University of Texas Rio Grande Valley

ScholarWorks @ UTRGV

Computer Science Faculty Publications and Presentations

College of Engineering and Computer Science

10-2020

The Majority Rule: A General Protection on Recommender System

Lei Xu
The University of Texas Rio Grande Valley, lei.xu@utrgv.edu

Lin Chen

Martin Flores
The University of Texas Rio Grande Valley

Hansheng Lei
The University of Texas Rio Grande Valley

Liyu Zhang
The University of Texas Rio Grande Valley, liyu.zhang@utrgv.edu

See next page for additional authors

Follow this and additional works at: https://scholarworks.utrgv.edu/cs_fac

 Part of the Computer Sciences Commons

Recommended Citation

Lei Xu, Lin Chen, Martin Flores, Hansheng Lei, Liyu Zhang, Mahmoud K. Quweider, Fitratullah Khan, and Weidong Shi. 2020. The Majority Rule: A General Protection on Recommender System. In Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence (SPAI '20). Association for Computing Machinery, New York, NY, USA, 40–46. https://doi.org/10.1145/3385003.3410923

This Conference Proceeding is brought to you for free and open access by the College of Engineering and Computer Science at ScholarWorks @ UTRGV. It has been accepted for inclusion in Computer Science Faculty Publications and Presentations by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact justin.white@utrgv.edu, william.flores01@utrgv.edu.

Authors

Lei Xu, Lin Chen, Martin Flores, Hansheng Lei, Liyu Zhang, Mahmoud K. Quweider, Fitratullah Khan, and Weidong Shi

This conference proceeding is available at ScholarWorks @ UTRGV: https://scholarworks.utrgv.edu/cs_fac/38

The Majority Rule: A General Protection on Recommender System

Lei Xu
xuleimath@gmail.com
University of Texas Rio Grande Valley
Brownsville, TX

Lin Chen
chenlin198662@gmail.com
Texas Tech University
Lubbock, TX

Martin Flores
martin.flores05@utrgv.edu
University of Texas Rio Grande Valley
Brownsville, TX

Hangsheng Lei
hansheng.lei@utrgv.edu
University of Texas Rio Grande Valley
Brownsville, TX

Liyu Zhang
liyu.zhang@utrgv.edu
University of Texas Rio Grande Valley
Brownsville, TX

Mahmoud K. Quweider
mahmoud.quweider@utrgv.edu
University of Texas Rio Grande Valley
Brownsville, TX

Fitratullah Khan
fitra.khan@utrgv.edu
University of Texas Rio Grande Valley
Brownsville, TX

Weidong Shi
wshi3@central.uh.edu
University of Houston
Houston, TX

ABSTRACT

Recommender systems are widely used in a variety of scenarios, including online shopping, social network, and contents distribution. As users rely more on recommender systems for information retrieval, they also become attractive targets for cyber-attacks. The high-level idea of attacking a recommender system is straightforward. An adversary selects a strategy to inject manipulated data into the database of the recommender system to influence the recommendation results, which is also known as a profile injection attack. Most existing works treat attacking and protection in a static manner, i.e., they only consider the adversary's behavior when analyzing the influence without considering normal users' activities. However, most recommender systems have a large number of normal users who also add data to the database, the effects of which are largely ignored when considering the protection of a recommender system. We take normal users' contributions into consideration and analyze popular attacks against a recommender system. We also propose a general protection framework under this dynamic setting.

CCS CONCEPTS

• **Security and privacy**; • **Information systems → Recommender systems**;

KEYWORDS

Recommender system, protection, VRF

1 INTRODUCTION

The rate at which information is being created is still accelerating, and people are relying on machines for information management and process more than ever. Such applications include automatic recommendation, website searching, and a variety of machine learning applications.

A recommender system [5] provides an effective way to help a user to navigate through the mass data. Unlike a search engine, a recommender system works in an active way, i.e., it pushes information to the end user actively instead of waiting for the user's instruction to retrieve information. A typical recommendation process consists of two steps: prediction and ranking. For prediction, the recommender system utilizes existing records (e.g., user's profile and existing reviews on items) to generate scores on items that a user does not have an existing review. The system then ranks these items using predicted scores and pushes those with high scores to the user.

Because the scores are generated based on users' profile and items with higher scores are recommended to the user first, the probability that the user likes them and purchases/reads/listens is high.

Automatic recommendation has been widely used in e-commerce systems [15], news distribution [9], and music/movie suggestions [2, 18]. Already, 35 percent of what consumers purchase on Amazon and 75 percent of what they watch on Netflix come from product recommendations based on such algorithms [11].

Since the recommendation results have a close connection with one's financial interests (e.g., increasing an item's exposure to potential customers), the security of the recommender system becomes

a critical issue. Besides utilizing the general attacking techniques against the servers that host the recommender system to directly control the recommendation results, an adversary can also influence the predictions by adding new reviews [12, 14, 17]. The second type of attack is also known as profile injection and does not require sophisticated tools to invade into the target system, which are usually easier to carry out. The non-invasive character of such attacks also makes it harder to be detected. Two strategies are proposed to mitigate the second type of attack:

- Anomaly detection. If the adversary's behavior is different from normal users, an anomaly detection algorithm can be deployed in front of the recommender system to filter out new records that are created by the adversary.
- Making the recommender system as a black-box. An adversary usually relies on knowledge of the recommendation algorithm and existing review records to construct new reviews to maximize the attack impacts. Therefore, preventing the adversary from learning the information can mitigate the attack.

Both strategies have some limitations. For example, all anomaly detection algorithms have a false positive rate, i.e., normal users may be misclassified as adversaries and their reviews will be rejected, which affects user experience and reduces the performance of the recommender system. Making the recommender system as a black-box does not have the misclassification issue but information (e.g., existing review values on items) is hidden from both adversaries and normal users to increase the hardness for the adversary. This also affects the user experience and is not desirable.

In this paper, a novel approach of recommender system protection that eliminates the above limitations is developed. The new solution takes normal users' behavior into consideration when studying the attack impacts and our analyses demonstrate that those reviews submitted by normal users can counteract the influence of the adversary, especially when only a small percentage of reviews are produced by the adversary. We also design a mechanism to enforce the limitation and prevent the adversary from submitting a large number of reviews to dominate the system. Specifically, the mechanism leverages cryptography primitives to link cyber world activities (submitting new reviews) to physical world activities (computation), e.g., to submit a review, one has to finish some computation task. Therefore, as long as the adversaries are only a small fraction of all users, they cannot submit an overwhelming amount of reviews.

In summary, our contributions in this works include:

- We prove and demonstrate that normal users' activities can mitigate the adversary's attack against a recommender system, especially when the adversary only contribute a small percentage of reviews;
- We design a novel mechanism to prevent the adversary from submitting a large number of reviews to manipulate the recommendation results regardless of the strategy the adversary adopts. The new mechanism does not rely on any assumption on anomaly detection, so it does not suffer from issues like false positives.

2 THREATS AGAINST RECOMMENDER SYSTEMS

This section presents the background of recommender systems and typical attacking methods against a recommender system.

2.1 Recommender System

A typical recommender system involves two types of subjects, *user*s and *item*s. A user can make *review*s for an item. The information can be organized as a matrix (rating matrix), where each row represents a user and each column represents an item. A user's review of an item is a value of the matrix at the corresponding place. Given a user, the recommender system provides an ordered recommendation list of items that the user may be interested in.

The recommendation process can be further divided into two phases, prediction and recommendation/ranking. During the prediction phase, the system determines how much a user may like an item that he/she does not have a review. For the recommendation/ranking step, the system determines which items should be pushed to the user. The second step is usually done by simply ranking the predictions and selecting the tops to recommend to the user. We only focus on the attack and protection of the prediction step in this paper.

The nature of prediction is guessing a user's attitude on an item that he/she does not have a review. The prediction is usually computed based on the rating matrix, i.e., existing reviews and users' profiles. For example, if two users share a similar profile, then it is more likely that they have the same preference. And the recommender system can use one user's review for the other user as a prediction. There are a variety of strategies to utilize existing information to calculate the prediction, such as collaborative filtering [5], rating matrix factorization [6], and contents based recommendation [10].

2.2 Attacks against a Recommender System

No matter how an attack is launched against the target recommender system, the adversary usually aims to achieve one of the purposes, *item push* and *item nuke* [14]. The goal of item push attack is to promote the predications on targeted items, while the item nuke attack tries to demote the predictions on selected items.

The adversary can attack the servers running the recommender system directly to take over using various hacking techniques. The adversary can then arbitrarily manipulate the recommendation computation to push or nuke items. General system security enhancement methods such as installing patches frequently and running intrusion detection can help to prevent such attacks.

Another type of attack is less invasive. Since the prediction is determined by the rating matrix where the items are reviews, the adversary can influence prediction results by adding selected reviews to achieve push or nuke purpose. For different recommendation algorithms, the adversary needs to use different strategies to build reviews to maximize the attack impacts.

The second type of attack poses a more serious threat to a recommender system as it is easier to launch such attacks, and hard to detect if the adversary is careful enough. We only consider the second type of attack in this work.

3 PROBLEM STATEMENT

We consider a recommender system $\mathcal{S}$ with n users $(u_1, u_2, \cdots, u_n)$, and use $v_{u_i,y}$ to denote the review value that user u_i given to item y. The predicated review value for user u_i on item y is calculated using Pearson's correlation based collaborative filtering as follows:

$$v_{u_i,y} = \hat{v}_i + \frac{\sum_j C_{i,j}(v_{u_j,y} - \hat{v}_j)}{\sum_j |C_{i,j}|}, \tag{1}$$

where $\hat{v}_i$ and $\hat{v}_j$ are the average review values of user u_i and u_j respectively.

The Pearson's correlation coefficient is a measure of similarity between two users based on their reviews on common items, and calculated as follows [5]:

$$C_{i,j} = \frac{\sum_y (v_{u_i,y} - \hat{v}_i)(v_{u_j,y} - \hat{v}_j)}{\sqrt{\sum_y (v_{u_i,y} - \hat{v}_i)^2}\sqrt{\sum_y (v_{u_j,y} - \hat{v}_j)^2}} \tag{2}$$

The value of $C_{i,j}$ lies in $[-1, 1]$. A value 1 implies that a linear equation with positive slop describes the relationship between the two users' reviews, and -1 implies a linear equation with negative slop describes the relationship.

After an adversary finishes the attack (i.e., adding new reviews to the system which are constructed according to his/her purpose), we use *prediction shift* to describe the effectiveness of the attack, which is difference defined as:

$$\mathcal{P}_y = \sum_u (\dot{v}_{u_i,y} - v_{u_i,y}), \tag{3}$$

where $\dot{v}_{u_i,y}$ is the predicted review value of item y for user u_i after the attack. Since $v_{u_i,y}$ is a constant value before the attack, the adversary only needs to maximize or minimize the term $\dot{v}_{u_i,y}$ for a push or nuke attack.

This definition can be easily extended to the situation of multiple adversaries. In the worst situation, these adversaries collaborate to maximize the effectiveness of the attack. However, the prediction shift defined in Equation (3) only considers the adversaries but ignores the activities of normal users, which also contribute to the database for the recommender system. In this work, we focus on the following two problems:

- When both adversaries and normal users add new reviews to the database of a recommender system, whether the effectiveness of the attack will be changed or not. It is desirable to have a quantitative analysis of the relationship between the impact of the attack and the ratio of adversaries from all users.
- If adversaries can generate an unlimited amount of review values, they can manipulate the prediction in an arbitrary way. It is desirable to design a mechanism to prevent an adversary from submitting a huge amount of reviews.

4 ANALYSIS OF THE IMPACTS OF NORMAL USERS

In this section, we analyze the impacts of normal users' behavior on the prediction shift when the recommender system is under attack.

We divide time into discrete epochs when analyzing the safety of the recommender system. At the beginning of each epoch, both normal users and adversaries can add new reviews to the system. The recommender system generates new predictions at the end of the epoch using the updated database. The adversary cannot control or predict the behavior of normal users. In the following analysis, we assume the adversary adopts the same strategy to generate new reviews to push or nuke the target items without considering normal users' contribution to the database of the recommender system.

4.1 Case 1: Inactive Target User

We divide all users of the system into three groups: S_1 is the set of collaborative adversaries who intentionally generate reviews to manipulate the prediction, S_2 is the set of normal users who generate new reviews in the current epoch, and S_3 is the set of normal users who do not generate new reviews. We first consider the situation that the target user $u_i \in S_3$, i.e., u_i does not generate new reviews during the current epoch.

In this case, the updated prediction value for user u_i on item y at the end of the epoch is given as follows:

$$\tilde{v}_{u_i,x} = \hat{v}_i + \frac{X_1 + X_2 + X_3}{\sum_{j\in S_1} |C_{i,j}| + \sum_{j\in S_2} |C_{i,j}| + \sum_{j\in S_3} |C_{i,j}|} \tag{4}$$

Here $X_1 = \sum_{j\in S_1} C_{i,j}(v_{u_j,x} - \hat{v}_j)$, $X_2 = \sum_{j\in S_2} C_{i,j}(v_{u_j,x} - \hat{v}_j)$, and $X_3 = \sum_{j\in S_1} C_{i,j}(v_{u_j,x} - \hat{v}_j)$. Since users in S_3 do not add new reviews, $\hat{v}_i, X_3$ and $\sum_{j\in S_3} |C_{i,j}|$ remain the same during the epoch. Furthermore, adversaries launch the attack without considering new reviews created by normal users and the user u_i does not change his/her profile during the epoch, so the values of X_1 and $\sum_{j\in S_1} |C_{i,j}|$ are also the same during the epoch.

In summary, the only two terms that affect the effectiveness of an attack within an epoch are X_2 and $\sum_{j\in S_2} |C_{i,j}|$, which are related to normal users' new reviews. Without loss of generality, we consider the scenario where the goal of the adversary is to push an item x for user u_i, the nuke scenario can be analyzed in the same way.

Let $c_1 \leftarrow X_1 + X_3$ and $c_2 \leftarrow \sum_{j\in S_1} |C_{i,j}| + \sum_{j\in S_3} |C_{i,j}|$. To achieve the goal of push, the adversary tries to maximize the prediction value. Therefore, the attack effect is weakened if the new prediction $\tilde{v}_{u_i,x}$ given in Equation (4) drops when the normal users' activities are taken into consideration. Specifically, the attack effect is reduced when

$$\begin{aligned}
\hat{v}_i + \frac{c_1}{c_2} > \hat{v}_i + \frac{c_1 + X_2}{c_2 + \sum_{j\in S_2} |C_{i,j}|} &\Leftrightarrow \\
\frac{c_1}{c_2} > \frac{c_1 + X_2}{c_2 + \sum_{j\in S_2} |C_{i,j}|} &\Leftrightarrow \\
c_1 \cdot (c_2 + \sum_{j\in S_2} |C_{i,j}|) > c_2 \cdot (c_1 + X_2) &\Leftrightarrow \\
\frac{c_1}{c_2}(\sum_{j\in S_2} |C_{i,j}|) > X_2 &
\end{aligned} \tag{5}$$

For a simplified case where $\{u_j\} = S_2$, the Pearson correlation coefficient between u_j and the target user u_i is

$$\begin{aligned}
C_{i,j} &= \frac{\sum_y (v_{u_i,y} - \hat{v}_i)(v_{u_j,y} - \hat{v}_j)}{\sqrt{\sum_y (v_{u_i,y} - \hat{v}_i)^2}\sqrt{\sum_y (v_{u_j,y} - \hat{v}_j)^2}} \\
&= \frac{\sum_y c_y (v_{u_j,y} - \hat{v}_j)}{\sqrt{\sum_y c_y^2}\sqrt{\sum_y (v_{u_j,y} - \hat{v}_j)^2}},
\end{aligned} \tag{6}$$

where c_y is a constant value, and X_2 is calculated as

$$X_2 = C_{i,j}(v_{u_j,x} - \hat{v}_j) = \frac{(v_{u_j,x} - \hat{v}_j)(\sum_y c_y(v_{u_j,y} - \hat{v}_j))}{\sqrt{\sum_y c_y^2}\sqrt{\sum_y (v_{u_j,y} - \hat{v}_j)^2}} \quad (7)$$

By plugging Equations (6) and (7) into Equation (5), we have

$$\frac{c_1}{c_2} > (-1)^t (v_{u_j,x} - \hat{v}_j). \quad (8)$$

The value of $t \in \{1, 2\}$ is determined by Equation (6). Depending on the review(s) newly created by u_j, Equation (8) can be either true or false. For instance, if the user u_j has a low review value on item x and generates a high review value on another item in the current epoch, it is very likely that $v_{u_j,x} - \hat{v}_j < 0$. In this case, if we also have $c_1 > 0$ and $t = 2$, Equation (8) holds, and the normal user's activity does not weaken the adversary's attack. On the other hand, if u_j has a high review value on item x and generates a new review with a low value for another item, then the probability that $v_{u_j,x} - \hat{v}_j > 0$ becomes high. Equation (8) does not hold in this case, which implies that the attack effect is offset by u_j's new activity.

4.2 Case 2: The Active Target User

The above analysis assumes the target user u_i does not generate new reviews during this epoch, and the situation becomes more complex when u_i is also active. In this case, all terms in Equation (4) can be affected by u_i's new review. To evaluate the impacts on the attack, we consider the difference between the two predictions, i.e., the prediction value that the attack is not affected by u_i's activity and the prediction value that the attack is affected by u_i's new review.

If u_i generates a new review on exactly the item that the adversary wants to push/nuke, the situation becomes trivial as the adversary's activity becomes irrelevant in this case. In the following, we assume the adversary aims at pushing an item x and u_i generates a new review on an item y' that is different from x. The difference is calculated as:

$$(\hat{v}_i + \frac{\sum_j C_{i,j}(v_{u_j,y} - \hat{v}_j)}{\sum_j |C_{i,j}|}) - (\hat{v}'_i + \frac{\sum_j C'_{i,j}(v_{u_j,y} - \hat{v}_j)}{\sum_j |C'_{i,j}|}) \quad (9)$$

Here $C_{i,j}$ and $\hat{v}_i$ are Pearson correlation coefficient and average review value where u_i is inactive, and $C'_{i,j}$ and $\hat{v}'_i$ represent the values where u_i is active.

The newly added review on y' affects both u_i's average review and the correlation values. If user u_j also has a review on y', two new terms are introduced to corresponding $C'_{i,j}$, i.e., $(v_{u_i,y'} - \hat{v}'_i)$ and $(v_{u_j,y'} - \hat{v}_j)$. According to the calculation given in Equation (2), $C'_{i,j}$ can be either larger or smaller than $C_{i,j}$ based on the value of the new view on y'. If y' is the only overlapped item that both u_i and u_j reviewed, it creates a new correlation coefficient, which can be any value in $[-1, 1]$.

In summary, whether Equation (9) is positive or negative depends on the newly generated review(s). If the adversary's goal is to push the item x, a negative value of Equation (9) means the attack effect is weakened. Similarly, a positive Equation (9) weakens a nuke attack. In certain cases, the newly added reviews may also enhance the attack impacts. But this is more like to happen when the adversary wants to push a popular item or nuke an unpopular one, which would not inflict much damage.

4.3 Impacts of Normal Users' Activities for a General Recommender System

The above analyses consider a specific recommender system that is built using Pearson correlation coefficients for prediction calculation. However, the same core concept is also applicable to other recommendation schemes. Under our security model, an adversary can only manipulate the prediction values in the recommendation process by adding carefully designed reviews. Based on the specific prediction algorithm, the adversary can adopt a strategy on adding new reviews to maximize the push/nuke effectiveness. For a given recommender system, no matter what the underlying prediction algorithm is, there are always reviews that can increase a prediction and those can reduce a prediction. This must be true otherwise the prediction values become monotonic, i.e., despite the users' feedback, an item's review only increases or decreases.

Generally, people have different opinions, and their reviews on the same item may contradict with each other. In this case, their impacts on the final prediction of such an item for another user may counteract. There are two possibilities when considering the impacts of normal users' behavior on an attack:

- Case 2a. Normal users do not have a consensus on the item and the reviews are polarized. In this case, it is relatively easy for an adversary to manipulate the prediction in either way (e.g., push or nuke). This is because reviews from normal users with different opinions cancel each other's impacts and the adversary can work in a neutral environment without any "noise", which is to some extent equivalent to the situation that the adversary is the only one who adds new reviews to influence the prediction.
- Case 2b. Normal users have a general agreement on the item. It is harder for an adversary to manipulate the prediction in this case, especially when he/she wants to push an item that is unpopular or nuke an item that is popular among normal users. The adversary's activity may still affect the prediction (e.g., hindering the process of an item getting more popular or unpopular) as his/her reviews will be taken into consideration anyway when calculating the prediction.

Based on the nature of the items that the recommender system manages, the likelihood of case 2a and case 2b can be different. For examples, polarization of opinions (case 2a) is relatively common on movies/books/music [1]. But for other types of items where there are common objective criteria, case 2b is more common.

5 RESTRICTING ADVERSARIES' CAPABILITY

In this section, we describe the solution to prevent an adversary from creating a large number of reviews to affect the predictions in the recommendation.

Most existing works use anomaly detection to prevent an adversary from generating too many reviews. Approaches along with this direction assume that an adversary behaves differently than a normal user when he/she tries to maximize the influence to manipulate the prediction. While this type of mechanisms are effective for certain adversaries, it is not easy to have a general anomaly

detection method that works for all attack strategies. Furthermore, if the adversary launches a Sybil attack [8] to create a large number of users in the system, and each malicious user behaves exactly as a normal user, there is no way for an anomaly detection scheme to distinguish them. To overcome these challenges, we develop a new mechanism that utilizes VDF, verifiable delay function [4] to restrict the adversary's capability on creating new reviews.

5.1 Verifiable Delay Function (VDF)

As the name suggested, a VDF is a function that requires a specified number of sequential steps to calculate, and produces a unique output that can be easily verified [4]. Based on the requirement on the computation complexity, one can initialize a VDF instance by providing a delay parameter, which reflects the number of sequential steps required to evaluate the function. More specifically, a VDF scheme has three algorithms: **setup**, **eval**, and **vrfy**. **setup** takes a security parameter λ and a delay parameter t as inputs, and outputs public parameters pp. $\textbf{eval}(pp, x) \rightarrow (\pi, y)$ represents the VDF calculation process and the output is a pair (π, y), where π is a short proof to verify the computation. $\textbf{vrfy}(pp, x, y, \pi)$ efficiently verifies that y is the correct output on x. Informally, a secure VDF satisfies three requirements:

- One-to-one mapping. For every input x, there is a unique y that can pass the verification.
- Sequential. An honest party can calculate $\textbf{eval}(pp, x)$ in t sequential steps, and no adversary equipped with a parallel computer that has a polynomial number of processors can distinguish the output y from a random value in significantly fewer steps.
- Efficiently verifiable. One with necessary information can compute the function **vrfy** in a more efficient way compared with **eval**, e.g., $O(polylog(t))$.

The above features remain valid even if an attacker can perform a polynomial bounded amount of pre-computation.

VDF provides a mechanism to control the waiting time of an entity with limited computation resources in the cyber world and we leverage it to set the limit of the adversary to prevent he/she from flooding the system with fake reviews.

5.2 Integration of VDF with Recommender System

A recommender system works in a client-server model. While a client can be malicious, we always assume the server is trusted, who keeps user/item/review information and runs the recommendation algorithm. The protocol of integrating a VDF with a recommender system consists of the following algorithms:

- `Initialization`. This algorithm is executed by the server. The server selects a VDF scheme, decides the desired computation steps and other auxiliary information. It then runs the algorithm **setup** to initialize the VDF, which outputs public parameters for the other two functions **eval** and **vrfy**. **setup** may require secure setup, i.e., secret information is involved in the generation of public parameters and one can compromise the security features of the VDF with such information. This is not a problem for our scenario as we assume the server is trusted, and only the server needs to know the secret information to carry out the **setup** function.
- `Submission`. This algorithm is executed by a client, which is a user of the recommender system. When a user u (either a normal user or an adversary) submits a review x to the system, u also runs **eval** on the review to obtain an output (π, y). In order to avoid the case that the adversary reuses the computation result and submits the same review multiple times under different pseudonyms, `Submission` requires the review x to be pre-processed before feeding into **eval**. A simple pre-processing method is to apply a cryptography hash function to the concatenation of the review and the identity to obtain the input for **eval**, i.e., the user needs to compute and send $(\textbf{eval}(\mathsf{hash}(x||id_u)), x, id_u)$ to the system.
- `Verification`. After receiving a new review together with the VDF result from a user, the system runs `Verification` to check whether the review should be accepted. The algorithm if further divided into two steps: the first step is computing the hash function, and the second step uses **vrfy** to determine whether the received result is valid. If it passes this verification, the corresponding review is added to the server's database, which stores all inputs to the recommender.
- `Recommendation`. This algorithm is run by the server, which is the same as the original recommendation algorithm and includes two functions, i.e., prediction and ranking. Note that only reviews with valid VDF evaluation results will be used in the prediction computing.

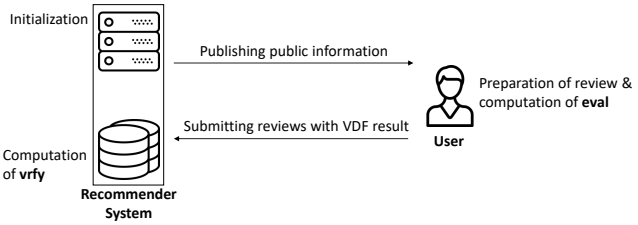

Figure 1: Preventing an adversary from submitting an unlimited number of reviews with VDF. Every time a user submits a review to influence the recommendation result, he/she also needs to finish a VDF evaluation. Note that the protocol is independent of the underlying VDF and can work with different VDF schemes.

Figure 1 demonstrates the way VDF is used to restrict the submission of reviews.

5.3 Analysis on the Recommender System with VDF

We analyze the recommender system with VDF from two perspectives, security and performance.

Security. The recommender system with VDF described above does not need to distinguish adversaries from normal users, and every new review is processed in the same way. Furthermore, this scheme establishes a connection between the computation resources a user has and the number of new reviews he/she can submit, which is similar to the case of Bitcoin [13]. We argue that the percentage of

reviews a user can submit is proportional to his/her computation power.

We first demonstrate that there is no way for an adversary to save any computation efforts to submit a new review to the system if the underlying VDF is secure. According to the protocol given in the previous section, an adversary cannot use replay attacks directly to save computation effort, as the same review from the same identity will only be accepted once. If the adversary can reduce the computation cost by using all of his/her previous reviews and carefully selected pseudonyms, such an adversary can be used to build a distinguisher $\mathcal{D}$ to compromise the sequential feature of the VDF because $\mathcal{D}$ can call this adversary to evaluate the constructed input to verify whether the output is a random value. Intuitively, the protocol enforces a one-to-one mapping between a review and a VDF evaluation.

We then argue that the percentage of attacking reviews is proportional to the percentage of computation power that adversaries control. The one-to-one mapping between reviews and VDF evaluations guarantees that the adversary cannot gain any benefits by creating a large pool of pseudonyms, and multiple adversaries collude with each other cannot gain extra benefits. Therefore, as long as the normal users are the majority, normal reviews are also the majority, which can greatly mitigate the consequences caused by those malicious reviews created by adversaries.

Performance. On the user side, an extra hash function and a VDF evaluation are required. Although the VDF evaluation can be expensive, it does not affect a normal user much as in most cases he/she only submits a limited number of reviews.

On the server side, an extra hash function and a VDF verification are required. Compared with VDF evaluation ($O(t)$), VDF verification is much cheaper ($O(polylog(t))$). Therefore, the new scheme only introduces a limited amount of extra work for the server and does not affect the performance much. However, this increases the susceptibility of the scheme to denial-of-service attacks. Specifically, adversaries can submit reviews without conducting VDF evaluation, but using randomly generated values. The server then needs to respond by executing the verification algorithm. Though the server can detect and reject these reviews, they cause extra computation on the server side at a low cost.

6 RELATED WORKS

In this section, we briefly review related works on recommender system attack and protection.

6.1 Attacking a Recommender System

While an adversary can attack a recommender system as a general IT infrastructure, we focus on profile injections, i.e., noninvasive attacks that try to manipulate the recommendation by adding carefully designed reviews. Lam and Riedl explored four open questions that will affect the effectiveness of profile injection (shilling attack) [7]. Metha proposed a strategy for profile injection to maximize correlation with maximum users [12]. Yang et.al. converted the profile injection strategy to a constrained linear optimization problem for co-visitation recommender systems [17]. All their analyses are based on the static model, i.e., normal users do not submit any new reviews to change the landscape while the adversary is attacking the recommender system.

6.2 Protecting a Recommender System

Various techniques have been proposed to protect a recommender system, and there are two main strategies. The first strategy is detecting attack activities. If the adversary behaves differently than normal users, an anomaly detection algorithm may be able to catch the adversary and reverse his/her newly added reviews. Bhaumik et.al. proposed a method of detecting suspicious rating trends based on statistical anomaly detection [3]. A classification approach was developed in [16] to detect profile injection, which identifies a number of attributes that can distinguish attack profiles in general. This type of approaches suffer from two limitations: (i) False positive instances. All anomaly detection mechanisms have false positives, and it will affect user experience if a normal user is classified as an adversary. (ii) Evolved adversary. The adversary can change his/her behavior accordingly after learning the detection methods to avoid being detected.

Another strategy is increasing the hardness of mounting an effective attack. In order to design efficient reviews to prepare the attack, the adversary usually needs to gain a good understanding of the existing database used to compute the recommendation. The major limitations of this type of protection mechanism are: (i) Reducing the usability of the system. The recommender system becomes less useful when it hides too much information from the user. (ii) Non-adaptable. Such a protection mechanism is usually specific to the recommendation algorithm, and hard to be generalized to new recommender systems.

7 CONCLUSION

Recommender systems are used in a wide range of applications including product recommendation in e-commerce, automatic news pushing, and customized music list generation. Due to its importance, recommender systems have become an attractive target for cyber-attacks. Unlike those attacks utilizing software vulnerabilities and other techniques to invade the system, an adversary can easily exploit a recommender system by profile injection, i.e., inserting carefully designed reviews into the system to manipulate the recommendation results, especially when he/she has a good understanding of the target system. This type of attack creates a challenge for designing a general purpose protection mechanism. In this paper, we demonstrated that normal users' activities can counteract the effectiveness of a profile injection attack regardless of the design of the recommender system. The effectiveness of this mitigation depends on the percentage of reviews that the adversary creates, and we propose a novel protocol to limit the review submission rate. This limitation does not affect normal users as they do not have the motivation to submit a large number of reviews in a short time period.

For future work, we plan to investigate more details on the relationship between normal users' activities and the adversary's attacks and conduct experiments to verify the theory.

REFERENCES

[1] Luca Amendola, Valerio Marra, and Miguel Quartin. 2015. The evolving perception of controversial movies. *Palgrave Communications* 1 (2015), 15038.

[2] Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol, Ofer Weintraub, and Irit Netanely. 2013. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 121–128.

[3] Runa Bhaumik, Chad Williams, Bamshad Mobasher, and Robin Burke. 2006. Securing collaborative filtering against malicious attacks through anomaly detection. In *Proceedings of the 4th Workshop on Intelligent Techniques for Web Personalization (ITWP'06), Boston*, Vol. 6. 10.

[4] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable delay functions. In *Annual International Cryptology Conference*. Springer, 757–788.

[5] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. 1997. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3 (1997), 77–87.

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[7] Shyong K Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*. ACM, 393–402.

[8] Brian Neil Levine, Clay Shields, and N Boris Margolin. 2006. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA* 7 (2006), 224.

[9] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 31–40.

[10] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 73–105.

[11] Ian MacKenzie, Chris Meyer, and Steve Noble. 2013. How retailers can keep up with consumers.

[12] Bhaskar Mehta. 2007. Unsupervised shilling detection for collaborative filtering. In *AAAI*. 1402–1407.

[13] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[14] Michael P O'Mahony, Neil J Hurley, and Guénolé CM Silvestre. 2005. Recommender systems: Attack types and strategies. In *AAAI*. 334–339.

[15] Kangning Wei, Jinghua Huang, and Shaohong Fu. 2007. A survey of e-commerce recommender systems. In *2007 international conference on service systems and service management*. IEEE, 1–5.

[16] Chad A Williams, Bamshad Mobasher, and Robin Burke. 2007. Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications* 1, 3 (2007), 157–170.

[17] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. 2017. Fake Co-visitation Injection Attacks to Recommender Systems.. In *NDSS*.

[18] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. 2008. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing* 16, 2 (2008), 435–447.