Cryptographic Framework for Analyzing the Privacy of Recommender Algorithms

Qiang Tang
DIES, Faculty of EEMCS
University of Twente
Enschede, the Netherlands
tonyrhul@gmail.com

***Abstract*—Recommender algorithms are widely used, ranging from traditional Video on Demand to a wide variety of Web 2.0 services. Unfortunately, the related privacy concerns have not received much attention. In this paper, we study the privacy concerns associated with recommender algorithms and present a cryptographic security model to formulate the privacy properties. We propose two privacy-preserving content-based recommender algorithms and prove their properties. Moreover, we show the potential weakness in some existing collaborative filtering algorithms which claim to provide privacy protection.**

***Keywords*—Recommender algorithms; Privacy; Cryptography.**

I. Introduction

Recommendation services are becoming ubiquitous in our daily life. When we subscribe a Video on Demand service, we get recommendations from our operator. When we use social networks, such as Facebook and Linkdin, we will frequently receive messages like "You know user X so you may also know user Y.". When we watch the video "You Are Not Alone" of Michael Jackson on Youtube.com, a list of other videos about Michael will be suggested to us. There are many other well-known recommendation services, such as those offered by Amazon and Netflex, and new services emerge every day.

In the core of recommendation services are recommender algorithms, which have been surveyed in-depth by Adomavicius and Tuzhilin [1]. Let $\mathcal{U}$ be the set of all users and $\mathcal{I}$ be the set of all items that can be recommended. A recommender algorithm can, for every user in the set $\mathcal{U}$, rank all the items in the set $\mathcal{I}$ so the top-X items can be recommended to the user, where X is an integer and usually much smaller than the size of $\mathcal{I}$. Typically, the recommender algorithm will be run by a server, which aims at providing the recommendation service. Roughly speaking, traditional recommender algorithms can be classified into three categories: (1) content-based, which ranks the items in the set $\mathcal{I}$ with respect to their similarities to those items that a target user has accessed before; (2) collaborative, which ranks the items in the set $\mathcal{I}$ based on the rating results of a group of users who are similar to the target user; (3) hybrid, which combines the above two.

A. Problem Statement

In most currently-deployed recommendation systems, an individual user reveals all his personal data to the recommendation server in order for the latter to compute recommendations. In fact, most of the time, the recommendation server keeps a database with all its users' personal data. This situation puts users in a vulnerable position regarding to their privacy, and it will pose serious risks of privacy loss to the users in case of unintended data leakage. In addition, it puts a burden on the recommendation server to protect users' private data in order to be compliant to privacy regulations and meet users' privacy expectations.

Perturbation and data obfuscation methods (e.g. [13], [15], [16], [18]) have been proposed to address the privacy concerns. However, with respect to such approach, it remains as a challenge to achieve both recommendation accuracy and expected privacy guarantee. In fact, there is a general issue of privacy leakage in using such approach in data mining [19]. McSherry and Mironov [11] proposed collaborative filtering algorithms in the differential privacy framework. Similar to other perturbation and anonymization based approaches, this approach still has a tradeoff between privacy and accuracy. In contrast, the other approach is to employ cryptographic techniques, such as homomorphic encryption and secure multiparty computation. A number of papers employ this approach, including those by Aïmeur et al. [2], Canny [4], [5], Polat and Du [17], Zhan et al. [20], and Erkin et al. [8]. However, so far, no rigorous security model has been proposed, and corresponding security analysis has not been done in these papers.

B. Contribution

The contribution of this paper is three-fold. The first contribution is to look at the privacy issues of recommender algorithms from a cryptographic perspective. We propose a security model and formulate the privacy properties against both a semi-trusted recommendation server and a group of malicious users. The security model provides a framework to formally prove the security of recommender algorithms. The second contribution is to propose two privacy preserving content-based recommender algorithms. In both cases, we introduce the concept of proxy user into the setting to help

978-1-4673-1382-7/12/$31.00 ©2012 IEEE

compute recommendations. Neither the proxy user nor the recommendation server will learn anything about honest users' sensitive data. The third contribution is to analyze some of the existing collaborative filtering algorithms, in particular that of Canny [4], which have been designed to provide privacy protection for users in a fully decentralized environment. We showed that there are many practical obstacles to protect users' privacy in such an environment without relying on a recommendation server.

C. Organization

The rest of the paper is organized as follows. In Section II, we describe the system model for recommendation services and propose a security model for privacy preserving recommender algorithms. In Section III, we propose two privacy preserving content-based recommender algorithms and analyze their privacy properties. In Section IV, we analyze some collaborative filtering algorithms. In Section V, we conclude the paper.

II. System and Threat Models

The following notation is used throughout the paper. $\overrightarrow{X} = (x_1, x_2, \cdots, x_n)$ denotes a vector, $\overrightarrow{X} \bigodot \overrightarrow{Y}$ denotes the inner product of $\overrightarrow{X}$ and $\overrightarrow{Y}$, $\overrightarrow{X} + \overrightarrow{Y}$ and $\overrightarrow{X} - \overrightarrow{Y}$ denote the element-wise addition and substraction of $\overrightarrow{X}$ and $\overrightarrow{Y}$ respectively.

A. System Model of Recommendation Services

A recommendation system usually adopts a centralized system structure, as depicted in Figure 1, and consist of three types of entities.

- Users, who want to receive recommendations from the recommendation server and retrieve items from the content server. Let all users be denoted by U_i $(1 \leq i \leq N)$ where N is an integer denoting the user population in the system. Let U_i's personal data be denoted as D_i from the domain $\mathcal{D}$. Note that the personal data may include profile, ratings, and recommendations received.
- A recommendation server, which will compute recommendations for all users. The recommendation server's parameter is denoted as D_S. For instance, in a content-based recommender algorithm, D_S contains the item features.
- A content server, which provides users with the items they want to retrieve.

At a high level, the workflow of a recommendation system can be divided into two stages. The first stage is recommendation retrieval: If a user wants to receive recommendations, he sends his personal data to the recommendation server. Based on the underlying recommender algorithm, the recommendation server predicts the top-X items which best fit the user's taste, and sends the item identifiers to the user. The second stage is content retrieval: After receiving the recommendations, the user retrieves the recommended items. At the same time, the user will generate ratings for these items.

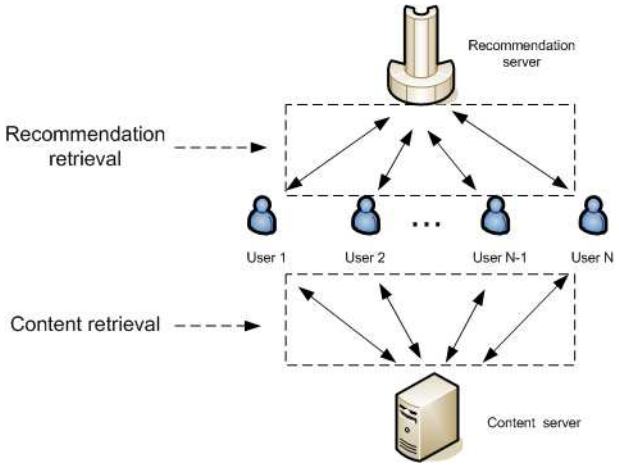

Figure 1. Centralized System Structure

In practice, a system may deviate from this system structure. One possibility is that the recommendation server and the content server can be the same entity or there is no explicit content server and the user will find the content from any available source on the Web. Another possibility is that a recommendation system may adopt a decentralized system structure, where a recommendation server is not required, and its function (the computation of recommendations) can be realized collaboratively by all (or a large portion of) the users in the system. The benefit of such a system structure is that the recommendations will not rely on a specific recommendation server. The downside of this system structure is that the accuracy of recommendations heavily rely on the participation of individual users. In this case, for the privacy of an honest user, we only need to consider a group of malicious users and can use exactly the same formulation defined below. We leave further investigation of these possibilities as a future work.

B. The Security Model

According to the system structure described in Figure 1, we assume that there is an security channel between every user and the recommendation server, where the channel provides both confidentiality and integrity. With respect to an honest user's privacy, the threats come from two types of adversaries: one is a semi-trusted recommendation server, and the other is a group of malicious users.

- We respect to a semi-trusted recommendation server, we assume that it will not deviate from the algorithm specification in order to learn a user's personal data. In particular, a semi-trusted recommendation server will not collude with a group of malicious users. Moreover, we assume it will not forge dummy users and related data, and then compute recommendations for an honest user

based on the forged data. We elaborate on this assumption in Section V.

- With respect to a group of malicious users, we assume they may collude. Note that there is a concern that malicious users may use arbitrary data as their input to the recommender algorithm. Nevertheless, we do not address this concern in our security model.

In the ideal situation, the recommendation server should be able to compute recommendations in an oblivious manner, i.e. learning nothing about any user's personal data. To formulate the security, we use the indistinguishability approach used in defining the security of encryption schemes [10].

Definition 1: A recommender algorithm achieves user privacy against a semi-trusted recommendation server if, for any $1 \leq i \leq N$, any polynomial-time adversary's advantage in the attack game (depicted in Figure 2) is negligible, where the advantage is denoted as $|\Pr[b' = b] - \frac{1}{2}|$.

1) The adversary, which simulates the recommendation server, sets up the parameter D_S and generates $D_{i_0}, D_{i_1} \in \mathcal{D}$ and sends them to U_i.
2) The challenger, which simulates U_i and other users, selects $b \in_R \{0,1\}$ and generates data D_j $(1 \leq j \leq N, j \neq i)$. The challenger then faithfully runs the recommender algorithm with the adversary on the input D_{i_b} and D_j $(1 \leq j \leq N, j \neq i)$.
3) The adversary outputs a guess bit b'.

Figure 2. Attack Game against the Recommendation Server

If the adversary's advantage is negligible then it cannot tell whether D_{i_0} or D_{i_1} has been used by U_i in the algorithm execution. Intuitively, this means that the recommendation server cannot distinguish U_i's input from any possible data from $\mathcal{D}$, so that the recommendation server learns nothing about U_i's input.

Against a group of malicious users, the ideal privacy requirement is that the algorithm does not leak more information than what can be inferred from the recommendation results that the malicious users receive. Note that the malicious users can deviate from the algorithm specification. For a formal definition, we use the simulation approach used in secure multiparty computation [3].

Definition 2: A recommender algorithm achieves user privacy against a group of t malicious users if, there exists a simulator $\mathcal{S}$, any polynomial-time adversary's advantage in the attack game (depicted in Figure 3) is negligible, where the advantage is denoted as $|\Pr[b' = b] - \frac{1}{2}|$.

Intuitively, if $b = 1$ in the game then the adversary learns nothing more than R_{j_k} $(1 \leq k \leq t)$ because this is the only information the simulator has access to. As a result, if the adversary cannot distinguish whether it has interacted with the challenger or the simulator, then the challenger leaks no more information than R_{j_k} $(1 \leq k \leq t)$.

1) The adversary, which simulates a group of t malicious users U_{j_k} $(1 \leq k \leq t, j_k \in \mathcal{C})$, sets up colluded users' data D_{j_k} $(1 \leq k \leq t)$. Note that $\mathcal{C}$ is a set containing the indexes of colluded users.
2) The challenger, which faithfully simulates the recommendation server and un-colluded users, sets up the un-colluded users' data D_i $(1 \leq i \leq N, i \notin \mathcal{C})$ and the recommendation server's parameter D_S. Let R_{j_k} $(1 \leq k \leq t)$ be the recommendations received by the colluded users when they behave honestly. The challenger selects $b \in_R \{0,1\}$ and follows step (a) if $b = 0$ and step (b) otherwise.
   a) In this case, the challenger runs the recommender algorithm with the adversary, which can behave maliciously.
   b) In this case, the simulator $\mathcal{S}$ takes R_{j_k} $(1 \leq k \leq t)$ as input and runs the recommender algorithm with the adversary, which can behave maliciously.
3) The adversary outputs a guess bit b'.

Figure 3. Attack Game against t Malicious Users

III. Secure Content-based Recommender Algorithms

Following the discussion in [1], we assume that a content-based recommender algorithm works in the following way. Let K_x $(1 \leq x \leq L)$ be a set of keywords used to define the attribute vectors for users and data items.

- For the user U_i, where $1 \leq i \leq N$, his personal data is an attribute vector $\overrightarrow{V_i} = (V_{i1}, V_{i2}, \cdots, V_{iL})$, where an element represents the user's preferences to the corresponding keyword.
- The recommendation server's parameter D_S is an item database $\mathcal{I}$ of size M, where the j-th element contains an item identifier d_j and an attribute vector $\overrightarrow{I_j} = (I_{j1}, I_{j2}, \cdots, I_{jL})$, where an element represents the item's relevance to the corresponding keyword.

As mentioned in [1], the above attribute vectors can be computed using the TF-IDF method. In order to generate recommendations for U_i, the recommendation server first computes the similarities between $\overrightarrow{V_i}$ and $\overrightarrow{I_j}$ $(1 \leq j \leq M)$. Here, we consider cosine similarities, defined below.

$$S_{ij} = \frac{\sum_{t=1}^{L} V_{it} \cdot I_{jt}}{\sqrt{\sum_{t=1}^{L} V_{it}^2} \cdot \sqrt{\sum_{t=1}^{L} I_{jt}^2}} \quad (1)$$

$$= \frac{\overrightarrow{V_i}}{\sqrt{\sum_{t=1}^{L} V_{it}^2}} \bigodot \frac{\overrightarrow{I_j}}{\sqrt{\sum_{t=1}^{L} I_{jt}^2}} \quad (2)$$

The recommendation server then can rank S_{ij} $(1 \leq j \leq M)$ and send the identifiers of top-X items to U_i. Without any privacy protection, to compute the recommendations for U_i, the recommendations server will directly access $\overrightarrow{V_i}$. Clearly, as shown by Equation (2), the cosine similarity between $\overrightarrow{V_i}$ and $\overrightarrow{I_j}$ is the inner product of normalized versions of both

vectors. In addition, the normalized vector elements and the ultimate similarities will be float numbers, which becomes an obstacle to apply standard cryptographic techniques. However, note that if we scale the normalized vectors using the same parameter (say, multiple every element with 100) and make their elements be integers, then the computed recommendations will stay the same.

To facilitate our discussion, we still use the notation $\overrightarrow{V_i}$ and $\overrightarrow{I_j}$ $(1 \leq j \leq M)$ to denote the firstly normalized and then scaled vectors, where the elements are integers from the domain $[0, 2^A - 1]$. Moreover, we still use S_{ij} $(1 \leq j \leq M)$ to denote the similarities, which are integers now.

A. First Privacy Preserving Algorithm

For a content-based recommender algorithm, in order to achieve the privacy properties defined in Section II, the recommendation server and the honest user should collaboratively compute the inner products and rank the results. To our knowledge, to achieve this goal, the computation and communication complexities will be linear for both the user and the recommendation server in the size of item database. For majority ordinary users in a recommendation system, the complexities may be unaffordable.

We introduce the concept of proxy user into the setting. As security is concerned, we assume the proxy user to be malicious in order to obtain the honest user's personal data. However, we do not consider the case that the proxy user sends manipulated data to the honest user to mount a DoS (denial of service) attack. In practice, a proxy user is such an entity that has adequate computation and communication capacities and can help other ordinary users to compute recommendations. Hence, a proxy user can be the powerful ones among the users in a recommendation system. As a result, the computation of recommendations will be a three-party protocol, running among U_i, the proxy user and the recommendation server. By doing so, the computation and communication workload can be shifted from the user U_i to the proxy user. The resulted algorithm is supposed to be much more efficient than a secure solution without using the concept of proxy user.

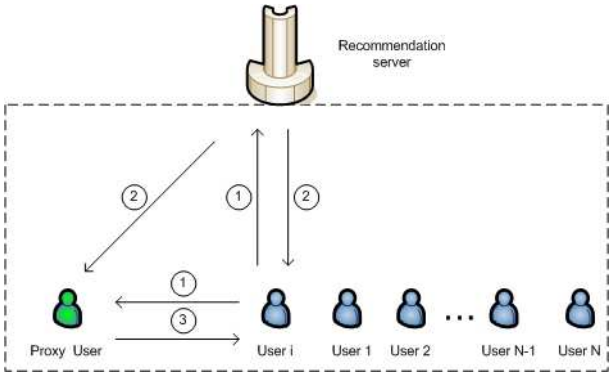

Figure 4. Illustration of the First Algorithm

The first privacy preserving algorithm is shown in Figure 4, with the details below. The step numbers correspond to those in Figure 4. Note that all communications are carried out through a secure channel.

1) U_i generates a vector $\overrightarrow{R_i}$, whose elements are randomly chosen from $[0, 2^B - 1]$ where B is an integer acting as the security parameter, and computes $\overrightarrow{S_i} = \overrightarrow{V_i} + \overrightarrow{R_i}$. Then it sends $\overrightarrow{S_i}$ to the recommendation server and sends $\overrightarrow{R_i}$ to the proxy user.
2) The recommendation server sends $\tilde{S_{ij}}$ $(1 \leq j \leq M)$ and the item identifiers to U_i, where $\tilde{S_{ij}} = \overrightarrow{S_i} \bigodot \overrightarrow{I_j}$. In addition, it sends $\overrightarrow{I_j}$ $(1 \leq j \leq M)$ to the proxy user.
3) The proxy user sends $\tilde{\tilde{S_{ij}}}$ $(1 \leq j \leq M)$ to U_i, where $\tilde{\tilde{S_{ij}}} = \overrightarrow{R_i} \bigodot \overrightarrow{I_j}$.
4) After receiving $\tilde{S_{ij}}$ and $\tilde{\tilde{S_{ij}}}$ for $1 \leq j \leq M$, U_i obtains the similarities $S_{ij} = \tilde{S_{ij}} - \tilde{\tilde{S_{ij}}}$ $(1 \leq j \leq M)$. Then, it can easily sort the similarities and obtain the identifiers of top-X items.

With this algorithm, U_i's communication complexity is $2(M \cdot (\log_2 L + 2(B+1)))$ bits, and his computation complexity is $M+1$ addition and $M \log M$ integer comparisons if a binary sorting algorithm is used. The recommendation server's communication complexity is $M \cdot (\log_2 L + 2(B+1)) + M \cdot L \cdot (B+1)$ bits, and its computation complexity is $M \cdot (L-1)$ addition and $M \cdot L$ multiplications. The proxy user's communication complexity is $M \cdot (\log_2 L + 2(B+1)) + M \cdot L \cdot (B+1)$ bits, and its computation complexity is $M \cdot (L-1)$ addition and $M \cdot L$ multiplications. Consider a naive solution, where the recommendation server sends U_i the item database. In this case, U_i's communication complexity is $M \cdot L \cdot (B+1)$ bits, and its computation complexity is $M \cdot (L-1)$ addition, $M \log M$ integer comparisons, and $M \cdot L$ multiplications. Note that, with the increase of M, for U_i, the efficiency of the proposed algorithm becomes more significant for U_i.

Next, we consider the security of the proposed algorithm in the security model proposed in Section II. The proxy user learns no information about $\overrightarrow{V_i}$, because it only receives a random vector $\overrightarrow{R_i}$. Certainly other users will learn nothing as well since they are not involved at all. Therefore, the following theorem holds.

Theorem 1: The proposed algorithm achieves user privacy against all $(N+1)$ users with respect to Definition 2 unconditionally.

The recommendation server receives $\overrightarrow{S_i}$, which is a randomized version of $\overrightarrow{V_i}$. Consider another vector $\overrightarrow{S_i^*} = \overrightarrow{V_i^*} + \overrightarrow{R_i^*}$, where the elements of $\overrightarrow{V_i^*}$ are from the domain $[0, 2^A - 1]$ and the elements of $\overrightarrow{R_i^*}$ are randomly chosen from the domain $[0, 2^B - 1]$. The statistical distance between $\overrightarrow{S_i}$ and $\overrightarrow{S_i^*}$ is at most $\frac{L \cdot 2^A}{2^A + 2^B}$. Therefore, the following theorem holds.

Theorem 2: The proposed recommender algorithm achieves user privacy against a malicious server with respect to Defi-

nition 1, if $\frac{L \cdot 2^A}{2^A + 2^B}$ is negligible.

Note the fact that, for any existing databases, the value of $L \cdot 2^A$ will be smaller than 2^{128}. Therefore, to achieve 128 bits security, the value of B can be set to be 256.

B. Second Privacy Preserving Algorithm

In the previous algorithm, the recommendation server needs to send the item database $\mathcal{I}$ to the proxy user. In practice, the recommendation server may not want to do so because $\mathcal{I}$ is its own asset and may be regarded as private. We propose a new recommender algorithm to mitigate the issue by only revealing the permuted item database to the proxy user.

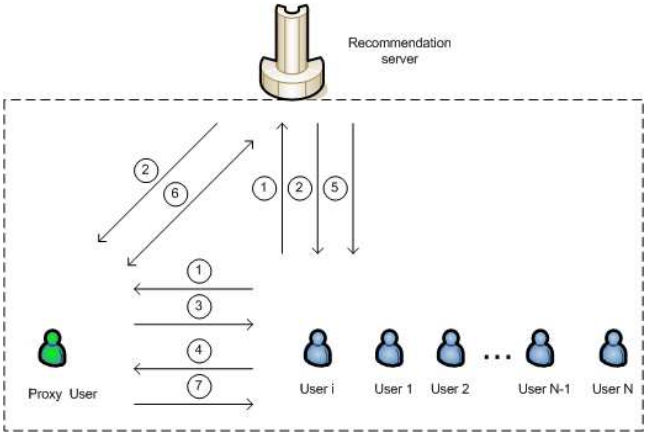

Figure 5. Illustration of the Second Algorithm

The proposed algorithm is shown in Figure 5, with the details below. The step numbers correspond to those in Figure 5. Note that all communications are carried out through a secure channel.

1) U_i generates a vector $\overrightarrow{R_i}$, whose elements are randomly chosen from $[0, 2^B - 1]$ where B is an integer acting as the security parameter, and computes $\overrightarrow{S_i} = \overrightarrow{V_i} + \overrightarrow{R_i}$. Then, it sends $\overrightarrow{S_i}$ to the recommendation server, and sends $\overrightarrow{R_i}$ to the proxy user.
2) The recommendation server generates a random permutation Φ, which is used to permute the locations of the elements in a vector of length M and generate a new vector[a]. Then, it computes $\Phi((\tilde{S_{i1}}, \tilde{S_{i2}}, \cdots, \tilde{S_{iM}}))$, where, for $1 \leq j \leq M$, $\tilde{S_{ij}} = \overrightarrow{S_i} \bigodot \overrightarrow{I_j}$, and sends $\Phi((\overrightarrow{I_1}, \overrightarrow{I_2}, \cdots, \overrightarrow{I_M}))$ to the proxy user and sends $\Phi((\tilde{S_{i1}}, \tilde{S_{i2}}, \cdots, \tilde{S_{iM}}))$ to U_i.
3) The proxy user sends $\Phi((\tilde{S_{i1}}, \tilde{S_{i2}}, \cdots, \tilde{S_{iM}}))$ to U_i, where $\tilde{\tilde{S}}_{ij} = \overrightarrow{R_i} \bigodot \overrightarrow{I_j}$.
4) U_i firstly sorts the elements of the vector $\Phi((\tilde{S_{i1}}, \tilde{S_{i2}}, \cdots, \tilde{S_{iM}})) - \Phi((\tilde{\tilde{S}}_{i1}, \tilde{\tilde{S}}_{i1}, \cdots, \tilde{\tilde{S}}_{iM}))$, which is a permuted version of the similarity vector. As a result, it obtains a vector $\overrightarrow{O} = (O_1, O_2, \cdots, O_X)$, where, for $1 \leq t \leq X$, O_t is an integer indicating that the O_t-th element in the permuted vector ranks in the top-X. Then, it sends $\overrightarrow{O}$ to the proxy user.
5) The recommendation server generates an encrypted identifier database DB of the form $DB = (\mathsf{Enc}(d_{f_1}, \mathcal{SK}), \mathsf{Enc}(d_{f_2}, \mathcal{SK}), \cdots, \mathsf{Enc}(d_{f_M}, \mathcal{SK}))$, where Enc is a symmetric encryption scheme such as AES, $\mathcal{SK}$ is a symmetric key, and f_i for $1 \leq i \leq M$ is determined by the permutation Φ in the following way: the permutation Φ turns the f_i-th element of a vector into the i-th element in a new vector. The recommendation server sends $\mathcal{SK}$ to the user U_i. Note that d_{f_i} $(1 \leq i \leq M)$ are item identifiers of the item database $\mathcal{I}$.
6) For every $1 \leq t \leq X$, the proxy user runs a PIR (Private Information Retrieval) protocol with the recommendation server to retrieve the O_t-th element from the database DB, namely $\mathsf{Enc}(d_{f_{O_t}}, \mathcal{SK})$, from DB. Many PIR protocols are surveyed available [9], and such protocols can be efficient as shown in [12].
7) The proxy user sends $\mathsf{Enc}(d_{f_{O_t}}, \mathcal{SK})$ $(1 \leq t \leq X)$ to U_i.
8) U_i decrypts $\mathsf{Enc}(d_{f_{O_t}}, \mathcal{SK})$ $(1 \leq t \leq X)$ and obtains the item identifiers of the top-X rated items.

With this algorithm, U_i's communication complexity is $2(M \cdot (\log_2 L + 2(B+1)))$ bits, considering M is much larger K, and its computation complexity is $M + 1$ addition, $M \log M$ integer comparisons, and M symmetric decryptions. The recommendation server's communication complexity is $M \cdot (\log_2 L + 2(B+1)) + M \cdot L \cdot (B+1)$ bits and communication caused by the PIR protocol, and its computation complexity is $M \cdot (L-1)$ addition, $M \cdot L$ multiplications, M symmetric encryptions, and the computation caused by the PIR protocol. The proxy user's communication complexity is $M \cdot (\log_2 L + 2(B+1)) + M \cdot L \cdot (B+1)$ bits and communication caused by the PIR protocol, and its computation complexity is $M \cdot (L-1)$ addition, $M \cdot L$ multiplications, and the computation caused by the PIR protocol. Regarding the communication and computation complexities for the involved parties, it is clear that this algorithm is less efficient than the algorithm proposed in Section III-A. The additional complexities are caused by the executions of the PIR protocol in step 6 of the algorithm. With PIR, the recommendation server will know the top-X rated items for U_i.

It is straightforward to see that, compared with the algorithm from Section III-A, we have added some extra steps in this algorithm to protect the privacy of the recommendation server. In contrast to the previous algorithm, here, the proxy user only obtains a randomly permuted version of $\mathcal{I}$. Even if the proxy user and U_i collude, they only obtain a randomly permuted version of $\mathcal{I}$ and those top-X item features. Based on the same arguments in III-A and the security properties of employed encryption scheme and PIR protocol, it is straightforward to verify that the Theorem 1 and 2, which hold for the

[a]For instance, let $M = 4$ and a permutation be $\Phi = (124)(3)$. Then, for any vector (a_1, a_2, a_3, a_4), after applying Φ the resulted vector is (a_2, a_4, a_3, a_1).

algorithm in Section III-A, still hold for the algorithm from this subsection.

IV. Collaborative Filtering Algorithms

In this section, we briefly comment on the algorithm by Erkin et al. [8], and then focus on the algorithm designed by Canny [4]. The algorithm by Erkin et al. [8] is rather straightforward in the sense that, in order to compute recommendations for a user, everything needs to be encrypted by this user's public key then this user and the recommendation server need to perform some two-party computation. This may cause efficiency problem if a lot of users want to compute recommendations at the same time. In contrast, with Canny's algorithm, users can compute their recommendations in a single protocol execution. There is no rigorous security analysis in either [8] or [4].

A. Canny's Recommender Algorithm

Suppose that the users in the system are denoted as U_i $(1 \leq i \leq N)$ and there are M items in the system. The user U_i has a preference attribute vector $\overrightarrow{V_i} = (V_{i1}, V_{i2}, \cdots, V_{iM})$, where V_{ij} denotes U_i's rating on the j-th item.

It is assumed that every user is also a tallier, which is the least structured case as claimed in [4]. Therefore, there are N talliers in the system. It is assumed that a fraction $\alpha > 50\%$ users (or talliers) are trustworthy, which means they will follow the protocol specification. Furthermore, it is assumed that there is a WORM (Write-Once, Read-Many) device, called blackboard, which will be used by users to communicate with each other. The system structure is shown in Figure 6. Finally, it is assumed that there is a common source of randomness, i.e. users can retrieve randomness using a global coin toss.

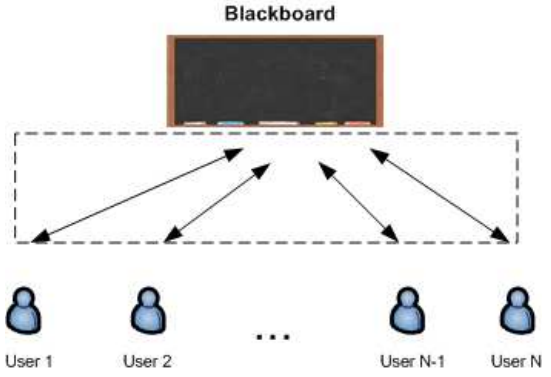

Figure 6. Canny's System Structure

Canny's algorithm has a global setup phase, in which, using the (t, N) threshold scheme [14], every user U_i obtains a private key share $\mathcal{SK}_i$ of a public/private key pair $(\mathcal{PK}, \mathcal{SK})$ of the ElGamal public key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ [7]. Note that $\frac{t}{N} > 50\%$ should hold. Canny's algorithm makes use of a conjugate gradient algorithm. Let $\mathbb{A}$ be a matrix with K rows and M columns, where the matrix elements are randomly-chosen integers. With respect to Canny's recommender algorithm, the following steps are required to be iterated about 40-60 times.

1) The user U_i, for any $1 \leq i \leq N$, computes his contribution to the gradient vector, which is $\mathbb{A}\overrightarrow{V_i}^T\overrightarrow{V_i}$. For the convenience, the contribution is written in the form $\overrightarrow{G_i} = (G_{i1}, G_{i2}, \cdots, G_{iY})$, where $Y = K \cdot M$. U_i then writes $\overrightarrow{C_i}$, $\overrightarrow{D_i}$, and E_i to the blackboard. These values are defined as follows.
   - $\overrightarrow{C_i} = (C_{i1}, C_{i2}, \cdots, C_{iY})$, which is an encrypted version of $\overrightarrow{G_i}$, i.e. $C_{ij} = \mathsf{Enc}(G_{ij}, \mathcal{PK})$ for $1 \leq j \leq Y$.
   - $\overrightarrow{D_i} = (D_{i1}, D_{i2}, \cdots, D_{iY})$, which is an encrypted version of the square of $\overrightarrow{G_i}$, i.e. $C_{ij} = \mathsf{Enc}(G_{ij}^2, \mathcal{PK})$ for $1 \leq j \leq Y$.
   - E_i contains the following proofs: for $1 \leq j \leq Y$, the value encrypted in D_{ij} is a square of the value encrypted in C_{ij}, and the sum of encrypted values in $\overrightarrow{D_i}$, namely $\sum_{j=1}^{Y} G_{ij}^2$, is less than a bound B.
2) Based on a global coin toss, U_i, for any $1 \leq i \leq N$, chooses a subset of $\log N$ users, denoted as U_{i_k} $(1 \leq k \leq \log N)$. For $1 \leq k \leq \log N$, U_i does the following.
   - Retrieve $\{\overrightarrow{C_{i_k}}, \overrightarrow{D_{i_k}}, E_{i_k}\}$ from the blackboard.
   - Verify the zero knowledge proofs in E_{i_k} are valid. If the verification passes, write "OK" on the backboard for U_{i_k}, otherwise write "not-OK".

   At the end of this step, every user will have $\log N$ votes, which are either "OK" or "not-OK".
3) Based on a global coin toss, U_i, for any $1 \leq i \leq N$, chooses a subset of $\log N$ items to aggregate. Let the subset with indexes from the set $\{j_{i_1}, j_{i_2}, \cdots j_{i_{\log N}}\}$. For $1 \leq k \leq \log N$, U_i computes $A_{j_{i_k}} = \prod_{h \in \mathcal{H}} C_{h_{j_{i_k}}}$, where $\mathcal{H}$ is the set of indexes for users with "OK". U_i writes $A_{j_{i_k}}$ $(1 \leq k \leq \log N)$ on the blackboard.
4) For $1 \leq j \leq Y$, U_i, for any $1 \leq i \leq N$, checks that the majority of the values computed for the j-th item are equal, denoted as A_j. If so, U_i then writes the partially decrypted values $\mathsf{Dec}(A_j, \mathcal{SK}_i)$ on the blackboard.
5) Recall from Step 3 that the user U_i, for any $1 \leq i \leq N$ is responsible for the data items with indexes from the set $\{j_{i_1}, j_{i_2}, \cdots j_{i_{\log N}}\}$. For $1 \leq k \leq \log N$, the user U_i then multiply all the partially decrypted values for the j_{i_k}-th item, and compute the value of $\sum_{h \in \mathcal{H}} G_{h j_{i_k}}$ where $\mathcal{H}$ is the set of users with "OK".
6) Users perform other operations following the conjugate gradient algorithm [4].

B. Observations on Canny's Algorithm

Canny's algorithm assumes a fully decentralized setting, in an attempt to avoid relying on a semi-trusted recommendation server. With respect to the security model described in Section II, the extensive usage of cryptographic techniques seems not

sufficient to make Canny's algorithm secure in practice. We have three major observations.

The first issue with this algorithm is the user membership management in a fully decentralized environment. In particular, in the initialization, the users should collaboratively setup their parameters. Thinking about a typical application of collaborative recommender algorithms such as Video on Demand service, it may contain thousands of users. Without any coordination, it seems to be a challenge to run the initialization. Additionally, users may dynamically join or leave the service, in a decentralized setting, it remains as a challenge to deal with the membership management because it requires add/delete secret shares of the employed threshold cryptosystem.

The second issue is that that the blackboard is purely a storage facility and it does not get involved in any other activities, hence, the users are assumed to be self-organized. Due to this nature, it may suffer from Sybil attacks [6], where malicious users will try to register a large number of new users in the system until they gain control over the whole system. As a consequence of a Sybil attack, the assumption that "A fraction $\alpha > 50\%$ users (or talliers) are trustworthy" will not hold anymore, and malicious users may recover the personal data of the honest users. In this case, the security notion of Definition 2 cannot be achieved. Without a commonly (semi-)trusted party such as the recommendation server in the centralized structure, it is unclear how to defend Sybil attacks.

The third issue is that the blackboard should be at least semi-trusted by all users, so that it fails to avoid relying on a semi-trusted party (if the blackboard is required to be semi-trusted, then it is equivalent to a semi-trusted recommendation server) as hoped by employing a decentralized approach. It and the users will mutually authenticate each other and establish a secure channel (with integrity protection) for all the communications. If the blackboard does not authenticate the users, then malicious users can impersonate honest users to upload forged data and influence the recommendation results. This will make the zero knowledge proofs in Step 1 of the algorithm ineffective. If the users do not authenticate the blackboard, then the same problem as described in the previous case remains because the malicious users can try to impersonate the blackboard. Moreover, the blackboard should not supply forged data to the users, then it can obtain the users' personal data, namely $\overrightarrow{V_i}$ $(1 \leq i \leq N)$. Additionally, in a fully decentralized environment, it is unclear how to choose a party to act as the blackboard.

V. Conclusion

In this paper, we presented the first cryptographic security model for analyzing the security of recommender algorithms. We proposed two privacy preserving content-based algorithms, and proved their securities in our security model. Both algorithms use lightweight cryptographic techniques such as random permutation and symmetric key encryption for the users involved, and the resulted communication and computation complexities are low enough for the algorithms to be efficient in practice. Moreover, we analyzed Canny's algorithm [4] and showed that there are many obstacles to protect users' privacy in a fully decentralized environment without relying on a recommendation server. This paper leaves us many lines of future work.

- One is to design a secure collaborative filtering algorithm which can be proved secure in our security model. With such an algorithm, there are other issues worth investigating. For example, how to address user membership management issue? We have shown that this issue may make Canny's algorithm insecure and unrealistic in practice. Another issue is that, if threshold cryptographic techniques are used, how to dynamically adjust the threshold? This is important because, for a recommender algorithm, user membership may be dynamic in nature.
- Another is that, in the security model, we explicitly make the assumption that a semi-trusted recommendation server will not forge dummy users and related data, and then compute recommendations for an honest user based on the forged data. For a content-based recommender algorithm, this assumption does not make a difference because recommendations are generated based on a honest user's own personal data. However, for a collaborative filtering algorithm, if this assumption is not true then the recommendation server can trivially figure out an honest user's personal data by computing recommendation for the dummy users based on the honest user's data. How to deal with this is an open issue for future research. It seems that we may need a decentralized setting to throttle a malicious recommendation server. However, in this direction, we need to address a lot of problems such as those we have pointed out for Canny's algorithm.
- Finally, in our security model, we have focused on protecting the privacy of users. In Section III, we have indicated that there may be privacy concerns for the recommendation server as well. It remains as a future work to extend the security model to cover such privacy requirements. The algorithm in Section III-B may still need to be improved.

Acknowledgement

The research for this work was carried out within the Kindred Spirits project, part of the STW Sentinels research program. The author would like to thank the anonymous reviewers and Michael Beye (from Delft University of Technology) for their comments.

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] R. Canetti. Security and composition of cryptographic protocols. *Journal of Cryptology*, 13:143–202, 1999.

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

[5] 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.

[6] J. R. Douceur. A threshold cryptosystem without a trusted party. In *International workshop on Peer-To-Peer Systems*, volume 2429 of *LNCS*, pages 251–260. Springer, 2002.

[7] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1985.

[8] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk. Efficiently computing private recommendations. In *International Conference on Acoustic, Speech and Signal Processing*, 2011.

[9] W. Gasarch. A survey on private information retrieval. http://www.cs.umd.edu/ gasarch/pir/pir.html.

[10] S. Goldwasser and S. micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[11] 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.

[12] Femi Olumofin and Ian Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. In *Financial Cryptography and Data Security*, 2011.

[13] R. Parameswaran. *A robust data obfuscation approach for privacy preserving collaborative filtering*. PhD thesis, Georgia Institute of Technology, 2006.

[14] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, pages 522–526, 1991.

[15] 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.

[16] H. Polat and W. Du. Privacy-preserving collaborative filtering. *International journal of electronic commerce*, 9:9–36, 2005.

[17] H. Polat and W. Du. Privacy-preserving top-N recommendation on distributed data. *J. Am. Soc. Inf. Sci. Technol.*, 59:1093–1108, 2008.

[18] 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.

[19] J. Vaidya and C. Clifton. Privacy-preserving data mining: Why, How, and When. *IEEE Security and Privacy*, 2:19–27, 2004.

[20] 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.