A Privacy Review of Vertically Partitioned Data-based Privacy-Preserving Collaborative Filtering Schemes

Murat Okkalioglu*, Mehmet Koc**, Huseyin Polat***[‡]

* Computer Engineering Department, Faculty of Engineering, Yalova University, 77200 Yalova

**Electrical and Electronics Engineering, Faculty of Engineering, Bilecik Seyh Edebali University, 11210 Bilecik

*** Computer Engineering Department, Faculty of Engineering, Anadolu University, 26470 Eskisehir

[‡] Corresponding Author; Address: Anadolu University, Computer Engineering Department, 26470, Eskisehir, Turkey Tel: +90 222 321 3550, Fax: +90 222 323 9501, e-mail: polath@anadolu.edu.tr

Abstract- E-commerce companies utilize collaborative filtering approaches to provide recommendations in order to attract customers. Consumer participation through supplying feedbacks is an important component for a recommendation system to produce accurate predictions. New companies in the marketplace might lack enough data for collaborative filtering purposes. Thus, they can come together to share their vertically partitioned data for better services. Although partitioned data-based recommendation schemes provide accurate predictions, privacy issues might pose different risks to the companies participating into such collaboration. Partitioned data-based privacy-preserving collaborative filtering schemes aim to provide accurate predictions without neglecting the privacy of such data holders. However, the collaborating parties' privacy, provided by these schemes, might not be protected as much as believed. In this study, the privacy, offered by vertically partitioned binary ratings-based privacy-preserving collaborative filtering schemes, is examined by three different attacks and experimentally tested. Empirical outcomes show that the collaborating parties are still able to derive each other's confidential data.

Keywords- Privacy; Collaborative filtering; Binary ratings; Vertically partitioned data; Attack scenarios.

1. Introduction

The Internet age has been offering great opportunities for companies to reach out their potential customers around the world. Anyone sitting in front of a computer can visit any site to browse, review, or buy an item. Thus, customers will be equipped with a large amount of data before making a decision about an item. *Information overload* refers to the fact that the amount of data human beings can process has some limits and this limit makes the decision making process difficult [1]. Information overload is an important problem for e-commerce companies hindering their customers from spotting right products.

E-commerce companies might collect implicit (browsing, purchase history, time spent, etc.) and explicit information (ratings, reviews, etc.) about their customers [2, 3]. They might provide referrals to their customers to overcome the information overload problem by utilizing the data collected from them. Collaborative filtering (CF) is a technique to offer such recommendations based on user data. CF was first coined with Tapestry project [4]. A typical CF system is composed of an $n \times m$ matrix with n users have a rating vector of m items. This matrix is usually sparse. The users can express their ratings in different scales such as numeric (5-star, 1 to 10, etc.) or binary (*like* or *dislike*). CF systems utilize the ratings to offer right products to their customers.

Data sparsity is a crucial problem in CF [5, 6]. The more ratings an e-commerce company has, the better recommendations will be produced. Dense matrix for a CF system will allow an e-company to mine more reliable relationships among users or items. User participation is therefore important to obtain accurate recommendations. However, users may be unwilling to participate in providing their true preferences due to privacy risks like unsolicited marketing, price discrimination, unauthorized access, government surveillance, and selling personal information in case of bankruptcy [7, 8].

Phelps et al. [9] report that the majority of users are concerned about how companies use their data and want control over their data. Users believe that the companies are not concerned about privacy and know too much about their users. True user participation is important in CF. Therefore, privacy-preserving collaborative filtering (PPCF) schemes are proposed to protect privacy. They aim to provide accurate referrals to users without neglecting privacy. PPCF schemes must achieve privacy, accuracy, and performance [6]. In a typical scenario, users mask their data before sending them to a central server for CF purposes. The server has an access to the perturbed data, which is different from the original one so that it is unable to retrieve individuals' private information.

Although PPCF schemes promise privacy, there are some studies arguing that privacy is not protected as much as believed [10, 11, 12, 13, 14]. A data disguising method, random perturbation, is studied in [10, 11]. It is argued that data perturbed by random perturbation techniques, which basically add random noise to the original data, have predictable nature. Thus, the data perturbed by this method can be extracted using spectral filtering (SF) [10, 11]. On the other hand, CF-based systems are examined by different scholars [12, 13, 14]. Zhang et al. [12] propose two different techniques to disclose the ratings of the users perturbed by the PPCF method proposed in [15]. In [13], live CF systems are attacked by utilizing auxiliary information and tracking the temporal changes of the public output on the targeted CF services. The authors in [14] analyze a specific PPCF scheme [20], which disguises binary data by randomized response technique (RRT) [17] and discloses which items are rated.

The related studies up to now focus on central server-based PPCF systems to obtain private information. Inspiring from these studies, this paper conducts a privacy review of vertically partitioned data-based PPCF schemes on binary ratings. Privacy has two aspects in PPCF. The first is to disguise the actual rating values; the other is to disguise the rated items. Our aim is to derive the collaborating parties' private data considering two aspects of privacy. Three different attack scenarios are devised, *acting as an active user in multiple scenarios*, *knn-based*, and *perfect match* attacks, to accomplish data reconstruction. The first attack monitors similarity scores between repeated queries differing by one cell only. Therefore, altered rating cell in each query could be reconstructed. *knn-based* attack exploits neighborhood information of CF schemes [13]. This attack assumes that history (ratings) of a target user is known or disclosed by an attacker. *k* fake users with identical to the target user are appended to the CF system and a prediction is asked for one of the fake users. It is expected that neighborhood will be formed from *k*-1 fake users and the target user. As a result, the predictions are expected to be produced from the target user because unrevealed ratings of her will be revealed. The third attack exploits the highest correlations, *perfect matches*, between an incoming query and users. Based on captured *perfect matches*, ratings in the incoming query could be reconstructed by carrying out intensive repeated queries.

In [18], the authors perform these attack scenarios on PPCF schemes, where binary data is horizontally partitioned between two-parties. In this study, our aim is to show *how much privacy can be achieved in terms of the first and the second aspect of privacy by the vertically partitioned data-based PPCF binary schemes in* [19, 20]. Note that data is partitioned between two parties by devising different attack techniques. We discuss possible attacks targeting these schemes and perform some experiments to display the results.

The paper is organized as follows. The next section covers the related work in the field. Section 3 introduces the target PPCF schemes. Section 4 clarifies attack techniques and Section 5 displays experiments. Section 6 gives a discussion about results and makes a comparison with a random attack. The last section lays out conclusions and general summary of the study.

2. Related Work

PPCF community offers different solutions to enhance the privacy of CF systems. Polat and Du [15] offer a method, which can be applied for numerically rated data. They propose a randomized perturbation technique, where each user calculates their z-scores and sends them to the server by adding up some random noise to the original z-scores. Other schemes for the central server-based PPCF utilize binary data [16, 21]. In [16, 21], the researchers apply RRTs on binary data to disguise ratings. RRT is a survey technique, proposed by Warner [17], to determine a sensitive attribute in a population. To calculate predictions, the probabilities due to RRTs are used in [16] while naïve Bayesian classifier (NBC) is utilized in [21]. In [22], the authors propose an item-based scheme claiming that item relationship is not sensitive.

Sparse data sets are obstacle for CF systems and companies planning to embark on new markets or newly established ones might lack enough data to provide accurate recommendations [6]. Hence, they might collaborate for better filtering services. Two companies could have ratings for the same set of items by different customers. This is called horizontally partitioned data (HPD). Likewise, if two parties hold ratings of the same users for different sets of items, then this partitioning is called vertically partitioned data (VPD). There are some studies providing different schemes to offer recommendation in both HPD and VPD cases [19, 20, 23, 24]. Two-party binary PPCF schemes are presented in [19, 20, 23]. VPD-based PPCF scheme for numerically rated data is studied in [19]. Kaleli and Polat [20] offer NBC-based scheme for both HPD and VPD. Polat and Du [23] propose a PPCF scheme for HPD. The scholars also propose multi-party schemes [25, 26, 27]. The authors in [25] utilize NBC for both HPD and VPD. Self-organizing maps-based recommendations are proposed for HPD [26] and VPD [27]. A detailed survey about PPCF schemes is presented in [6].

Privacy is meant to be preserved by aforementioned techniques. However, a group of researchers examine if privacy is really protected. In privacy-preserving data mining community, Kargupta et al. [10] propose an SF technique to extract the original data perturbed by random perturbation. Their method is based on obtaining theoretical boundaries of maximum and minimum values of eigenvalues of the noise matrix. They extend their study for discrete graph structure [11]. Some researchers study the bounds of the reconstruction error by SF methods [28, 29]. Principal component analysis is also utilized to reconstruct the original data by exploiting data correlations [30]. When the correlation is high, reconstructions that are more accurate can be performed for random perturbation.

Zhang et al. [12] target a PPCF scheme proposed by Polat and Du [15]. They utilize singular value decomposition and *k*-means clustering to reconstruct original data. They apply *k*-means clustering to get the data in groups for discrete and continuous valued data. This method can be applied to discrete data without any modification; however, the continuous data need some preprocessing. They discretized the continuous data into *k* segment and an item is assigned to the median value of the segment it belongs to after clustering. Calandrino et al. [13] target live CF systems by exploiting auxiliary information. They propose passive inference attacks that exploit the temporal changes in the output that CF systems make publicly available. Binary PPCF scheme proposed in [16] is investigated in terms of disclosing the rated items [14]. The authors utilize publicly collected information about the target data set and manage to retrieve this private information.

The aforementioned attacks generally focus on the systems with central data. The study in [18] handles how much privacy is offered when data is partitioned horizontally between two parties. They utilize possible attack techniques (*acting as an active user* and *knn-based*) and propose an attack technique called *perfect match* attack. Our approach in this paper focusses on vertically partitioned binary ratings between two parties. We extend the prior studies [18, 31] for a VPD-based binary PPCF scheme. This study covers and extends the attack technique given in [31] for two-party PPCF schemes. Unlike [31], we have added two-party binary PPCF NBC prediction [20] and try to derive target site's matrix as a whole. In [31], the primary intention is not to build the target site's data matrix; it aims to show the applicability of the attacks.

3. Preliminaries

The first targeted scheme is proposed by Polat and Du [19]. This scheme provides top-*N* recommendation (TN) for an active user, *AU*, among the item list (N_a) she wants a prediction. The second targeted scheme is based on NBC to provide predictions on partitioned data [20]. These schemes employ privacy measures to prevent the other party from disclosing similarity information. From now on, *A* and *B* will denote each party. First, we introduce the method introduced in [19] to offer private TN for two-party PPCF.

3.1. TN recommendation

This scheme selects the users who have high positive and negative correlations with *AU* claiming that accuracy might be increased if the best similar and dissimilar users are selected [19]. The similarity metric to determine neighbors is a modification of Tanimoto coefficient as follows:

$$W_{au} = \frac{t(R_s) - t(R_d)}{t(R)} \qquad (1)$$

In Eq. (1), W_{au} is the similarity weight between the user *u* and *AU*. $t(R_s)$, $t(R_d)$, and $t(R)$ are the numbers of similarly, dissimilarly, and commonly rated items by both *u* and *AU*, respectively. If $W_{au} > 0$, then *u* and *AU* are similar, otherwise they are dissimilar. They are not correlated if the similarity weight is 0.

After determining W_{au}, neighbors are picked based on two different criteria, *best-N* or *threshold*. In the *best-N* neighbors' selection, *N* users with the highest correlations (either positive or negative) are picked as neighbors. *Threshold* neighbors' selection method picks its neighbors among the users whose correlations (either positive or negative) surpassing a threshold (τ_n) value. Note that users with negative correlations with *AU* are dissimilar to *AU*. These users would vote opposite; therefore, their ratings are reversed. Since the scheme handles binary data, reversing can be performed by converting *likes* (*1s*) to *dislikes* (*0s*) and *dislikes* (*0s*) to *likes* (*1s*).

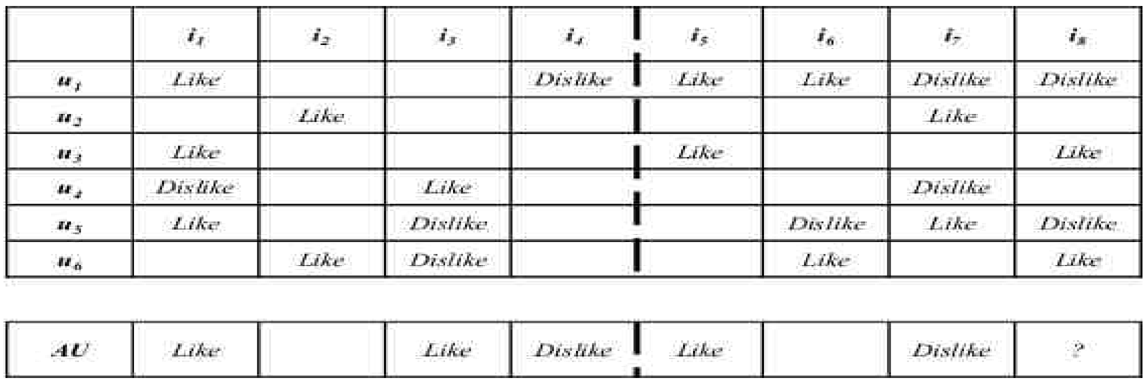
Upon selecting the neighbors, Polat and Du [19] find the number of *likes* (l_j) and *dislikes* (d_j), where *j* is the item number, among the selected neighbors. Then, $ld_j = l_j - d_j$ is calculated. If $ld_j > 0$, the item will be liked by *AU*. Otherwise, it will be disliked.

There are two different cases based on how N_a items are shared between parties. The first deals with the case, where all N_a items belong to the one of the parties. The second case is designed when N_a items are shared between parties. These cases will be hereafter called as the first case *Case-All* and the second *Case-Split*.

3.1.1. Case-All

N_a items, for which *AU* is looking for referrals, might belong to one party. *Case-All* deals with this special case assuming that *B* has all items of N_a and *A* has none [19].

- ➢ *AU* sends her corresponding ratings to both parties and N_a to only *B*. *A* computes the



	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	*Like*			*Dislike*	*Like*	*Like*	*Dislike*	*Dislike*
u_2		*Like*					*Like*	
u_3	*Like*				*Like*			*Like*
u_4	*Dislike*		*Like*				*Dislike*	
u_5	*Like*		*Dislike*			*Dislike*	*Like*	*Dislike*
u_6		*Like*	*Dislike*			*Like*		*Like*

AU	*Like*		*Like*	*Dislike*	*Like*		*Dislike*	?

Fig. 1. NBC-based CF

required values (partial similarity) utilizing a privacy protocol called private similarity calculation protocol (PSCP).

- *A* sends the partial similarity values to *B* through *AU*. *B* finds its own partial similarity values between users it holds and *AU*. Then, *B* calculates the final similarities (W_{au}) adding the partial similarity values received from *A* to its own calculated ones.
- After finding the similarities, *B* selects the best neighbors using the *threshold* and the *best*-N_n approaches. It uses random τ_n and N_n to prevent *A* from learning them. *B* generates a random number of $r_{B\_}\tau_n$ from a range [-α_B, α_B] and adds it to τ_n. Likewise, *B* adds a random number, r_{B_Nn}, to the number of the best neighbors, N_n, to be selected to mask how many best users are picked. *B* picks r_{B_Nn} among a range [-γ_B, γ_B].
- ld_j values are calculated and sorted by *B*. TN recommendation is returned to *AU*.

3.1.2. Case-Split

While the previous case designed when all N_a items belong to a single party, this one handles the case when these items are split between parties [19].

- *AU* sends a query and her ratings to both parties. *B* finds the partial similarities between its users and *AU* using PSCP. Partial similarities are sent to *AU* and *AU* lets *A* know partial similarities.
- *A* computes its own partial similarities and finds the similarities (W_{au}) by adding values from *B*.
- *A* selects the best N_n neighbors by employing random τ_n and fixed N_n values. Since *B* needs the neighbor information, *A* lets *B* know which neighbors are selected and the similarity signs.
- *A* forms a neighborhood by employing random N_n and τ_n. *A* computes ld_{Aj} with this new neighborhood and lets *B* know ld_{Aj} values. Since data is partitioned vertically, *B* needs to know ld_{Aj} values, which are the other party's aggregated values to come up with final ld_j values. After receiving ld_{Aj} values, *B* calculates ld_j values by adding ld_{Aj} values from *A* to the corresponding ld_{Bj} values it has computed. *B* finally sends TN list to *AU*.

3.2. NBC-based prediction

NBC can be utilized for CF purposes [32]. Kaleli and Polat [20] also employ NBC (*Case-NBC*) for two-party VPD-based schemes considering privacy in their study. In this scheme, users correspond to features and items correspond to feature values. An illustration of *Case-NBC* is given in Fig. 1. *AU* is looking for a prediction for i_8, which is marked with a question mark. The equation whether an item belongs to a class (*cl*) for a non-partitioned centralized data, where *cl* is *like* or *dislike* can be described as follows:

$$p(cl \mid f_1, f_2, ..., f_{n-1}, f_n) = p(cl)\prod_{i}^{n} p(f_i \mid cl) \qquad (2)$$

In Eq. (2), $p(cl)$ is the prior probability of *like or dislike* based on *cl*, which can be calculated from the active query. f_i is the rating of the queried item, *q*, which is i_8 in Fig. 1. Probabilities are only calculated if *q* is rated. Therefore, unrated f_i values (i_8) are not taken into account. The conditional probability will only be calculated for f_1 = *dislike*, f_3 = *like*, f_5 = *dislike*, and f_6 = *like* in Fig.1 and the repeated multiplication from *i* to *n* thus covers 1, 3, 5, and 6 in Eq. (2).

Assume that V_i is a vector, where *i* is associated with the user and *rated*(r_{ij}) is a function that takes an item value (r_{ij}) as an argument and returns *true* when an item is rated or false otherwise. Vector definitions given below are utilized to calculate conditional probabilities.

1. $V_i=\{r_{ij}: rated(r_{ij})=true, i=\{1,2,…, n\}, j=\{1,2…,m\}\}$
2. $(V_i)_{cl} = \{r_{ij}: cl \in \{Like, Dislike\}, r_{ij}=\{cl\}, i=\{1,2,…, n\}, j=\{1,2…,m\}\}$

Based on these definitions, the following set of equations display how conditional probability, $p(f_i|cl)$, can be reached for each feature vector:

$$\begin{aligned} & f_i = V_{iq}, q \text{ is the queried item} \\ & (D_i)_{cl} = (AU)_{cl} \cap V_i \\ & (N_i)_{cl} = (D_i)_{cl} \cap (V_i)_{f_i} \\ & \#(D_i)_{cl} = n((D_i)_{cl}) \\ & \#(N_i)_{cl} = n((N_i)_{cl}) \\ & p(f_i \mid cl) = \frac{\#(N_i)_{cl}}{\#(D_i)_{cl}} \end{aligned} \tag{3}$$

Eq. (3) lays out the calculation of *p(f$_i$|cl)* by breaking down numerator, *(#N$_i$)$_{cl}$*, and denominator, *#(D$_i$)$_{cl}$*. *(D$_i$)$_{cl}$* is the intersect set of items rated by *AU* as *cl∈ {like, dislike}* and rated by *i-th* user regardless of being *like* or *dislike*. *(N$_i$)$_{cl}$* is the intersect set of *(D$_i$)$_{cl}$* and *i-th* user's vector items rated same as *f$_i$*. This set is picked to quantify similarity between *AU* and users by intersecting *f$_i$*'s in a user vector. After finding out these sets, *n()* is a function determining number of elements in a set. To illustrate, *p(f$_1$ = like | cl $_=$ like) = 2/2* and *p(f$_1$ = dislike | cl $_=$ like) = 0/2*. To avoid multiplication following features by 0, Laplace smoothing could be utilized [32]. Conditional probability calculation needs to be repeated for the remaining *f$_i$*'s to obtain the final probability. The item is assigned to the class with the highest probability.

When data is partitioned vertically between two parties as depicted by a dashed line in Fig. 1, each *p(f$_i$|c$_j$)* has to be calculated collaboratively because only one of the parties knows if the rating of *q* is *like* or *dislike*. Therefore, the party who does not know the rating of *q* has to calculate partial, *(#N$_i$)$_{cl}$* and *#(D$_i$)$_{cl}$* values and let the other party know them. Once partial *(#N$_i$)$_{cl}$* and *#(D$_i$)$_{cl}$* are received, the party with *q* adds these values to the ones calculated by itself. In this scheme, the party having *q* acts as a master site. The full application of the scheme is given in the following assuming that *A* is the master party [20]:

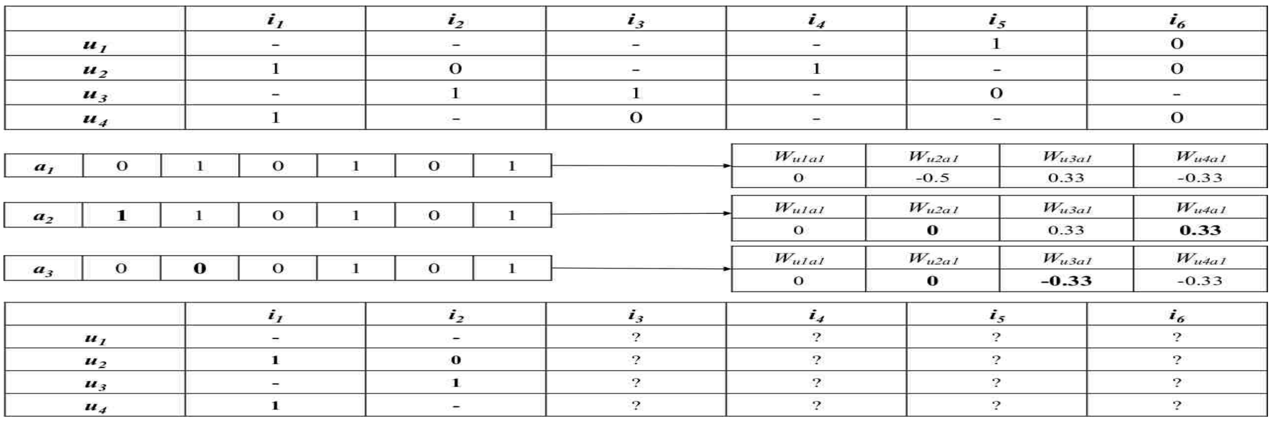
- *AU* sends her query to *A* and *B*. *AU* also computes *p*(*cl*) and sends it to *A*.
- Since *B* does not know the value of *q*, it computes partial probability values for *f$_i$* is *like* and *dislike*. For class membership assignment, conditional probability has to be calculated considering *cl* is *like* and *dislike*. Thus, four calculation of *p*(*f$_i$*=*like*|*cl*) is needed to obtain the partial conditional probability. However, notice that *p*(*f$_i$*=*like*|*cl*)+*p*(*f$_i$*=*dislike*|*cl*)=1; thus, it is enough to calculate for only *f$_i$*=*like* or *f$_i$*=*dislike* for *cl*, *like* and *dislike.* Because *A* can disclose *B*'s data matrix by observing partial probability values, *B* utilize a privacy protocol very similar to PSCP to prevent data disclosure.
- Upon getting partial conditional probabilities from *B*, the master party *A* picks true *#(N$_i$)$_{cl}$* and *#(D$_i$)$_{cl}$* values based on value of *q* and calculates final conditional probabilities.

3.3. Privacy by perturbing active query

To prevent data disclosure, the authors in [19, 20] propose techniques by perturbing an active query so that results from the active query do not really reflect the exact relation with the original user vector. Both studies present a similar solution to perturb the active query. The scholars discuss PSCP that ratings should be appended to or removed from the active query based on its density [19]. If the active query is dense, meaning that more than half of the items are rated, then some items are removed according to a random number drawn from [1, *M*], where *M* is the number of rated items in the active query. In the case of sparse active query, some items are randomly appended.

In *Case-NBC* [20], a similar approach is employed by appending default votes to the active query by a random percentage drawn from [1, 100]. These two approaches have similar foundations; however, the active queries are mostly very sparse and removing ratings could barely occur. On the other hand, appending a random percentage of ratings from a larger range such as [1, 100] would be misleading due to sparse nature of active queries.

These approaches have evolved to a more coherent form in [25] by associating the volume of ratings to be appended to the density, *d*, of an active query. This protocol is called *hiding rated items* (HRI). First, the number of unrated items is determined. Then, a random value is drawn from the range [1, *δ*], where *δ* might be factors of *d* such as 1/8*d*, 1/4*d*, 1/2*d*, or *d*. The unrated cells are filled up to a percent drawn between [1, *δ*]. Therefore, *AU*'s query is filled with *δ*/2 on average. This protocol makes a connection between density and ratings to be appended due to



	i_1	i_2	i_3	i_4	i_5	i_6
u_1	-	-	-	-	1	0
u_2	1	0	-	1	-	0
u_3	-	1	1	-	0	-
u_4	1	-	0	-	-	0

a_1	0	1	0	1	0	1

W_{u1a1}	W_{u2a1}	W_{u3a1}	W_{u4a1}
0	-0.5	0.33	-0.33

a_2	**1**	1	0	1	0	1

W_{u1a1}	W_{u2a1}	W_{u3a1}	W_{u4a1}
0	**0**	0.33	**0.33**

a_3	0	**0**	0	1	0	1

W_{u1a1}	W_{u2a1}	W_{u3a1}	W_{u4a1}
0	**0**	**-0.33**	-0.33

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	-	-	?	?	?	?
u_2	**1**	**0**	?	?	?	?
u_3	-	**1**	?	?	?	?
u_4	**1**	-	?	?	?	?

Fig. 2. Attack – acting as an active user in multiple scenarios

sparse nature of CF active queries. In this study, HRI protocol is utilized to perturb active queries instead of individual privacy methods by each PPCF technique.

In addition to HRI protocol, the authors in [25] propose to disguise *AU*'s query. They propose to utilize RRT [17] by partitioning the query into groups. The proposed RRT protocol works as follows:

- The master site defines *G* or number of groups.
- *AU*'s query is divided into *G* groups.
- For each group, *g*, two uniformly random values β_g and Θ_g values are drawn.
- If $\Theta_g > \beta_g$, ratings are reversed for *g*-th group.

Since which groups are reversed or preserved is known by the master party, it can later manipulate the interim results calculated by the other party to make corrections.

4. Attack Scenarios

We describe three attack scenarios that can be applied to VPD-based PPCF schemes; and evaluate their performance in terms of privacy. Privacy has two aspects [16]: (1) preserving the actual rating made by users and (2) disguising if an item is rated or not. We will refer to them as *the first* and *the second aspect* of the privacy, respectively.

4.1. Acting as an active user in multiple scenarios

If there exists a malicious party, it can try sending multiple queries to learn the other party's matrix. Consider a case where the malicious party sends multiple queries and alters only one cell each time. In such a case, the malicious party can track the changes in the output (similarity weights) and decide items' rating whose values have been manipulated. Assume that the malicious party invokes an initial query and stores the similarity weights. Then, the malicious party manipulates a single rating and sends the altered query to the other party to learn the new similarity weights. After receiving the similarity weights for the manipulated query, it compares them with the ones from the earlier query. If there is an increase in the similarity weight between *AU* and *u*, then the manipulated value is kept by the user. The malicious party can reach such a decision because the increase in the similarity weights means a higher correlation between *AU* and *u*. If there is a decrease, then the malicious party concludes that *u* has the value in the first query. If there is no change in the similarity value, this means that the manipulated item is not rated by *u*. This notion can be applied for each user so that the whole matrix can be disclosed as depicted in Fig. 2. This attack is a threat for the first and the second aspects of privacy because it reveals both the actual ratings and if an item is rated.

Case-All, *Case*-Split, and *Case-NBC* VPD-based PPCF schemes [19, 20] are subjected to this attack. Remember that data is vertically partitioned between two parties. Therefore, the collaborating parties have to exchange the partial calculations of individual users to obtain the results for both schemes. This interaction makes it possible to perform this attack because the malicious party will have an access to the partial information for each user. As a measure of privacy, HRI will be applied as discussed in the previous section. By HRI, active query is appended based on δ, which is related to d. Hence, it is expected that the success of this attack will be affected by increasing δ values. We examine how effective HRI is against this attack by trails.

4.2. knn-based scenarios

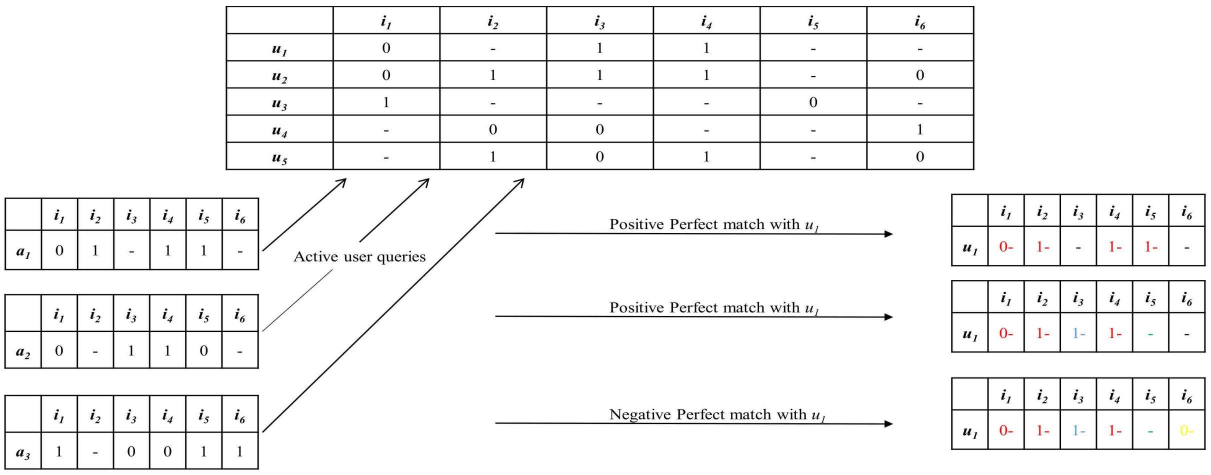
knn-based attack is proposed by Calandrino et al. [13]. This attack targets CF systems that select k nearest neighbors. It assumes that the attacker has a history of ratings of a target user and appends k fake users into the CF system with an exact copy of known history. When a prediction is requested for one of the fake users, it is highly probable that k nearest neighbors will be selected among k-1 fake users and the target user. Since k-1 users are identical, the predictions are expected to come from the target user. This attack discloses whether an item is rated or not, so the second aspect of privacy is under threat with this attack scenario.

Since *Case-All* and *Case-Split* schemes targeted in this paper utilize the best neighbors approach, *knn-based* attack can be performed. As *Case-NBC* does not make use of neighborhood approach, this attack is not valid for it. Although *knn-based* attack needs a history of a user, the attacker does have such a history inherently in VPD-based schemes. Owing to the vertical partitioning, where items are split between parties, the parties have already had such a history of each user. Thus, the malicious party can use its own part of the ratings as the history of the target user and manipulate neighbors by inserting k fake users into system. We hypothesize that HRI is not an effective privacy measure to prevent from this attack because randomly removing or appending ratings into *AU*'s vector does not change the similarity weights between *AU* and k-1 users plus the target user.

4.3. Perfect match attack

This attack is proposed in [18] for an HPD-based binary PPCF scheme. We apply this attack for VPD-based schemes in this study. Although HRI is applied by each party as a privacy measure, this scheme is subjected to *perfect match* attack. Assume that *B* acts as the master party and no privacy measure is taken. *A* calculates the similarities between its user and *AU*, sends them to *B*. If the similarity between any user of *A* and *AU* is either 1 or -1, such similarities are called as *perfect matches* [18]. This means that the commonly rated items between these two users are either identical or opposite, respectively. Hence, *B* can conclude that the corresponding user who has a *perfect match* with *AU* either identically voted or not voted for any of its items if the similarity is 1. If the similarity is -1, they either vote opposite or not voted for any of the items. The attack is depicted in Fig. 3.

In Fig. 3, three active queries are listed. Note that the similarity weights between u_1 and a_1 and u_1 and a_2 are 1. These are all *perfect matches*. Once the first two active queries are sent, the malicious party finds out that i_5 is not rated


Active user queries
Positive Perfect match with u_1
Positive Perfect match with u_1
Negative Perfect match with u_1

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	-	1	1	-	-
u_2	0	1	1	1	-	0
u_3	1	-	-	-	0	-
u_4	-	0	0	-	-	1
u_5	-	1	0	1	-	0

	i_1	i_2	i_3	i_4	i_5	i_6
a_1	0	1	-	1	1	-

	i_1	i_2	i_3	i_4	i_5	i_6
a_2	0	-	1	1	0	-

	i_1	i_2	i_3	i_4	i_5	i_6
a_3	1	-	0	0	1	1

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0-	1-	-	1-	1-	-

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0-	1-	1-	1-	-	-

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0-	1-	1-	1-	-	0-

Fig. 3. Perfect match attack

because the similarity metric is calculated based on commonly rated items and i_5 is not identical in a_1's and a_2's rating vectors. On the other hand, it is also concluded that i_1, i_2, i_3, and i_4 are rated *dislike* (0), *like* (1), *like* (1), and *like* (1), respectively or unrated. The malicious party has no idea about i_6 before dispatching the third query. After capturing a *perfect match* with a_3, i_6 is disclosed as rated *dislike* or unrated. This attack discloses two privacy breaches: (1) even a single *perfect match* reveals that the actual rated value of the target item or it is unrated and (2) unrated entries might be disclosed if multiple *perfect matches* occur as it happens to i_5.

Case-All, *Case*-Split, and *Case-NBC* schemes calculate the interim value by two-party collaboration. One party calculates its own partial similarities and sends them to the master party for a final calculation. As a result, final calculations are performed by using the incoming partial values with the ones calculated off-line by the master party. The master party can use these interim similarity values received from the other party to identify *perfect matches* in all of three attacks. Thus, *perfect match* attack is applicable for all two-party VDP schemes discussed in this study.

5. Experiments

5.1. Data sets and evaluation criteria

Experiments were performed using MovieLens Million (MLM) and Netflix data sets. MLM was collected by GroupLens research group (www.cs.umn.edu/research/GroupLens). Netflix is a challenge data set to improve the accuracy of prediction on a web-based movie service (http://www.netflixprize.com/). Both data sets are movie-rating data sets on a 5-star scale. Netflix was selected to represent a rather sparse data set with a density of 1.08% while MLM represents a data set with a regular density (4.19%) for a CF system. MLM data set contains about a million of rating from 6,040 users for 3,952 users while Netflix has 480,189 users and 17,770 items.

Three different evaluation metrics were utilized in this study, *accuracy*, *precision*, and *recall* for the first and the second aspects of privacy. *Accuracy* measures how much the original data is recovered. Each item in the derived data set is compared to its original value to calculate *accuracy*. Since CF data sets are usually sparse, *accuracy* results are dominated by unrated entries. Therefore, we used *precision* and *recall* to evaluate the attacks in more detail. In this study, *precision* and *recall* was calculated by only determining *likes* and *dislikes* without considering unrated item due to their domination. *Precision* is the ratio of correctly recovered items to total items recovered in terms of *likes* and *dislikes*. *Precision* is important for an attacker to determine how much of the derived data is indeed genuine. *Recall* is the ratio of correctly recovered items to the total original items. By *recall*, it can be evaluated how good *likes* and *dislikes* are derived. *Recall* could be considered important for a target site because it shows the percentage of correctly derived items to the original data set. An attacker might end up a high *precision*; however, *recall* could be very low which means that derived items for the attacker constitutes a small margin of the original data.

Table 1 displays two confusion matrices for the first and the second aspects of privacy. Eq. (4) shows how three evaluation metrics are calculated based on Table 1. *Accuracy* values are calculated over all diagonal values to the all matrix summation in both cases. For the first aspect of

Table 1. Confusion matrices

a) First aspect of privacy

		Original		
		Likes	**Dislikes**	**Unrated**
Classified	**Likes**	V_{11}	V_{12}	V_{13}
	Dislikes	V_{21}	V_{22}	V_{23}
	Unrated	V_{31}	V_{32}	V_{33}

b) Second aspect of privacy

		Original	
		Rated	**Unrated**
Classified	**Rated**	Z_{11}	Z_{12}
	Unrated	Z_{21}	Z_{22}

privacy, *precision* and *recall* values are only calculated for correctly identified *likes* and *dislikes* to eliminate the unrated items' domination in the *accuracy*. *Precision* and *recall* metrics are calculated by dividing correctly classified *likes* and *dislikes* by summing row values and column values of *likes* and *dislike* in Table 1, respectively as given in Eq. (4).

$$Acc_1 = \frac{\sum_{i=1}^{3} V_{ii}}{\sum_{i=1}^{3}\sum_{j=1}^{3} V_{ij}},\ Acc_2 = \frac{\sum_{i=1}^{2} Z_{ii}}{\sum_{i=1}^{2}\sum_{j=1}^{2} Z_{ij}}$$

$$prec_1 = \frac{\sum_{i=1}^{2} V_{ii}}{\sum_{i=1}^{2}\sum_{j=1}^{3} V_{ij}},\ prec_2 = \frac{Z_{11}}{Z_{11}+Z_{12}} \qquad (4)$$

$$rec_1 = \frac{\sum_{i=1}^{2} V_{ii}}{\sum_{i=1}^{3}\sum_{j=1}^{2} V_{ij}},\ rec_2 = \frac{Z_{11}}{Z_{11}+Z_{21}}$$

MLM and Netflix data sets are on a numeric scale between 1 and 5; however, the concentration in this study is on binary data. Thus, ratings were converted to a binary scale [32]. Netflix and MLM ratings greater than or equal to 3 were converted to 1 (*like*) and the rest converted to 0 (*dislike*). 1,000 random users were picked for the experiments from both data sets. Item size and densities of selected data sets with 1,000 users are given in Table 2.

Recall that *perfect match* attack might disclose that (1) an item is unrated or its possible actual value or (2) an item is rated or not if multiple positive and negative *perfect matches* occur for it. A coin toss is performed to determine the value for an item for case (1) so that evaluation metrics can be calculated for the first aspect of privacy. Items falling into case (2) are marked as unrated and items falling into case (1) are marked as rated to calculate the second aspect of privacy.

5.2. *Methodology*

Experiments were repeated 100 times and N_n, number of neighbors, was set to 200 for *Case-All* and *Case-Split*. Experiments are given in three different sub-titles. Throughout the experiments,

Table 2. Data sets

	Item Size	Overall Density	Like Density	Dislike Density
MLM	3,952	0.0434	0.0253	0.0181
Netflix	17,770	0.0108	0.0064	0.0044

three different parameters were controlled. The first is how varying δ could affect the evaluation metrics. The second and the third are how number of groups, *G*, and how varying densities of each party affect the success of data recovery, respectively. For the first case, HRI was utilized by varying δ. HRI provides privacy by appending items to the active query. The amount of items to be inserted is determined by δ, which is related to the density. Therefore, increasing δ values should display an inclination toward privacy. This hypothesis will be tested when data is equally partitioned by varying δ between 0*d*, 0.125*d*, 0.25*d*, 0.5*d*, and 1*d*, where 0*d* means no privacy measures have been applied for the scheme. The second control parameter is *G* or number of groups. *G* will be varied between 1, 3, 5, 7, and 10. The last control parameter will be the effects of varying the *target party distribution* (TPD). TPDs will be manipulated between 0.125, 0.25, 0.5, 0.75, and 0.875 while δ = 0.125*d* and three different attacks will be monitored against varying values of TPD. Results in terms of the first and thesecond aspect of privacy will be reported throughout the experiments if applicable.

5.3. *Experiments*

5.3.1. *Varying δ values*

This experiment displays how three PPCF schemes are affected by varying values of δ when attacks are activated. It is expected that both accuracy, precision, and recall will decrease for increasing δ due to altered active query by HRI. *AU* query will contain more appended ratings for larger δ values so that it will differ more from the original *AU* query. In Table 3, results are displayed.

After δ is met, metrics are reported to decline for both MLM and Netflix data sets for all attack types. Metrics have a general tendency toward decline for large δ values. It can be noted that appending random ratings via HRI have a negative effect for reconstructions. Decrease in all metrics are the most obvious after δ is met for *acting as an active user* attack compared to the other two attacks. *Acting as an active user* attack guarantees a full recovery if no privacy measures are taken. Therefore, decline for large δ is more noticeable for this attack. For MLM data set, precision and recall values remain almost stable until δ=1d for both aspects of privacy; however, decline in these metrics are more observable when δ=1d. On the other hand, accuracy reports more observable declines after δ=0.5d. Netflix, the sparse data set, demonstrates decline after privacy measures are introduced; however, all metrics follow more stable trend for all δ values in this experiment. We attribute this to sparsity of Netflix data set. Recall that *knn-based* attack is built upon exploiting neighborhood by trying to inject k-1 users into the neighbors and this attack is applicable for *Case-All* and *Case-Split*.

Although HRI protocol appends some ratings to the active query, such an effort does not affect the neighborhood of k-1 fake users with the target user. Similarity scores between *AU* and k-1 fake users before and after HRI stay same because *Case-All* and *Case-Split* schemes do only consider commonly rated items. However, appending ratings each time will alter similarity score between *AU* and other users. TN recommendations therefore might differ and success of *knn-based* attack depends on TN recommendations. After δ is met, *knn-based* attack records declines in terms of evaluation metrics for both data sets and all evaluation metrics. *knn-based* attack performs similar outcomes for increasing density rates including δ=1d unlike *acting as an active user* attack for precision, recall, and accuracy for both privacy aspects and for both data sets.

Perfect match attack for three PPCF schemes in this study demonstrates decline after δ is introduced similar to previous attacks. *Case-All*, *Case-Split*, and *Case-NBC* achieve very high accuracy rates above 0.9 in all δ values compared to the previous attacks. Additionally, accuracy could be considered acting steadily for larger δ values for both data sets. In terms of recall, *Case-NBC* follows a constant and higher trend than the other two schemes for both data sets. Beside accuracy and recall, precision rates perform higher ranges compared to *acting as an active user* and *knn-based* attacks in terms of both aspects of privacy.

In general, our intuition about increasing δ values will cause reconstruction to deteriorate is not accomplished as much as expected for the range between δ=0.125d and δ=0.5d for most of the cases. However, we believe the decline trend would be more apparent if larger δ were chosen. The reason why δ values are varied up to δ=1d is to set up an experiment environment close to a PPCF realities. In terms of PPCF, both privacy and prediciton accuracy are considered. Setting privacy measures to great extents inevitably affects recommendation accuracy. Therefore, it is aimed to set up an experiment to mimic a reasonable PPCF environment.

5.3.2. *Varying number of groups*

RRT protocol is proposed to disguise *AU*'s query by dividing into different *G*. As *G* increases, one can claim that *AU*'s query become more private due to increased grouping. In this experiment, it is tested how increasing *G* values have an effect on reconstruction. Therefore, *G* values are varied between 1, 3, 5, 7, and 10.

Remember that HRI protocol appends ratings into *AU*'s query up to δ which is associated to density d. An *AU*'s query is filled with an average of δ/2 fake ratings, which contribute to privacy and harm reconstruction. However, we hypothesize that increasing *G* values under a constant δ will help reconstruction results. Notice that interim values will be calculated for each group g because the master party knows which group is reversed or preserved. When *G* is increased, the possibility of each group to be appended by random ratings by HRI decreases. Assume that G=m, where m is number of items, interim calculations are made for each group and none of rated groups/items are manipulated by HRI if *G* is m. Only unrated groups are appended by HRI. Since the master party knows true *AU*'s query and which items are indeed rated or not, it can easily capture true interim results. Table 4 displays the experimental results, where δ = 0.25 and TPD = 0.5. *G* is increased up to 10 due to runtime costs.

Table 4 clearly demonstrates that *acting as an active user* attack, which manipulates an item at a time, performs as expected. For all three metrics and PPCF algorithms, the best performing cases is the one, where *G* is the maximum for both data sets. Besides, all metrics for the first and the second aspect of privacy demonstrate an increasing pattern for ML and Netflix data sets for larger *G*.

knn-based attack performs quite similar with slight margins for varying *G* values for MLM and Netflix data sets. Note that evaluation metrics are only calculated for the second aspect of privacy because this attack can reveal if an item is rated or not instead of disclosing its value. When compared by data sets, *knn-based* attack yields better output for MLM than Netflix, which is a sparse data set.

Perfect match attack has almost an increasing trend for *Case-All* and *Case-Split* for all metrics in MLM data set. The best performing *G* value only alternate between 7 and 10. For the sparse data set Netflix, the trend fluctuates. Our assumption for *Case-All* and *Case-Split* is not obvious, this might be due to large number of items that Netflix has. Since *perfect match* attack captures *perfect match*es for each group, *G*=10 could be insufficient. On the other hand, *Case-NBC* demonstrates a clear declining trend for larger *G* values up to 10 contrary to our hypothesis in terms of precision and accuracy. Recorded recall values for *Case-NBC* could be considered in an increasing trend by small margins.

5.3.3. Varying TPD

In the previous experiments, it is assumed that data is equally partitioned between parties. However, this case might occur rarely. In this experiment, TPD values are varied to observe how different densities affect the evaluation metrics. TPD values are varied between 0.125, 0.25, 0.5, 0.75, and 0.875 for each PPCF algorithm. δ is set 0.25 and *G*=5. Table 5 displays the results.

Acting as an active user attack displays the best results when TPD = 0.125 for all evaluation metrics and data sets. For larger TPD values, a decline in precision, recall, and accuracy values is observed. *Case-NBC* seems to be more prone to *acting as an active user* attack compared to *Case-All* and *Case-Split* algorithms for both MLM and Netflix data sets considering evaluation metrics of the first and the second aspect of privacy.

knn-based attack, which seems so far more resilient to privacy measures, performs a general trend toward declining for larger TPD values for *Case-All* algorithm for both data sets. For MLM data set, *Case-Split* performs the best for precision and accuracy at TPD=0.250. Recall results with *Case-Split* in MLM data set is the best at TPD=0.500. On the other hand, both *Case-All* and *Case-Split* record decline for Netflix data set for larger TPD values.

Perfect match attack with MLM data set demonstrates the best reconstruction rates when TPD = 0.125. However, *Case-NBC* is the only exception with precision and accuracy rates in terms of the first and the second aspect of privacy. The sparse data set Netflix does not show any reliable pattern common to all metrics. While rec_1 and rec_2 perform the best when TPD = 0.125. Accuracy and precision metrics display better results toward larger TPDs. *Case-Split* and *Case-All* display similar trends with each other in terms of accuracy, recall, and precision. While recall and precision decrease for larger TPDs in terms of the first and the second aspect of privacy, accuracy follows a steadier trend for MLM. For *Case-NBC*, accuracy has a slightly increasing trend in terms of the first and the second aspect of privacy for MLM. Contrary to MLM data set, *perfect match* attack for all PPCF algorithms shows an increasing behavior toward TPD=0.750 for precision and accuracy while it performs a slight decrease in terms of recall for both aspects of privacy.

Reconstruction metrics display relatively better results for smaller TPD values in general. In most of the cases, recall and accuracy metrics are promising. By this experiment, it could be stated that lower TPDs would be more prone to *acting as an active user* and *knn-based* attack for both data sets. However, *perfect match* attack performs better for representative MLM data set for lower TPD values while the trend reverses for larger TPD for the representative sparse data set Netflix.

6. Discussion

Throughout the experiments, most of the trials report higher recall and accuracy rates compared to precision. Note that precision relies on the ratio of relevant items retrieved to all retrieved items, which is marked as *like* or *dislike* by attack scenario. Precision is calculated for only *likes* and *dislike*. MLM data set's density is about 0.04 while Netflix density is about 0.01. The reason behind low precision rates is due to high density of unrated items. The overwhelming majority of the unrated items and marking them as *like* or *dislike* dominates this metric because of the modified calculation of precision as $prec_1$ and $prec_2$ for the first and the second aspect of privacy given in Eq. 4. Recall results for most of the cases are comparably higher than precision. Recall rates rely on the ratio of relevant retrieved items to the all relevant items, which are the items originally marked as *likes* or *dislike*. The reason why recall might outperform precision could be due to the fact that retrieved relevant items are compared to the originally rated items as *like* and *dislike*. Note that for precision, relevant retrieved items are compared to the all retrieved items, which are under the dominance of incorrectly marked unrated items. If recall result is higher than precision, then it can be understood that incorrectly retrieved unrated items as *like* or *dislike*, which is related to precision calculation, outnumber the number of all relevant items as *like* or *dislike*, which is associated to recall. Accuracy deals with correctly retrieved items to the total items. Therefore, it includes unrated items. Contrary to precision, better accuracy results rely on correctly retrieved unrated items count in the calculation of accuracy metrics.

It can be also discussed which attack type would be preferred based on different control parameters in this study. The first control parameter is to vary δ values to see how appending more random items into *AU*'s query effect the evaluation criteria. Although *acting as an active user* attack guarantees full data recovery when there are no privacy measures, its effectiveness especially in terms of precision and accuracy degrades. On the other hand, *knn-based* attack demonstrates a decline after δ is met; however, it displays more stable outputs for growing δ values. Notice that *knn-based* attack can only disclose if an item is rated or not.

Perfect match attack produces much better precision and accuracy results than the previous two attacks although recall is lower. A malicious party intending to disclose the other party's data could prefer this attack if precision is more important than recall. Precision could be preferred to recall if the attacker wants to be sure the higher ratio of retrieved relevant items to the all retrieved items (precision) instead of the ratio of retrieved relevant items to the all relevant items (recall). The authors in [12] discuss that precision is more important for an attacker. The other control parameters are *G* and TPD. Similar to varying δ parameter, *perfect match* attack makes a difference and beats the other attacks especially in terms of precision. Therefore, for all attack types and parameters, if the attacker puts precision into priority, then *perfect match* attack should be considered. Compared to *knn-based* attack to *acting as an active user* attack in terms of the second aspect of privacy, one can prefer *knn-based* attack for a more stable attack in most of the cases.

Beside the attacks discussed in the experiments, a malicious party could also devise a *random* attack. Since there are three possibilities (*unrated*, *like*, or *dislike*) that an item could have, each item could be assigned randomly. A possible and intuitional option could mark each item among three possibilities with prior knowledge of malicious party's density. Based on this idea, a *random* attack is implemented to compare how the attacks in this paper perform against a *random* predictor, which utilizes density rates of malicious parties. The *random* predictor works as follows: First, the attacking party finds out its overall densities of *likes*, *dislikes*, and *unrated* items. Second, the attacking party constructs a range for each density. Then, for each item, this attack generates a uniform random number in the interval (0, 1). Finally, item is assigned as *like*, *dislike*, or *unrated* based on random number. Random discovery creates a matrix from scratch. Table 6 displays a random discovery option to estimate an original target data matrix. Precision results in terms of both aspects of privacy is lower than recorded values for MLM and Netflix in attacks given in this paper. Similar to precision, very low recall values are recorded compared to previous attacks.

Table 3. Attacks with varying δ values

		MLM					**Netflix**				
	***δ*=**	**0**	**0.125d**	**0.25d**	**0.5d**	**1d**	**0**	**0.125d**	**0.25d**	**0.5d**	**1d**
Acting as an active user attack	**prec$_1$-Case-All**	**1.000**	0.069	0.069	0.069	0.057	**1.000**	0.019	0.019	0.019	0.019
	prec$_1$-Case-Split	**1.000**	0.069	0.068	0.067	0.056	**1.000**	0.019	0.019	0.019	0.019
	prec$_1$-Case-NBC	**1.000**	0.155	0.157	0.151	0.124	**1.000**	0.045	0.045	0.045	0.045
	rec$_1$-Case-All	**1.000**	0.836	0.836	0.834	0.795	**1.000**	0.790	0.790	0.789	0.790
	rec$_1$-Case-Split	**1.000**	0.835	0.835	0.830	0.793	**1.000**	0.788	0.789	0.788	0.790
	rec$_1$-Case-NBC	**1.000**	0.886	0.887	0.880	0.861	**1.000**	0.851	0.852	0.852	0.851
	Acc$_1$-Case-All	**1.000**	0.507	0.506	0.504	0.424	**1.000**	0.488	0.489	0.487	0.486
	Acc$_1$-Case-Split	**1.000**	0.509	0.506	0.494	0.417	**1.000**	0.484	0.489	0.484	0.486
	Acc$_1$-Case-NBC	**1.000**	0.789	0.794	0.785	0.737	**1.000**	0.773	0.774	0.773	0.775
	prec$_2$-Case-All	**1.000**	0.076	0.076	0.076	0.066	**1.000**	0.022	0.022	0.022	0.022
	prec$_2$-Case-Split	**1.000**	0.076	0.075	0.074	0.065	**1.000**	0.022	0.022	0.022	0.022
	prec$_2$-Case-NBC	**1.000**	0.175	0.177	0.172	0.144	**1.000**	0.052	0.052	0.052	0.053
	rec$_2$-Case-All	**1.000**	0.919	0.919	0.919	0.917	**1.000**	0.916	0.916	0.916	0.916
	rec$_2$-Case-Split	**1.000**	0.920	0.919	0.919	0.917	**1.000**	0.916	0.916	0.916	0.916
	rec$_2$-Case-NBC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Acc$_2$-Case-All	**1.000**	0.511	0.509	0.508	0.429	**1.000**	0.489	0.491	0.489	0.488
	Acc$_2$-Case-Split	**1.000**	0.512	0.510	0.498	0.422	**1.000**	0.485	0.491	0.486	0.674
	Acc$_2$-Case-NBC	**1.000**	0.794	0.798	0.790	0.743	**1.000**	0.775	0.776	0.774	0.776
***knn*-based attack**	**prec$_2$-Case-All**	**0.205**	0.150	0.153	0.153	0.152	**0.173**	0.079	0.079	0.079	0.079
	prec$_2$-Case-Split	**0.248**	0.180	0.181	0.182	0.185	**0.127**	0.087	0.089	0.091	0.090
	rec$_2$-Case-All	**0.975**	0.834	0.838	0.840	0.843	**0.962**	0.791	0.794	0.798	0.809
	rec$_2$-Case-Split	**0.998**	0.763	0.767	0.781	0.782	**0.998**	0.709	0.716	0.731	0.736
	Acc$_2$-Case-All	**0.834**	0.788	0.795	0.793	0.789	**0.942**	0.883	0.884	0.883	0.880
	Acc$_2$-Case-Split	**0.868**	0.839	0.841	0.838	0.840	**0.914**	0.904	0.905	0.906	0.904
Perfect match attack	**prec$_1$-Case-All**	**0.518**	0.427	0.426	0.422	0.389	**0.211**	0.157	0.158	0.156	0.156
	prec$_1$-Case-Split	**0.519**	0.427	0.427	0.424	0.388	**0.208**	0.157	0.157	0.156	0.158
	prec$_1$-Case-NBC	**0.677**	0.669	0.672	0.668	0.670	0.098	**0.120**	**0.120**	**0.120**	**0.120**
	rec$_1$-Case-All	**0.379**	0.344	0.343	0.343	0.327	**0.458**	0.367	0.367	0.366	0.365
	rec$_1$-Case-Split	**0.379**	0.344	0.344	0.343	0.327	**0.457**	0.368	0.367	0.367	0.367
	rec$_1$-Case-NBC	**0.497**	0.496	**0.497**	**0.497**	**0.497**	**0.499**	0.498	0.498	0.498	0.498
	Acc$_1$-Case-All	**0.958**	0.952	0.951	0.951	0.948	**0.972**	0.967	0.968	0.967	0.968
	Acc$_1$-Case-Split	**0.958**	0.951	0.951	0.951	0.948	**0.971**	0.968	0.968	0.968	0.968
	Acc$_1$-Case-NBC	**0.968**	0.967	**0.968**	0.967	**0.968**	0.937	**0.948**	**0.948**	**0.948**	**0.948**
	prec$_2$-Case-All	**0.519**	0.427	0.426	0.422	0.390	**0.211**	0.158	0.158	0.156	0.157
	prec$_2$-Case-Split	**0.519**	0.427	0.427	0.425	0.388	**0.208**	0.157	0.157	0.156	0.158
	prec$_2$-Case-NBC	**0.677**	0.669	0.671	0.667	0.670	0.098	0.119	0.120	0.119	**0.121**
	rec$_2$-Case-All	**0.759**	0.687	0.687	0.685	0.654	**0.916**	0.735	0.735	0.733	0.732
	rec$_2$-Case-Split	**0.759**	0.687	0.689	0.687	0.654	**0.914**	0.734	0.733	0.733	0.735
	rec$_2$-Case-NBC	**0.993**	**0.993**	**0.993**	**0.993**	**0.993**	**0.998**	0.996	0.996	0.996	0.995
	Acc$_2$-Case-All	**0.959**	0.947	0.946	0.945	0.940	**0.956**	0.948	0.948	0.947	0.948
	Acc$_2$-Case-Split	**0.959**	0.946	0.946	0.945	0.940	**0.955**	0.947	0.948	0.948	0.948
	Acc$_2$-Case-NBC	**0.979**	0.978	0.978	0.978	0.978	0.886	**0.909**	**0.909**	**0.909**	**0.909**

Table 4. Attacks with varying number of groups, *G*

		MLM					Netflix				
	G=	1	3	5	7	10	1	3	5	7	10
Acting as an active user attack	prec$_1$-Case-All	0.069	0.098	0.124	0.149	**0.175**	0.019	0.026	0.033	0.040	**0.047**
	prec$_1$-Case-Split	0.068	0.094	0.122	0.146	**0.171**	0.019	0.026	0.033	0.039	**0.047**
	prec$_1$-Case-NBC	0.157	0.220	0.281	0.335	**0.390**	0.045	0.064	0.084	0.103	**0.124**
	rec$_1$-Case-All	0.836	0.894	0.926	0.946	**0.960**	0.790	0.848	0.885	0.910	**0.928**
	rec$_1$-Case-Split	0.835	0.890	0.925	0.944	**0.959**	0.789	0.848	0.883	0.908	**0.929**
	rec$_1$-Case-NBC	0.887	0.922	0.941	0.954	**0.963**	0.852	0.890	0.915	0.930	**0.943**
	Acc$_1$-Case-All	0.506	0.634	0.713	0.763	**0.801**	0.489	0.604	0.675	0.724	**0.763**
	Acc$_1$-Case-Split	0.506	0.628	0.709	0.761	**0.797**	0.489	0.603	0.675	0.722	**0.765**
	Acc$_1$-Case-NBC	0.794	0.858	0.896	0.917	**0.934**	0.774	0.838	0.876	0.899	**0.917**
	prec$_2$-Case-All	0.076	0.102	0.127	0.151	**0.176**	0.022	0.028	0.035	0.041	**0.048**
	prec$_2$-Case-Split	0.075	0.099	0.125	0.149	**0.173**	0.022	0.028	0.035	0.041	**0.048**
	prec$_2$-Case-NBC	0.177	0.239	0.299	0.351	**0.405**	0.052	0.072	0.092	0.111	**0.132**
	rec$_2$-Case-All	0.919	0.936	0.949	0.959	**0.968**	0.916	0.923	0.932	0.941	**0.949**
	rec$_2$-Case-Split	0.919	0.934	0.949	0.959	**0.968**	0.916	0.923	0.931	0.941	**0.950**
	rec$_2$-Case-NBC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Acc$_2$-Case-All	0.509	0.636	0.714	0.763	**0.801**	0.491	0.605	0.676	0.725	**0.764**
	Acc$_2$-Case-Split	0.510	0.630	0.710	0.761	**0.797**	0.491	0.604	0.675	0.723	**0.765**
	Acc$_2$-Case-NBC	0.798	0.861	0.898	0.919	**0.935**	0.776	0.839	0.877	0.900	**0.918**
knn-based attack	prec$_2$-Case-All	**0.153**	0.152	**0.153**	**0.153**	**0.153**	**0.079**	0.075	0.075	0.075	0.075
	prec$_2$-Case-Split	0.181	**0.185**	0.181	0.183	0.182	**0.089**	0.088	0.087	0.087	0.087
	rec$_2$-Case-All	0.838	0.833	0.838	0.839	**0.842**	0.794	0.797	0.796	0.801	**0.804**
	rec$_2$-Case-Split	0.767	0.773	0.765	0.772	**0.776**	0.716	**0.721**	0.715	0.719	0.715
	Acc$_2$-Case-All	**0.795**	0.791	0.792	0.794	0.790	**0.884**	0.875	0.874	0.875	0.874
	Acc$_2$-Case-Split	**0.841**	**0.841**	0.840	0.839	0.839	**0.905**	0.904	0.903	0.903	0.902
Perfect match attack	prec$_1$-Case-All	0.426	0.721	0.897	0.963	**0.974**	0.158	**0.185**	0.164	0.136	0.111
	prec$_1$-Case-Split	0.427	0.721	0.897	0.961	**0.974**	0.157	**0.184**	0.161	0.134	0.110
	prec$_1$-Case-NBC	**0.672**	0.486	0.337	0.261	0.210	**0.120**	0.083	0.066	0.056	0.049
	rec$_1$-Case-All	0.343	0.443	0.485	0.498	**0.500**	0.367	0.456	0.492	0.499	**0.500**
	rec$_1$-Case-Split	0.344	0.444	0.485	0.497	**0.500**	0.367	0.455	0.492	**0.500**	**0.500**
	rec$_1$-Case-NBC	0.497	0.499	**0.501**	**0.501**	**0.501**	0.498	0.500	**0.501**	0.500	0.500
	Acc$_1$-Case-All	0.951	0.968	0.975	0.977	**0.978**	**0.968**	**0.968**	0.962	0.954	0.944
	Acc$_1$-Case-Split	0.951	0.968	0.975	0.977	**0.978**	**0.968**	**0.968**	0.962	0.954	0.944
	Acc$_1$-Case-NBC	**0.968**	0.955	0.935	0.916	0.896	**0.948**	0.925	0.905	0.888	0.873
	prec$_2$-Case-All	0.426	0.721	0.898	0.963	**0.974**	0.158	**0.185**	0.164	0.136	0.111
	prec$_2$-Case-Split	0.427	0.721	0.896	0.961	**0.974**	0.157	**0.184**	0.161	0.134	0.110
	prec$_2$-Case-NBC	**0.671**	0.486	0.336	0.261	0.210	**0.120**	0.083	0.066	0.056	0.049
	rec$_2$-Case-All	0.687	0.886	0.970	0.995	**1.000**	0.735	0.912	0.984	0.999	**1.000**
	rec$_2$-Case-Split	0.689	0.887	0.969	0.995	**1.000**	0.733	0.910	0.984	0.999	**1.000**
	rec$_2$-Case-NBC	0.993	0.999	**1.000**	**1.000**	**1.000**	0.996	**1.000**	**1.000**	**1.000**	**1.000**
	Acc$_2$-Case-All	0.946	0.980	0.994	0.998	**0.999**	0.948	**0.949**	0.938	0.921	0.901
	Acc$_2$-Case-Split	0.946	0.980	0.994	0.998	**0.999**	0.948	**0.949**	0.936	0.920	0.900
	Acc$_2$-Case-NBC	**0.978**	0.954	0.914	0.876	0.836	**0.909**	0.863	0.823	0.788	0.758

Table 5. Attacks with varying TPDs

		MLM					**Netflix**				
	TPD=	**0.125**	**0.250**	**0.500**	**0.750**	**0.875**	**0.125**	**0.250**	**0.500**	**0.750**	**0.875**
Acting as an active user attack	**prec$_1$-Case-All**	**0.187**	0.140	0.124	0.116	0.116	**0.043**	0.036	0.033	0.031	0.032
	prec$_1$-Case-Split	**0.150**	0.130	0.122	0.115	0.117	**0.041**	0.035	0.033	0.031	0.032
	prec$_1$-Case-NBC	**0.347**	0.291	0.281	0.268	0.266	**0.108**	0.089	0.084	0.079	0.079
	rec$_1$-Case-All	**0.963**	0.938	0.925	0.917	0.918	**0.918**	0.894	0.883	0.877	0.878
	rec$_1$-Case-Split	**0.946**	0.929	0.923	0.918	0.919	**0.914**	0.892	0.884	0.877	0.878
	rec$_1$-Case-NBC	**0.956**	0.945	0.941	0.938	0.937	**0.934**	0.921	0.915	0.910	0.910
	Acc$_1$-Case-All	**0.803**	0.743	0.712	0.692	0.693	**0.742**	0.697	0.673	0.661	0.665
	Acc$_1$-Case-Split	**0.763**	0.727	0.709	0.692	0.695	**0.733**	0.694	0.675	0.663	0.664
	Acc$_1$-Case-NBC	**0.921**	0.902	0.896	0.888	0.887	**0.903**	0.883	0.876	0.869	0.869
	prec$_2$-Case-All	**0.188**	0.142	0.127	0.119	0.119	**0.044**	0.038	0.035	0.033	0.034
	prec$_2$-Case-Split	**0.152**	0.133	0.125	0.119	0.120	**0.042**	0.037	0.035	0.033	0.033
	prec$_2$-Case-NBC	**0.362**	0.308	0.299	0.286	0.284	**0.115**	0.097	0.092	0.087	0.087
	rec$_2$-Case-All	**0.970**	0.956	0.949	0.944	0.946	**0.944**	0.937	0.931	0.929	0.930
	rec$_2$-Case-Split	**0.960**	0.952	0.948	0.945	0.946	**0.942**	0.936	0.931	0.929	0.930
	rec$_2$-Case-NBC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Acc$_2$-Case-All	**0.804**	0.744	0.713	0.693	0.694	**0.742**	0.698	0.673	0.661	0.665
	Acc$_2$-Case-Split	**0.764**	0.728	0.710	0.693	0.696	**0.734**	0.695	0.675	0.663	0.664
	Acc$_2$-Case-NBC	**0.923**	0.904	0.898	0.891	0.890	**0.904**	0.884	0.877	0.870	0.870
***knn*-based attack**	**prec$_2$-Case-All**	**0.178**	0.170	0.153	0.113	0.083	**0.098**	0.091	0.075	0.056	0.040
	prec$_2$-Case-Split	0.165	**0.200**	0.181	0.118	0.083	**0.121**	0.111	0.087	0.055	0.040
	rec$_2$-Case-All	**0.884**	0.863	0.838	0.828	0.830	**0.853**	0.829	0.796	0.800	0.800
	rec$_2$-Case-Split	0.535	0.707	**0.765**	0.742	0.754	**0.764**	0.740	0.715	0.719	0.731
	Acc$_2$-Case-All	**0.815**	0.811	0.792	0.708	0.596	**0.901**	0.896	0.874	0.828	0.756
	Acc$_2$-Case-Split	0.858	**0.863**	0.840	0.747	0.628	**0.927**	0.923	0.903	0.837	0.774
Perfect match Attack	**prec$_1$-Case-All**	**0.975**	0.887	0.897	0.892	0.886	0.129	0.144	0.164	**0.178**	0.173
	prec$_1$-Case-Split	**0.973**	0.886	0.897	0.893	0.884	0.120	0.142	0.161	**0.177**	0.168
	prec$_1$-Case-NBC	0.267	0.279	0.337	**0.375**	0.358	0.054	0.060	0.066	**0.069**	0.068
	rec$_1$-Case-All	**0.499**	0.484	0.485	0.484	0.483	**0.500**	0.491	0.492	0.492	0.491
	rec$_1$-Case-Split	**0.499**	0.484	0.485	0.484	0.483	**0.499**	0.491	0.492	0.492	0.490
	rec$_1$-Case-NBC	**0.502**	0.500	0.501	0.500	0.500	**0.501**	0.500	**0.501**	0.500	0.500
	Acc$_1$-Case-All	**0.978**	0.975	0.975	0.975	0.975	0.951	0.957	0.962	**0.965**	0.964
	Acc$_1$-Case-Split	**0.978**	0.975	0.975	0.975	0.975	0.949	0.956	0.962	**0.965**	0.963
	Acc$_1$-Case-NBC	0.917	0.923	0.935	**0.942**	0.939	0.884	0.897	0.905	**0.909**	**0.909**
	prec$_2$-Case-All	**0.975**	0.887	0.898	0.892	0.886	0.129	0.144	0.164	**0.178**	0.173
	prec$_2$-Case-Split	**0.973**	0.886	0.896	0.893	0.884	0.120	0.142	0.161	**0.178**	0.168
	prec$_2$-Case-NBC	0.266	0.279	0.336	**0.375**	0.358	0.054	0.061	0.066	**0.068**	**0.068**
	rec$_2$-Case-All	**0.999**	0.968	0.970	0.967	0.966	**1.000**	0.983	0.984	0.983	0.981
	rec$_2$-Case-Split	**0.998**	0.968	0.969	0.968	0.965	**1.000**	0.983	0.984	0.983	0.981
	rec$_2$-Case-NBC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Acc$_2$-Case-All	**0.999**	0.993	0.994	0.994	0.993	0.915	0.926	0.938	**0.943**	0.941
	Acc$_2$-Case-Split	**0.999**	0.993	0.994	0.994	0.993	0.911	0.925	0.936	**0.943**	0.939
	Acc$_2$-Case-NBC	0.877	0.889	0.914	**0.927**	0.922	0.780	0.807	0.823	**0.830**	**0.830**

Table 6. Random discovery results

		ML	Netflix
Random Discovery	**prec$_1$**	0.018	0.005
	rec$_1$	0.019	0.006
	Acc$_1$	0.916	0.975
	prec$_2$	0.042	0.012
	rec$_2$	0.045	0.013
	Acc$_2$	0.917	0.975

On the other hand, accuracy results seem to be higher than 0.90 for both aspects of privacy and data sets. The reason for such a case is that domination of unrated items and their domination is taken into account in accuracy calculations.

7. Conclusions and Future Works

In this study, three different attack techniques are experimentally tested on VPD-based binary PPCF schemes. *Acting as an active user* in multiple scenarios attack tracks temporal changes in the similarity weights for subsequent queries altered with only one cell. If a history of a user is known, which is an inherent case for VPD-based schemes, *knn-based* attack appends *k* fake users to the system to derive items by observing related predictions. *Perfect match* attack tracks similarity weights with 1 or -1 to derive meaningful information out of them.

Experiments show that *acting as active user* attack guarantees full recovery if no privacy measure is taken. However, it is highly affected by increasing δ values. The other control parameter G helps reconstruction for this attack type. Besides, the last control parameter TPD shows that this attack performs better for lower TPD values. Therefore, *acting as an active user* attack could be preferred for moderate δ values, larger G, and lower TPD values. On the other hand, *knn-based* attack demonstrates stabiltiy after δ is met. This attack performs similar for varying G and the best for lower TPD values. Similar to *acting as active user* attack, *knn-based* attack could be preferred if TPD is lower; nonetheless, *knn-based* attack can be preferred to *acting as an active user* attack when δ is large for the second aspect of privacy. In general, regardless of control parameters, the most prominent point for *perfect match* attack is that it yields much better precision results compared to the other two attacks for most of the cases. As stated before, this attack could be chosen if precision is considered. As a final remark, a *random* discovery has been performed and the attacks in this paper beat such a prior-knowledge discovery of a data holder's matrix especially in terms of precision and recall.

As a future goal, we plan to investigate multi-party horizontally and vertically distributed data-based privacy-preserving collaborative filtering schemes in terms of privacy by analyzing current attacks and devising possible attack techniques.

Acknowledgements

This work is supported by the Grant 113E262 from TUBITAK.

References

[1]. J. Jacoby, "Information Load and Decision Quality: Some Contested Issues," *J. Mark. Res.*, Vol. 14, No. 4, pp. 569-573, 1977.

[2]. J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, San Fransisco, CA, USA, 43-52, 1998.

[3]. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Inf. Syst.*, Vol. 22, No. 1, pp. 5-53, 2004.

[4]. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Commun. ACM*, Vol. 35, No. 12, pp. 61-70, 1992.

[5]. X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Adv. Artif. Intell.*, Vol. 2009, pp. 4:2-4:2, 2009.

[6]. A. Bilge, C. Kaleli, I. Yakut, I. Gunes, and H. Polat, "A Survey of Privacy-Preserving Collaborative Filtering Schemes," *Int. J. Softw. Eng. Knowl. Eng.*, Vol. 23, No. 08, pp. 1085-1108, 2013.

[7]. J. Canny, "Collaborative Filtering with Privacy", in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 45-57, 2002.

[8]. L. F. Cranor, "'I Didn't Buy it for Myself'", in *Proceedings of the ACM Workshop on Privacy in the*

Electronic Society, Washington, DC, USA, 111-117, 2003.

[9]. J. Phelps, G. Nowak, and E. Ferrell, "Privacy Concerns and Consumer Willingness to Provide Personal Information," *J. Public Policy Mark.*, Vol. 19, No. 1, pp. pp. 27-41, 2000.

[10]. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the Privacy Preserving Properties of Random Data Perturbation Techniques," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, Melbourne, FL, USA, 99-106, 2003.

[11]. H. Dutta, H. Kargupta, S. Datta, and K. Sivakumar, "Analysis of Privacy Preserving Random Perturbation Techniques: Further Explorations," in *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, Washington, DC, USA, 31-38, 2003.

[12]. S. Zhang, J. Ford, and F. Makedon, "Deriving Private Information from Randomly Perturbed Ratings," in *Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, MD, USA, 59-69, 2006.

[13]. J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, "``You Might Also Like:'' Privacy Risks of Collaborative Filtering," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 231-246, 2011.

[14]. M. Okkalioglu, M. Koc, and H. Polat, "On the Discovery of Fake Binary Ratings," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, Salamanca, Spain, 901-907, 2015.

[15]. H. Polat and W. Du, "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, Melbourne, FL, USA, 625-628, 2003.

[16]. H. Polat and W. Du, "Achieving Private Recommendations Using Randomized Response Techniques," *Lect. Notes Comput. Sci.*, Vol. 3918, pp. 637-646, 2006.

[17]. S. L. Warner, "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias," *J. Am. Stat. Assoc.*, Vol. 60, No. 309, pp. 63-69, 1965.

[18]. M. Okkalioglu, M. Koc, and H. Polat, "On the Privacy of Horizontally Partitioned Binary Data-based Privacy-Preserving Collaborative Filtering," *Lect. Notes Comput. Sci.*, Vol. 9481, 2015.

[19]. H. Polat and W. Du, "Privacy-Preserving top-*N* Recommendation on Distributed Data," *J. Am. Soc. Inf. Sci. Technol.*, Vol. 59, No. 7, pp. 1093–1108, 2008.

[20]. C. Kaleli and H. Polat, "Providing Naïve Bayesian Classifier-based Private Recommendations on Partitioned Data," *Lect. Notes Comput. Sci.*, Vol. 4702, pp. 515-522, 2007.

[21]. C. Kaleli and H. Polat, "Providing Private Recommendations Using Naïve Bayesian Classifier," *Advances in Soft Computing*, Vol. 43, pp. 168-173, 2007.

[22]. M. Tada, H. Kikuchi, and S. Puntheeranurak, "Privacy-Preserving Collaborative Filtering Protocol Based on Similarity between Items," *in Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, Australia, 573-578, 2010.

[23]. H. Polat and W. Du, "Privacy-Preserving top-*N* Recommendation on Horizontally Partitioned Data," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Paris, France, 725-731, 2005.

[24]. H. Polat and W. Du, "Privacy-Preserving Collaborative Filtering on Vertically Partitioned Data" *Lect. Notes Comput. Sci.*, Vol. 3721, pp. 651-658, 2005.

[25]. C. Kaleli and H. Polat, "Privacy-Preserving Naïve Bayesian Classifier‐Based Recommendations on Distributed Data," *Comput. Intell.*, Vol. 31, No. 1, pp. 47-68, 2015.

[26]. C. Kaleli and H. Polat, "Privacy-Preserving SOM-based Recommendations on Horizontally Distributed Data," *Knowledge-Based Syst.*, Vol. 33, pp. 124-135, 2012.

[27]. C. Kaleli and H. Polat, "SOM-based Recommendations with Privacy on Multi-party Vertically Distributed Data," *Journal of the Operational Research Society*, Vol. 63, No. 6, pp. 826-838, 2012.

[28]. S. Guo and X. Wu, "On the Use of Spectral Filtering for Privacy Preserving Data Mining," in *Proceedings of the ACM Symposium on Applied Computing*, Dijon, France, 622-626, 2006.

[29]. S. Guo, X. Wu, and Y. Li, "Determining Error Bounds for Spectral Filtering based Reconstruction Methods in Privacy Preserving Data Mining," *Knowl. Inf. Syst.*, Vol. 17, No. 2, pp. 217-240, 2008.

[30]. Z. Huang, W. Du, and B. Chen, "Deriving Private Information from Randomized Data," in *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data*, Baltimore, MD, USA37-48, 2005.

[31]. M. Okkalioglu, M. Koc, and H. Polat, "Deriving Private Data in Vertically Partitioned Data-based PPCF Schemes," in *Proceedings of the 9th International Conference on Information Security and Cryptology*, Ankara, Turkey, 1-7, 2015.

[32]. K. Miyahara and M. Pazzani, "Collaborative Filtering with the Simple Bayesian Classifier," *Lect. Notes Comput. Sci.*, Vol. 1886, pp. 679-689, 2000.