{f,b}$. This is accomplished by changing the weight $w_{u,f}$ for that specific friend. For example, suppose that the user has three friends who have rated two books. The first friend rated the first book with a 5, the second friend rated both books with a 4, and the third friend rated the second book with a 3. When the user request a prediction with all weights set to 1, he will receive a prediction of 4.5 for the first book and 3.5 for the second book. Next, the user requests a prediction with the weights of the first and second friend set to 1, and the weight of the third friend set to 0.5. He will receive a prediction of 4.5 for the first book and 3.67 for the second book, thus he learns that the third friend rated the second book. Given enough runs, the user can learn $q_{f,b}$ for all his friends.
2. Because the user knows $p_{u,b}$, $w_{u,f}$, and $q_{f,b}$, the only unknown values are that of $r_{f,b}$. Given enough runs, the user can also compute $r_{f,b}$ and completely breach the privacy of his friends.

Consequently, when using this formula, we cannot achieve privacy at all. Intuitively, the user has too much control, and the friends have no input beyond their fixed ratings. This asymmetry in the formula leads to an asymmetrical relationship between the user and his friends. As stated by Carley and Krackhardt [6], friendship is not necessarily symmetric, but tends in the direction of symmetry. In general, long strong friendships are symmetric, and newly forged friendships are not symmetric. As such, we aim to bring symmetry to the recommendation formula and balance out the power in the relationship between the user and his friends.

Since the weight from the user to his friends is asymmetrical, we propose to make the weight, and thus the formula, symmetrical. This is accomplished by taking the average of the weight from the user to his friend and the weight from the friend to the user. This results in the following formula:

$$p_{u,b} = \frac{\sum_{f=1}^{F_u} q_{f,b} \cdot r_{f,b} \cdot (\frac{w_{u,f}+w_{f,u}}{2})}{\sum_{f=1}^{F_u} q_{f,b} \cdot (\frac{w_{u,f}+w_{f,u}}{2})}, \tag{2}$$

where $w_{f,u}$ is the weight given by friend f to user u, with range between 0 and 1 excluding 0. Note that this also requires a bi-directional relationship between

the friends. When looking back to the two points made before in light of this adjusted formula, we can say:

1. Since the user can still change the weights that are given to his friends $w_{u,f}$, the user can influence the averaged weight, $\frac{w_{u,f}+w_{f,u}}{2}$. Based on the changed weights and change in predictions, the user can still determine $q_{f,b}$ as before.
2. When the user knows $p_{u,b}$, $w_{u,f}$, and $q_{f,b}$, the values for $r_{f,b}$ and $w_{f,u}$ remain unknown. The fact that both the upper and lower part of the prediction formula remain unknown greatly increases the difficulty of breaching privacy.

To prevent the user from learning $q_{f,b}$, the user's influence on the weight can be removed. However, then this recommender system would lose the user's control and reduce the value of the predictions. Instead, we refer to profile aggregation methods [24], methods that add random ratings [8, 20], or methods that add randomness to the output [22]. These solutions can be applied *independent* of our solution and will not be addressed in this paper.

Note that the impact on accuracy of this adjusted formula has not been determined. As this paper focusses on privacy and efficiency, and a suitable dataset to test accuracy could not be found, we leave this as future work.

3.3 Security Model

Both the user and his friends are considered to be honest-but-curious; they will follow the protocol but try to learn the taste of their friends. More specifically, the user u will try to learn $r_{f,b}$ and $w_{f,u}$, while the friends of u will try to learn $w_{u,f}$.

We also assume that the social network server is honest-but-curious; the server will follow the protocol, while trying to learn the tastes of users. The server will try to learn $q_{f,b}$, $r_{f,b}$, $w_{u,f}$, $w_{f,u}$, and $p_{u,b}$. We assume that the users do not collude with the server, as they do not want to impact the privacy of their friends too much.

4 Cryptographic Primitives

To build our solutions, we make use of the cryptographic primitives described in this section. The primitives of additive secret sharing and proxy re-encryption are only used in the solution with offline friends.

4.1 Somewhat Homomorphic Encryption

To protect information during the protocol, we use the somewhat homomorphic encryption scheme of Brakerski and Vaikuntanathan [4]. Specifically, we use that this somewhat homomorphic encryption scheme allows both addition and multiplication of the encrypted messages (though a limited, but configurable amount), and the fact that the message space is the same across multiple key pairs (given the same public parameters).

In the setup phase of the encryption system, the public parameters are chosen. Among others, these are: the message space (which equals $\mathbb{Z}_t$ for some prime number t), the encrypted messages (which are represented in the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ of polynomials over $\mathbb{Z}_q$ for some prime number q, where the polynomial $f(x)$ is cyclotomic and of degree n), and the degree D of allowed homomorphism (which indicates the amount of multiplications that can occur under encryption). The choice of the ring R_q in relation to the prime t and degree of homomorphism D defines the security of the encryption system.

Each party can, based on these public parameters, create a key pair consisting of the secret key SK and the public key PK. The secret key is randomly chosen and the public key is based on the secret key and some randomness. The public key of user u is denoted by PK_u. Given an encryption of m under the public key PK_u, denoted by $[m]_u$, the following homomorphic properties hold (until the error overflows, typically when the degree D has been reached): $[m_1]_u + m_2 = [m_1 + m_2]_u$, $[m_1]_u + [m_2]_u = [m_1 + m_2]_u$, $[m_1]_u \cdot m_2 = [m_1 \cdot m_2]_u$, $[m_1]_u \cdot [m_2]_u = [m_1 \cdot m_2]_u$.

This scheme is semantically secure under the polynomial learning with errors assumption. For more details, we refer to the work of Brakerski and Vaikuntanathan [4].

4.2 Encrypted Division

Because the homomorphic encryption system can only encrypt integers, and thus only operate on integers, division of encrypted values is not straightforward. For example $[5]/[2] \neq [2.5]$ as $[2.5]$ cannot be represented as such. Given that the message space $\mathbb{Z}_t$ is known and the range of the predictions $p_{u,b}$ is also known and significantly smaller, a lookup table can be constructed (and precomputed) to quickly translate the integers after division into the actual fractions they represent. The lookup table looks like this: given two integers x and y, with $\gcd(x, y) = 1$ and x/y as a possible result for $p_{u,b}$, the index is $x \cdot y^{-1} \mod t$ and the resulting value x/y. For integers x' and y' with $\gcd(x', y') \neq 1$, the division result is the same as for $x = x'/\gcd(x', y')$ and $y = y'/\gcd(x', y')$. We denote the set of possible integers for x, X, the set of possible integers for y, Y, and the range of possible predictions $p_{u,b}$, P. The lookup table then has size $|\{x/y \mid \gcd(x, y) = 1, x/y \in P, x \in X, y \in Y\}|$. The size of the lookup table is upper bounded by the size of the message space $\mathbb{Z}_t$. As such, division can happen under encryption and after decryption a table lookup retrieves the actual result.

4.3 Additive Secret Sharing

An alternate method to protect information from multiple parties, while still providing operations on that information, is additive secret sharing [10]. Unlike encryption, where only the party with the key can decrypt it, anybody with enough shares can extract the information. Distribution of the shares prevents extraction of the information, but still allows us to run a protocol to use the information. When a party has a value x that it wants to protect, it creates a

random value $r \in_R \mathbb{Z}_k$, where k is a security parameter. The party then creates $s = x - r$. It can give r to a second party, and s to a third. Together the second and third party can reconstruct x by $x = r + s$.

It is also possible to secret share a vector of values, X, of length n. The secret sharing algorithm is then applied to each element of X individually, resulting in the two vectors R and S, both of length n. When combined the vectors R and S sum up to the vector X, $x_i = r_i + s_i$, where $1 \leq i \leq n$.

4.4 Proxy Re-encryption

To share information between two users of the social network without a direct connection, we use proxy re-encryption [3]. Proxy re-encryption allows us to send a (secret) message from one user to his friends through the social network. In proxy re-encryption, based on the keys of two users a re-encryption key can be derived. This re-encryption key is then given to the proxy (the social network server). When given a message encrypted under the key of one user, using the re-encryption key the proxy can translate the message, to a message encrypted under the key of the second user. This way an offline user can store his information on the social network encrypted under his own key. When a friend requires access to that information, the server can translate the information to be encrypted under the key of the friend (provided a re-encryption key has been setup). The friend can then decrypt and use the information left by the offline user.

We require that the re-encryption scheme is unidirectional.In a unidirectional scheme the users do not have to share their private keys to create a re-encryption key. To create a re-encryption key from the user to a friend, only the user's private key and the friend's public key are needed. We further require that the re-encryption scheme is one-hop only, so that only friends of the user can read his information. Some examples of schemes that satisfy these requirements are: Ateniese et al. [1], Libert and Vergnaud [18], and Chow et al. [7]. The proxy re-encryption scheme can be chosen independent of our protocol and is only used to give the friends' information to the user beforehand.

5 Proposed Solutions

In this section we provide the details of the protocols to compute the book recommendations. A protocol is given when all friends are online, and a protocol is given when all friends are offline. For convenience, we make some small cosmetic alterations to the prediction formula 2. *We set the value of $r_{f,b}$ to 0 when $q_{f,b} = 0$, thus $r_{f,b}$ becomes equal to $q_{f,b} \cdot r_{f,b}$. We also divide $w_{u,f}$ and $w_{f,u}$ by 2 before running the protocols (without renaming), remove the need to divide by 2 during the protocol.*

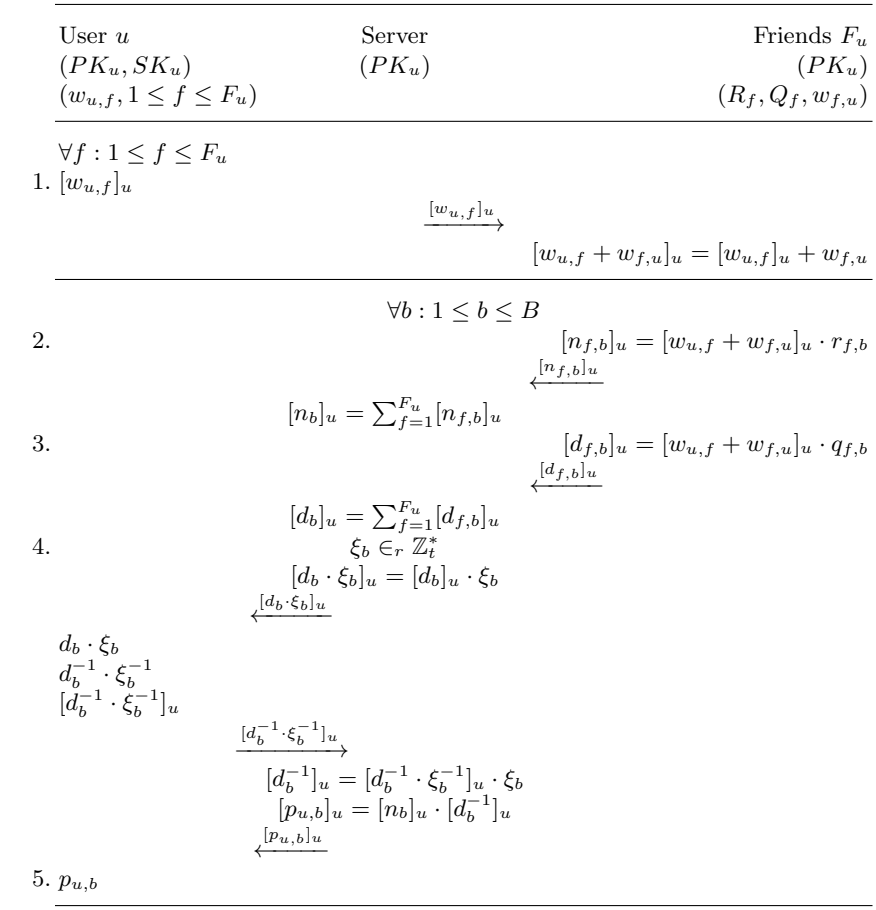

Fig. 1. Book Recommendation Protocol with Online Friends

5.1 Solution with Online Friends

Fig. 1 shows the recommendation protocol for user u with online friends. We assume that, before the protocol is run, the user u has set up his keys for the somewhat homomorphic encryption scheme, $\{PK_u, SK_u\}$, and distributed the public key. The protocol works as follows:

1. Each friend f of the user u computes their weight $w_{u,f}+w_{f,u}$. To do this, the user u encrypts $w_{u,f}$ for each friend under his own key, and sends $[w_{u,f}]_u$ to the corresponding friend f. The friends compute $[w_{u,f} + w_{f,u}]_u = [w_{u,f}]_u + w_{f,u}$.
2. Given the encrypted weight, each friend computes the impact of his ratings, $(w_{u,f} + w_{f,u}) \cdot r_{f,b}$, for each book. Recall that $r_{f,b} = 0$, when the book is unrated. The friends compute $[n_{f,b}]_u = [w_{u,f} + w_{f,u}]_u \cdot r_{f,b}$, and send

$[n_{f,b}]_u$ to the server. The server sums the values received by the friends into $[n_b]_u = \sum_{f=1}^{F_u} [n_{f,b}]_u$ for each book.

3. In similar fashion, the normalization factor d_b is computed. The friends compute $[d_{f,b}]_u = [w_{u,f} + w_{f,u}]_u \cdot q_{f,b}$, and send $[d_{f,b}]_u$ to the server. The server sums the values received by the friends into $[d_b]_u = \sum_{f=1}^{F_u} [d_{f,b}]_u$ for each book.
4. To compute the predictions $p_{u,b}$, a division has to be performed. Towards this end, the server selects random values ξ_b from the multiplicative domain of the message space $\mathbb{Z}_t^*$ and blinds d_b multiplicatively for each book, $[d_b \cdot \xi_b]_u = [d_b]_u \cdot \xi_b$. The resulting values $[d_b \cdot \xi_b]_u$ are sent to the user u. The user u decrypts to $d_b \cdot \xi_b$ and computes the inverse, $d_b^{-1} \cdot \xi_b^{-1}$, for each book. These inverses are encrypted again under the users key, $[d_b^{-1} \cdot \xi_b^{-1}]_u$, and sent to the server. The server removes the blinding by multiplying with the random values ξ_b again, $[d_b^{-1}]_u = [d_b^{-1} \cdot \xi_b^{-1}]_u \cdot \xi_b$. The server then divides n_b by d_b for each book to determine the predictions, $[p_{u,b}]_u = [n_b]_u \cdot [d_b^{-1}]_u$. The encrypted predictions are then sent to the user u.
5. The user u decrypts the received predictions and uses the precomputed division lookup table to determine the actual predictions.

5.2 Solution with Offline Friends

Usage of Secret Sharing and Proxy Re-encryption Each friend f of the user secret shares the rating vector R_f and weight $w_{f,u}$. The rating vector R_f is split into the vectors S_f and T_f following the secret sharing method. Similarly, the weight $w_{f,u}$ is split into $x_{f,u}$ and $y_{f,u}$. As the secrets will be reconstructed under encryption, we set the security parameter k of the secret sharing scheme equal to the message space t of the homomorphic encryption system. The friend stores S_f and $x_{f,u}$ on the server. The vectors T_f and Q_f as well as the value $y_{f,u}$ will be distributed to the user u using proxy re-encryption. Therefore, these values are stored under encryption at the server and the re-encryption key to the user u is computed and also stored on the server.

Protocol Fig. 2 shows the recommendation protocol for user u with offline friends. We assume that, before the protocol is run, the required secrets T_f, Q_f, $y_{f,u}, 1 \leq f \leq F_u$ have been distributed and that both the user u and the server have set up their keys for the somewhat homomorphic encryption scheme, $\{PK_u, SK_u\}$ and $\{PK_s, SK_s\}$ respectively, and exchanged public keys. The protocol works as follows:

1. Both user u and the server compute the weight, $w_{u,f} + w_{f,u}$, for each friend under one another's public key. The weight is computed by $w_{u,f} + w_{f,u} = w_{u,f} + y_{f,u} + x_{f,u}$, where u holds $w_{u,f}$ and $y_{f,u}$, and the server holds $x_{f,u}$. The user u computes $[w_{u,f} + y_{f,u}]_u$ and sends this to the server, while the server computes and sends $[x_{f,u}]_s$. This allows the user to compute $[w_{u,f} + w_{f,u}]_s$ and the server to compute $[w_{u,f} + w_{f,u}]_u$.

User u	Server
(PK_u, SK_u, PK_s)	(PK_u, PK_s, SK_s)
$(T_f, Q_f, w_{u,f}, y_{f,u}, 1 \leq f \leq F_u)$	$(S_f, x_{f,u}, 1 \leq f \leq F_u)$
	$\forall f : 1 \leq f \leq F_u$
1. $[w_{u,f} + y_{f,u}]_u$	$[x_{f,u}]_s$
	$\xrightarrow{[w_{u,f}+y_{f,u}]_u}$
	$\xleftarrow{[x_{f,u}]_s}$
$[w_{u,f} + w_{f,u}]_s = [x_{f,u}]_s + (w_{u,f} + y_{f,u})$	$[w_{u,f} + w_{f,u}]_u = [w_{u,f} + y_{f,u}]_u + x_{f,u}$
	$\forall b : 1 \leq b \leq B$
2. $[z_b]_s = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_s \cdot t_{f,b}$	$[a_b]_u = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_u \cdot s_{f,b}$
$\xi_{1,b} \in_r \mathbb{Z}_t$	
$[z_b + \xi_{1,b}]_s = [z_b]_s + \xi_{1,b}$	
$[-\xi_{1,b}]_u$	
	$\xrightarrow{[z_b+\xi_{1,b}]_s, [-\xi_{1,b}]_u}$
3. $[d_b]_s = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_s \cdot q_{f,b}$	$z_b + \xi_{1,b}$
$\xi_{2,b} \in_r \mathbb{Z}_t^*$	$[z_b]_u = [-\xi_{1,b}]_u + (z_b + \xi_{1,b})$
$[d_b \cdot \xi_{2,b}]_s = [d_b]_s \cdot \xi_{2,b}$	$[n_b]_u = [z_b]_u + [a_b]_u$
$[\xi_{2,b}]_u$	
	$\xrightarrow{[d_b \cdot \xi_{2,b}]_s, [\xi_{2,b}]_u}$
4.	$d_b \cdot \xi_{2,b}$
	$d_b^{-1} \cdot \xi_{2,b}^{-1}$
	$[d_b^{-1}]_u = [\xi_{2,b}]_u \cdot (d_b^{-1} \cdot \xi_{2,b}^{-1})$
	$[p_{u,b}]_u = [n_b]_u \cdot [d_b^{-1}]_u$
	$\xleftarrow{[p_{u,b}]_u}$
5. $p_{u,b}$	

Fig. 2. Book Recommendation Protocol with Offline Friends

2. Given the encrypted weights, both the user u and the server can compute the impact of the secret shared ratings $r_{f,b} = t_{f,b} + s_{f,b}$ for each book. The user u computes $[z_b]_s = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_s \cdot t_{f,b}$ and the server computes $[a_b]_u = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_u \cdot s_{f,b}$. Together, this sums up (ignoring encryption for a moment) to $z_b + a_b = \sum_{f=1}^{F_u} (w_{u,f} + w_{f,u}) \cdot (t_{f,b} + s_{f,b}) = \sum_{f=1}^{F_u} (w_{u,f} + w_{f,u}) \cdot r_{f,b} = n_b$. The user u selects random values $\xi_{1,b}$ from the domain of message space $\mathbb{Z}_t$ and uses them to blind $[z_b]_s$. The resulting encryptions, $[z_b + \xi_{1,b}]_s$, and the encryptions to remove the blinding, $[-\xi_{1,b}]_u$, are sent to the server. Note that the server can only remove the blinding using encryptions under the user's public key.
3. The user u computes the combined weight to normalize the prediction using $[d_b]_s = \sum_{f=1}^{F_u} [w_{u,f} + w_{f,u}]_s \cdot q_{f,b}$ for each book. These are blinded multiplicatively with random values $\xi_{2,b}$, taken from the multiplicative domain

of the message space $\mathbb{Z}_t^*$. The resulting encryptions, $[d_b \cdot \xi_{2,b}]_s$, and encryptions to remove the blinding after inversion, $[\xi_{2,b}]_u$, are sent to the server. Meanwhile, the server removes the blinding values $\xi_{1,b}$ and reconstructs $[n_b]_u = [z_b]_u + [a_b]_u$.

4. The server decrypts the received encryptions, $d_b \cdot \xi_{2,b}$, and inverts them, resulting in $d_b^{-1} \cdot \xi_{2,b}^{-1}$. Under the public key of u, the blinding values $\xi_{2,b}$ are removed, resulting in the encryptions $[d_b^{-1}]_u$. The server divides n_b by d_b under the public key of u, $[p_{u,b}]_u = [n_b]_u \cdot [d_b^{-1}]_u$, for each book. The resulting encrypted predictions $[p_{u,b}]_u$ are sent to the user u.
5. The user u decrypts the received predictions and uses the precomputed division lookup table to determine the actual predictions.

6 Analysis of the Solutions

In this section, we first look at the privacy that the two protocols offer in relation to the security model. Then we look at the complexity (computational and communicational) of the protocols. Finally, we look at the performance (runtime) of the protocols with different sized datasets.

6.1 Privacy

Recall from the security model that all parties are honest-but-curious. The user u will try to learn $r_{f,b}$ and $w_{f,u}$. Friends will try to learn $w_{u,f}$. The server will try to learn $q_{f,b}$, $r_{f,b}$, $w_{u,f}$, $w_{f,u}$, and $p_{u,b}$. Given that the parties are honest-but-curious, each party should not be able to distinguish between a protocol execution and a simulation of the protocol based only on the party's input and output. However, only the user u has an output in the protocol. As such, for the server and friends, each message they receive should be indistinguishable from random messages. For the user, messages may depend on the output $p_{u,b}$.

Online Friends In this protocol, the user's friends only see encrypted values, encrypted under the key of the user u. Given that the homomorphic encryption scheme is semantically secure [4], the encrypted values are indistinguishable from encryptions of random messages. As the friends also get no output from the protocol, the protocol can easily be simulated and the friends learn nothing from the protocol.

The server also only sees encrypted values. As the homomorphic encryption scheme is semantically secure, the encrypted values are indistinguishable from encryptions of random messages. The server receives no output from the protocol, and the protocol can easily be simulated. Thus the server learns nothing from running the protocol.

After the user encrypts and sends $w_{u,f}$, the user only receives $d_b \cdot \xi_b$ and $p_{u,b}$ for all books. As $p_{u,b}$ is the output of the prediction formula 2, the user should always learn this and does not constitute a breach of privacy. The other

value, d_b, is randomized multiplicatively over the full multiplicative domain by ξ_b, and is thus indistinguishable from a value chosen at random from the domain. Because this can also be easily simulated, the privacy of d_b is preserved. The only exception to this is when $d_b = 0$, in this case $d_b \cdot \xi_b$ is also equal to 0. This only happens when none of the users friends have given a rating for b, i.e. $q_{f,b} = 0$ for $1 \leq f \leq F_u$. This situation is deemed acceptable as $q_{f,b}$ is not required to be private. By setting $d_b^{-1} \cdot \xi_b^{-1}$ to 0, the protocol can continue without the server learning anything, resulting in the prediction $p_{u,b} = 0$.

Offline Friends In the protocol with offline friends, the privacy of the user towards his friends is not in danger, as they are not involved in the protocol. In the other direction, each friend shares some information with both the user and the server. The user receives through the proxy re-encryption T_f, Q_f, and $y_{f,u}$, and the server receives S_f and $x_{f,u}$. Except for Q_f, all these values are additive secret shares and hence indistinguishable from random values [10]. This means that these values can be used as inputs to the protocol. Given that the proxy re-encryption scheme is secure, and Q_f is not required to be private from the user u, the privacy of each friend is not breached.

During the protocol, next to encrypted values, the user only receives $p_{u,b}$. As the homomorphic encryption scheme is semantically secure, the encrypted values are indistinguishable from encryptions of random messages. These messages can thus be simulated. Furthermore, the user receives $p_{u,b}$, as intended, as output of the prediction function. Thus from the user's perspective the protocol can be completely simulated.

Next to encrypted values, which are indistinguishable from encryptions of random values, the server only receives $z_b + \xi_{1,b}$ and $d_b \cdot \xi_{2,b}$. The value of z_b is protected by additive blinding, using $\xi_{1,b}$, and thus indistinguishable from a random value and possible to simulate. For d_b, as in the protocol with online friends, multiplicative blinding, using $\xi_{2,b}$, is used. Thus d_b is indistinguishable from a random value and can be simulated. Only in the case that $d_b = 0$, will the server learn something about $q_{f,b}$, which is a violation of the privacy of the user's friends. This can be avoided by setting $[d_b \cdot \xi_{2,b}]_s$ to $[\xi_{2,b}]_s$ and $[\xi_{2,b}]_u$ to $[0]_u$ when $d_b = 0$. This is only the case when $q_{f,b} = 0$, for $1 \leq f \leq F_u$, which the user knows. The server will receive $\xi_{2,b}$ instead of 0, which is a random value, and be unable to decrypt $[0]_u$ as it is protected by the user's key. The resulting prediction $p_{u,b}$ will then still be 0.

6.2 Complexity

Table 1 shows the complexity of the computational (comp) and communicational (comm) costs of each step in the protocol with online friends. The costs are given in big-O notation and for each party. The first step shows a complexity related to the number of friends for the user u, and constant for each friend. The second and third step, where the friends contribution is calculated, shows a complexity in the order of number of books for each friend, and in the order of both the number

of books and friends for the server. These steps have the largest complexity. The fourth step shows a complexity in the order of number of books for both the user and the server. The final step shows a complexity on the order of the number of books for the user. All steps together it seems that the server has the most work to do.

Table 1. Complexity of the protocol with online friends, F_u is the number of friends and B the number of books

	User u		Server		Friend	
step	comp	comm	comp	comm	comp	comm
1.	$O(F_u)$	$O(F_u)$			$O(1)$	$O(1)$
2.			$O(BF_u)$	$O(BF_u)$	$O(B)$	$O(B)$
3.			$O(BF_u)$	$O(BF_u)$	$O(B)$	$O(B)$
4.	$O(B)$	$O(B)$	$O(B)$	$O(B)$		
5.	$O(B)$					

Table 2 shows the complexity of the protocol with offline friends. The notation is the same as the previous table. The first step shows a complexity in the order of number of friends for both the user u and the server. The second step shows a complexity related to both the number of books and number of friends for both the user and the server. This step has the greatest complexity in the protocol. The third step shows a complexity in the order of number of books and number of friends for the user, and a complexity in the order of number of books for the server. The fourth step shows a complexity in the order of the number of books. The final step shows a complexity in the order of number of books for the user.

Table 2. Complexity of the protocol with offline friends, F_u is the number of friends and B the number of books

	User u		Server		Friend	
step	comp	comm	comp	comm	comp	comm
1.	$O(F_u)$	$O(F_u)$	$O(F_u)$	$O(F_u)$		
2.	$O(BF_u)$	$O(B)$	$O(BF_u)$	$O(B)$		
3.	$O(BF_u)$	$O(B)$	$O(B)$	$O(B)$		
4.		$O(B)$	$O(B)$	$O(B)$		
5.	$O(B)$					

The complexity of the homomorphic operations on the ciphertexts depends mainly on the degree of the used polynomials n. However, n also has an impact on the ring R_q and thus on the security of the encryption scheme. As such, there exist a trade-off between the complexity (and efficiency) of the individual homomorphic operations and the security offered to the user. In the performance section, we shall come back to this trade-off.

6.3 Performance

To analyze the performance of the two protocols, an implementation of the somewhat homomorphic encryption scheme has been made in C++ based on the FLINT library. Based on this implementation a prototype program of the protocols has been constructed. The prototype is single threaded and computes the different steps for each party sequentially on the same machine. As such, network latency is not taken into account. All tests are carried out on an Intel Xeon at 3GHz, with 2GB of RAM. As input data, a synthetic dataset has been constructed, as there are no publicly available datasets that have explicit fine-grained familiarity values. Some datasets have friendship links, but only as a binary value. The synthetic dataset consists of either 50, 100, or 200 friends that have each rated 25 books. The total number of books is either 500, 1000, or 2000. Note that it is not possible for 50 friends to rate 2000 books, with only 25 ratings per friend (denoted with n/a). This gives us performance information for different numbers to observe how the solutions scale. A rating is a score between 1 and 100, and the weights between users, after division by 2, is between 1 and 50.

We set the parameters of the somewhat homomorphic encryption scheme to the following, based on the suggestions of Naehrig et al. [23]. The message space t is set to 5000011, to allow for protocol runs with a maximum of 500 friends. For n we take 4096, resulting in a q of 84 bits and a logarithm of the attacker runtime of 255 for the decoding attack [19]. Successfully running the decoding attack breaks the security of the encryption scheme, therefore Naehrig et al. [23] suggest an attacker runtime for the decoding attack of at least 128, giving an equivalent of 128 bits security, or an attack complexity of 2^{128}. Table 3 shows the runtime performance of the prototype implementation with these parameters.

Table 3. Runtime of the prototype with attacker runtime logarithm of 255

online	books		
friends	500	1000	2000
50	113s	236s	n/a
100	149s	309s	706s
200	222s	456s	988s

offline	books		
friends	500	1000	2000
50	132s	282s	n/a
100	182s	387s	1021s
200	282s	588s	1477s

As can be seen from the table, the prototype for the protocol with online friends requires just under 2 minutes for the smallest dataset and over 16 minutes for the largest dataset. As expected, the prototype for the protocol with offline friends is slower. This prototype takes a little over 2 minutes for the smallest dataset and over 24 minutes for the largest dataset. This protocol has the benefit that friends need not be online, but requires more time to protect the information of those friends. When looking at the running times for the different datasets, we see a linear trend with respect to the number of books and a sub linear trend with respect to the number of friends. When looking at the protocol complexity,

this is to be expected. Most operations have to be done per book and not per friend, but computing the impact of each friend on each book is linear in both (and the slowest step in the protocols).

We can lower the security of the somewhat homomorphic encryption scheme in order to gain a speed increase of the protocols. This lowered security implies that it is easier, but still very difficult, to break the semantic security of the encryption scheme and recover encrypted messages. Should encrypted messages be recovered, privacy is lost. Towards this end, we take for n 2048, resulting in a q of 83 bits and a logarithm of the attacker runtime of 75. Table 4 shows the runtime performance with these parameters offering lowered security, but more speed.

Table 4. Runtime of the prototype with attacker runtime logarithm of 75

online	books		
friends	500	1000	2000
50	50s	102s	n/a
100	68s	137s	287s
200	104s	209s	441s

offline	books		
friends	500	1000	2000
50	59s	120s	n/a
100	85s	170s	442s
200	134s	267s	617s

From the table we can see that these parameters result in runtimes that are more than 2 times faster than the more secure parameters. As expected, the running time relations between the different datasets remains the same. The desired level of security has a large impact on the running time of the protocols, but it does not change the basic properties of the protocols.

7 Conclusion

In this paper, we proposed an efficient privacy-enhanced familiarity-based recommender system. We proposed an adjusted recommendation formula that provides more privacy than weighted average with user supplied weights. Furthermore, two different protocols have been given, one where all friends of the user are online, and another where friends are offline. In both cases, a bi-directional friendship is assumed. The privacy of these protocols has been analyzed, and two edge cases have been found and fixed. The protocols achieve privacy in the honest-but-curious model.

We have implemented the somewhat homomorphic encryption scheme of Brakerski and Vaikuntanathan [4]. Based on this implementation, a prototype of the two protocols has been built and the efficiency of them has been analyzed. The prototype is limited to a single machine and single thread, and does not show the impact of latency. The prototype shows a runtime in the order of minutes with a linear trend with regards to scaling of the input set. This is a significant improvement over the work of Hoens et al. [14], the previous privacy-enhanced recommender systems with user supplied weights, which also assumed

honest-but-curious participants and ran in the order of hours. Furthermore, not all users need to be online at some or all stages of the protocol, which is required by most related work. When we compare our work to the work of Erkin et al. [9], which assumes honest-but-curious participants and allows for offline users, we can see the difference in slowdown of the protocol when going from online to offline. The slowdown caused by our protocol is less than 1.5 times, while the slowdown of Erkin et al. is more than 6 times.

For future work, we would like to see if the efficiency of the protocols can be improved further. Furthermore, given our implementation, we would like to see the influence of somewhat homomorphic encryption, as opposed to additive homomorphic encryption, on similar problems.

Acknowledgement This work is partially funded by the THeCS project as part of the Dutch national program COMMIT.

References

1. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, Feb. 2006.
2. A. Basu, J. Vaidya, and H. Kikuchi. Efficient privacy-preserving collaborative filtering based on the weighted slope one predictor. *Journal of Internet Services and Information Security (JISIS)*, 1(4):26–46, Nov. 2011.
3. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin / Heidelberg, 1998.
4. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Proceedings of the 31st annual conference on Advances in cryptology*, CRYPTO'11, pages 505–524, Berlin, Heidelberg, 2011. Springer-Verlag.
5. J. F. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.
6. K. M. Carley and D. Krackhardt. Cognitive inconsistencies and non-symmetric friendship. *Social Networks*, 18(1):1–27, 1996.
7. S. S. M. Chow, J. Weng, Y. Yang, and R. H. Deng. Efficient unidirectional proxy re-encryption. In *Proceedings of the Third international conference on Cryptology in Africa*, AFRICACRYPT'10, pages 316–332, Berlin, Heidelberg, 2010. Springer-Verlag.
8. N. Dokoohaki, C. Kaleli, H. Polat, and M. Matskin. Achieving optimal privacy in trust-aware social recommender systems. In *Proceedings of the Second international conference on Social informatics*, SocInfo'10, pages 62–79, Berlin, Heidelberg, 2010. Springer-Verlag.
9. Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk. Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE Transactions on Information Forensics and Security*, 7(3):1053–1066, 2012.
10. O. Goldreich. Foundations of cryptography: a primer. *Foundations and Trends in Theoretical Computer Science*, 1:1–116, Apr. 2005.

11. G. Groh and C. Ehmig. Recommendations in taste related domains: Collaborative filtering vs. social filtering. In *In Proc ACM Group07*, pages 127–136, 2007.
12. I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 53–60, New York, NY, USA, 2009. ACM.
13. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.
14. T. Hoens, M. Blanton, and N. Chawla. A private and reliable recommendation system for social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 816–825, Aug. 2010.
15. T. R. Hoens, M. Blanton, and N. V. Chawla. Reliable medical recommendation systems with patient privacy. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 173–182, New York, NY, USA, 2010. ACM.
16. A. Jeckmans, Q. Tang, and P. Hartel. Privacy-preserving collaborative filtering based on horizontally partitioned dataset. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pages 439–446, May 2012.
17. K. Lerman. Social networks and social information filtering on digg. *Computing Research Repository (CoRR)*, abs/cs/0612046:1–8, 2006.
18. B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Proceedings of the Practice and theory in public key cryptography, 11th international conference on Public key cryptography*, PKC'08, pages 360–379, Berlin, Heidelberg, 2008. Springer-Verlag.
19. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011*, CT-RSA'11, pages 319–339, Berlin, Heidelberg, 2011. Springer-Verlag.
20. A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. *Proceedings of the VLDB Endowment*, 4(7):440–450, Apr. 2011.
21. J. Masthoff and A. Gatt. In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. *User Modeling and User-Adapted Interaction*, 16:281–319, 2006.
22. 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.
23. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.
24. R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. 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, New York, NY, USA, 2009. ACM.
25. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001.
26. P. Victor, M. Cock, and C. Cornelis. Trust and recommendations. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 645–675. Springer US, 2011.