Attacking Recommender Systems With Plausible Profile

Xuxin Zhang, Jian Chen, *Student Member, IEEE*, Rui Zhang, *Member, IEEE*, Chen Wang, *Senior Member, IEEE*, and Ling Liu, *Fellow, IEEE*

***Abstract*— Recommender systems (RS) have become an essential component of web services due to their excellent performance. Despite their great success, RS have proved to be vulnerable to data poisoning attacks, which inject well-crafted fake profiles into RS, so that the target items can be maliciously recommended. In this paper, we first reveal that existing poisoning attacks in RS can be detected effortlessly, as the features of the generated fake profiles cannot be inconsistent with those of normal profiles all the time. We further propose RecUP, a poisoning attack in RS that can generate plausible profiles whose features stay almost the same as the normal ones, based on Generative Adversarial Networks (GAN). To tailor GAN for poisoning in RS, we develop HRGAN and devise a loss function to guide the training of the generator, along with a masking operation with selected potentially powerful profiles, so that the final generated profiles can perform malicious recommendations as expected. Evaluations against various defense methods using three real-world datasets show that, RecUP can generate the most plausible profiles while maintaining comparable attacking performance compared with state-of-the-art attacks.**

***Index Terms*— Recommender systems, shilling attack, generative adversarial network.**

I. Introduction

RECOMMENDER systems (RS) aim to find the data that matches a user's preference, given the user's historical user-item interaction behavior (e.g., ratings or clicks, referred to as user profiles, or profiles for short) and other useful information [1], [2]. RS help people find their interested items by analysing profiles, which brings huge economic benefits. As a result, RS have become the key component of many web services such as Amazon [3], Facebook [4], Google [5].

Meanwhile, various studies [6]–[12] have pointed out that RS are vulnerable to the so-called data poisoning attacks (a.k.a. shilling attacks [13]), which inject well-crafted fake profiles into RS, so that the target items can be recommended to more/less users for shopping/competition [14]–[17]. To avoid being detected easily by existing defenders in RS [18]–[20], recent studies try to launch poisoning attacks with special considerations, such as designing attacks with stealth in mind [8], [9], or making other recommendations of the system less perturbed [14].

In this paper, we argue that existing poisoning attacks in RS can still be detected effortlessly, since most if not all the attacks generate fake profiles relying on some global statistics, e.g., average rating value and rating variance for each item [13], [21], [22]. Using a simple detector based on seven representative features commonly used by the existing detection techniques in RS, we examine the feature difference between normal profiles and the fake ones generated by different attacks. The results show that the significant difference exists in more than one features for all the attacks (c.f. Table I), indicating an over-estimation of existing attackers' ability to disguise fake profiles.

Against this background, we move one step forward and propose RecUP, a poisoning attack in RS that can generate plausible profiles whose features stay almost the same as the normal profiles. RecUP leverages the generative adversarial networks (GAN) to capture the complex user-item correlation, considering that the well-trained generator of GAN is able to generate synthetic data with features indistinguishable from those of the ground-truth. To tailor GAN for poisoning in RS, we develop HRGAN and devise a loss function to guide the training of the generator, so that the generated profiles can perform malicious recommendations as expected. In addition, inspired by the fact that top-N recommendations may be more affected by some influential profiles, we further select the most positive influential profiles as additional input to the generator to further enhance the attack effect. It should be emphasized that RecUP requires neither the learning algorithms nor parameters of the target RS but only the user-item rating matrix, which does not exceed what existing attacks require to know.

In summary, our major contributions are as follows.

- We devise a simple yet effective detection classifiers incorporating seven typical features commonly adopted by existing detection techniques in RS, and reveal that

Manuscript received March 24, 2021; revised August 9, 2021; accepted September 13, 2021. Date of publication October 1, 2021; date of current version October 19, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61872416, Grant U20A20181, Grant 52031009, Grant 62002104, Grant 62071192, and Grant 62171189; in part by the Fundamental Research Funds for the Central Universities of China under Grant 2019kfyXJJS017; and in part by the Special Fund for Wuhan Yellow Crane Talents (Excellent Young Scholar). The work of Ling Liu was supported in part by the National Science Foundation CISE Grant 2038029, Grant 2026945, and Grant 1564097; in part by IBM Faculty Award; and in part by Cisco Grant on Edge Computing. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Issa Traore. *(Corresponding author: Chen Wang.)*

Xuxin Zhang, Jian Chen, and Chen Wang are with Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xuxinz@hust.edu.cn; jianchen@hust.edu.cn; chenwang@hust.edu.cn).

Rui Zhang is with Hubei Key Laboratory of Transportation Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China (e-mail: zhangrui@whut.edu.cn).

Ling Liu is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ling.liu@cc.gatech.edu).

Digital Object Identifier 10.1109/TIFS.2021.3117078

existing poisoning attacks in RS are easy to be detected as the features of their generated fake profiles cannot be inconsistent with those of normal profiles all the time.
- We design a novel GAN-based framework that considers both the purpose of data poisoning attacks and simulating the features of normal profiles, central to which is the novel design of a loss function, along with a masking operation with selected potentially powerful profiles to enhance the attack effect.
- We conduct a thorough evaluation on RecUP against various defense methods using three real-world datasets. The results show that RecUP can generate the most plausible profiles while maintaining comparable attacking performance compared with state-of-the-art attacks. It is also noticed that RecUP is still effective when only partial rating profiles are available, and can perform even better given extra knowledge of user-specific attributes.

The remainder of this paper is as follows. Section II reviews some related work on poisoning attack/detection methods. Section III presents our motivation, as well as the threat model, followed by the detailed design of RecUP in Section IV. Section V evaluates the performance of RecUP, and finally, Section VI concludes our work. The code of RecUP has been released for reproducibility purposes.[1]

II. Related Work

In this section, we review some poisoning attacks and detection methods in RS that are directly related to our work. More related work can be referred to some recent surveys such as [2], [23]–[25].

A. Poisoning Attacks in RS

Data poisoning attacks have been recognized in RS for many years [7], [9], [13], [26], [27]. Earlier works focus on creating hand-crafted fake profiles and do not achieve satisfactory attack performance, e.g., the random attack (Random) [13], the average attack (Average) [13], and the bandwagon attack (Bandwagon) [28]. Since 2016, there are several studies designed poisoning attacks for specific types of RS, e.g., association-rule-based RS [15], graph-based RS [16], and matrix-factorization-based RS [8]. For instance, Li *et al.*, [14] propose a poisoning attack to make the root-mean-squared-error larger than its original value in RS. They confirm that target attacks (dubbed as PGA attack and SGLD attack) are more effective in manipulating ratings of specific items. In recent work [8], Fang *et al.* use a subset of influential profiles to optimize the malicious ratings instead of using all normal profiles. Based on the given subset S of influential profiles, they develop a series of gradient-based optimization algorithms (including S-TNA-Inf and U-TNA) to determine the malicious rating scores. Lin *et al.* [9] present a novel augmented shilling attack framework (AUSH), one more specially tailored for RS instead of directly using adversarial attacks against the learning model. In recent work [7], Christakopoulou *et al.* use the GAN-based algorithm to satisfy the unnoticeability goal and take the learning algorithm for the adversary's strategy to achieve the intended attack against the oblivious recommender. However, these attacks are at the risk of being detected due to significant difference in features between their fake profiles and normal ones.

[1] https://www.dropbox.com/s/ssmfyhwtha28a8h/RecUP-code.zip?dl=0

B. Poisoning Detection in RS

Usually, poisoning attack detection is considered as a binary classification task, where the classification result of each profile can be normal or abnormal [29]. Thus, most detection methods are performed using machine learning techniques to distinguish abnormal profiles among all profiles based on detection features. These methods can be roughly categorized into supervised classification [30], [31], unsupervised clustering [20], [32], semi-supervised techniques [18] and other techniques [33].

In particular, Cao *et al.*, [18] propose a semi-supervised learning based shilling attack detection algorithm (SemiSAD). It first trains a naive Bayes classifier on a small set of labeled profiles and then incorporates unlabeled profiles with EM-λ to improve the initial naive Bayes classifier. In [34], the naive Bayes algorithm is exploited as a basic model and the final detector called PopSAD is trained on both collected normal and fake profiles. Aktukmak *et al.*, [20] propose a probabilistic factorization model to embed the available ratings and mixe user attributes into the latent space to generate anomaly statistics for new profiles. They identify the persistent outliers by unsupervised sequential attack detection algorithm. Cai *et al.*, [35] propose a novel unsupervised detection model based on analysis of user rating behavior.

Considering that supervised detection methods can achieve higher accuracy in detecting various poisoning attacks with the prior knowledge of the attacks, we evaluate our attack based on supervised detection methods.

III. Problem Formulation

In this section, we introduce seven representative features that were commonly used in poisoning detection in RS, and show that existing poisoning attacks can be simply detected using these features. On this basis, we present the threat model of our RecUP, with the basic idea how RecUP strengthens the attack effect.

A. Detection Features

The key features used by the existing detection methods can be classified into two classes: generic attributes and model-specific attributes [23], [36], [37]. In our work, we only select seven typical features for investigation. More features can be adopted upon particular requirements.

1) Generic Attributes: Generic attributes are based on the general abnormal behavior of the user, and can be used for almost all attack types.

(1) Rating Deviation from Mean Agreement (RDMA): the average deviation of profiles per item, weighted by the reciprocal of the rating number of the item:

$$RDMA_u = \frac{\sum_{i=0}^{n_u} \frac{|r_u^i - \overline{r^i}|}{n_{r^i}}}{n_u} \quad (1)$$

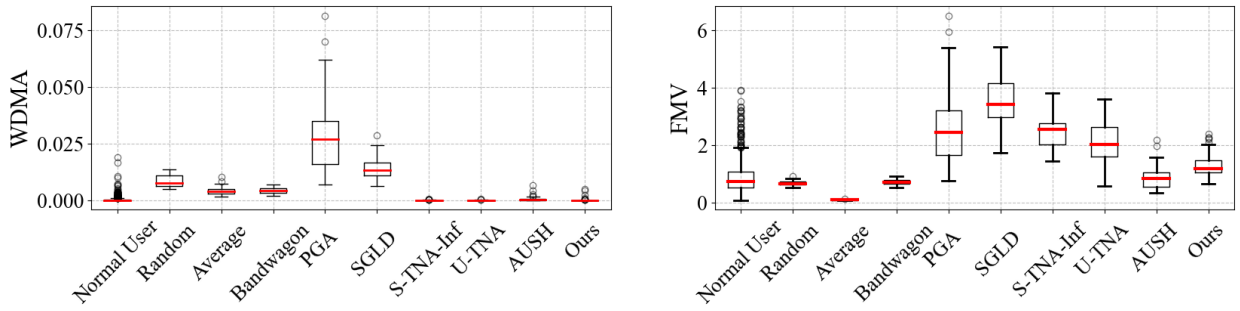


Fig. 1. The box-plot of different profiles regarding to two instanced features, WDMA and FMV, in ML-100K dataset. Outliers are plotted as individual points in the box-plot.

where n_u is the number of items user u has rated, r_u^i is the rating given by user u to item i, $\overline{r^i}$ is the average of ratings assigned to item i, and n_{r^i} is the overall number of ratings in the system given to item i.

(2) Weighted Deviation from Mean Agreement (WDMA): a weighted version of RDMA for sparse items:

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{|r_u^i - \overline{r}_i|}{n_{r^i}^2}}{n_u} \tag{2}$$

(3) Length Variance (LengthVar): a measure of how much the length of a given profile varies from the average length among all profiles:

$$LengthVar_u = \frac{n_u - \overline{n_u}}{\sum_{u \in U} (n_u - \overline{n_u})^2} \tag{3}$$

where n_u is the length of user u (i.e., the number of items that user u has rated), and $\overline{n_u}$ is the average length of all profiles in the system.

(4) Degree of Similarity with Top Neighbors (DegSim): the average similarity weighted with the top-k neighbors of user u [23]:

$$DegSim_u = \frac{\sum_{v=1}^{k} sim_{u,v}}{k} \tag{4}$$

where k is the number of neighbors, and $sim_{u,v}$ is the similarity between profiles u and v, that can be calculated via Pearson's correlation as in [18]:

$$sim_{u,v} = \frac{\sum_{i \in I}(r_u^i - \overline{r_u})(r_v^i - \overline{r_v})}{\sqrt{\sum_{i \in I}(r_u^i - \overline{r_u})^2 \sum_{i \in I}(r_v^i - \overline{r_v})^2}} \tag{5}$$

where $\overline{r_u}$ is the average of all ratings that given by user u.

2) Model Specific Attributes: Model specific attributes are used for specific types of attacks, e.g., some attributes for the average attack, while some for the random attack. They normally involve the filler items which are filled with fake ratings, except for the target item, to obstruct detection of the attack.

(1) Filler Mean Variance (FMV): for the average attack and is defined as:

$$FMV_u = \sum_{i \in l_f} \frac{(r_u^i - \overline{r^i})^2}{|l_f|} \tag{6}$$

where l_f is the filler item set that all items scored by user u except the target items. This attribute aims to capture abnormal variances between the individual mean of each item and the ratings of the filler items of the profile.

(2) Filler Mean Difference (FMD): for the average attack and is defined as:

$$FMD_u = \frac{\sum_{i=0}^{n_u} |r_u^i - \overline{r^i}|}{n_u} \tag{7}$$

This attribute computes the average value of the absolute value of the difference between the user's rating score and the average rating score for the hypothesized filler items (unlike FMV, which uses the squared value).

(3) Filler Average Correlation (FAC): for the random attack and is defined as:

$$FAC_u = \frac{\sum_{i=0}^{n_u} |r_u^i - \overline{r}_i|}{\sqrt{\sum_{i=0}^{n_u} |r_u^i - \overline{r}_i|^2}} \tag{8}$$

This attribute calculates the correlation between the filler ratings in the profile and the average rating for each item.

B. Motivation

We employ the aforementioned seven features to examine whether existing attacks are easy to be detected. Specifically, we take all the profiles in MovieLens-100K (ML-100K) dataset [38] as normal profiles, and generate fake profiles based on different attack methods with 5% attack size. After that, we calculate the values of each feature of normal profiles and fake profiles for each attack. In this way, we can observe the distinct distribution difference in features between normal and fake profiles. For example, c.f. Fig. 1 shows the distribution of two features, $WDMA$ and FMV for different profiles, and we can see that compared with normal profiles, fake profiles have evident differences, significant for some attacks while less significant for others.

On this basis, we further train a series of binary classifiers based on different detection methods (including SemiSAD [18], PopSAD [34], and others described in Section V-A.2), and test the difference in each feature between normal profiles and fake ones generated by each attack. For example, we calculate the values of $WDMA$ of normal profiles and fake ones generated by the random attack, and then we split the results, 70% for training and the rest for testing.

TABLE I
FEATURE DIFFERENCE BETWEEN NORMAL PROFILES AND GENERATED FAKE PROFILES IN ML-100K DATASET

Feature \ Attack	Random [13]	Average [13]	Bandwagon [28]	PGA [14]	SGLD [14]	S-TNA-Inf [8]	U-TNA [8]	AUSH [9]	**RecUP**
RDMA	√	√	√	√	×	√	√	√	√
LengthVar	×	×	×	×	√	×	×	×	√
DegSim	×	×	×	×	×	√	√	√	√
WDMA	×	×	×	×	×	√	√	√	√
FMV	√	×	√	√	×	×	√	√	√
FMD	√	×	√	×	×	√	√	√	√
FAC	√	√	×	√	√	×	×	×	√

Inconspicuous difference : √ Significant difference : ×

Next, we train a binary classifier via SemiSAD with the above training set, and assess the classifier on the testing set. When the output is positive, we believe that there is a significant difference in the feature $WDMA$ between normal profiles and fake ones generated by the random attack, otherwise it is not. By changing the features and attack methods, we can distinguish the generated fake profiles and the normal ones (see the results in c.f. Table I). Note that we use different detection methods to train the binary classifier, and the final results are determined by the majority rule.

As can be observed in c.f. Table I, the fake profiles generated by different attacks can be resemblant with normal profiles in some features, while not in others. Therefore, there is still much room for the improvement of the plausibility of the fake profiles, and right now these attacks can be easily detected when different features are considered by our simple detection classifiers. This motivates us to develop RecUP that can generate plausible profiles in all features, as indicated in c.f. Table I, to enhance the degree of camouflage.

C. Threat Model

In this section, we provide a detailed threat model for our RecUP. The threat model consists of identifying the attacker's goal, attacker's knowledge of the RS, attacker's capability of manipulating the training data, and eventually lead to the attack strategy.

1) Attacker's Goal: There are generally two types of poisoning attacks in RS [24]: the push attack (resp. the nuke attack), which aims to promote (resp. demote) the target item i^{tg} to as many normal profiles as possible. Considering that these two types of attacks can be essentially converted to each other, we thus focus on push attacks in this paper, as generally done in recent works [7]–[12]. Meanwhile, we also try to avoid the detection as much as possible, so that the attack effect can takes a longer time. Note that the nuke attack version of RecUP can be implemented in a similar way as done in this work.

2) Attacker's Knowledge: We consider the target RS based on mere the rating matrix, e.g., the neighborhood-based RS [39], the matrix-factorization-based RS [40], and the deep learning based RS [41], [42]. In some studies, the attacker is assumed to have full knowledge of the target RS, including all the rating data (i.e., the rating matrix consisting of m profiles and n items), and both the learning algorithms and parameters of the learner [14]. However, in practice, the recommendation model is often deployed as a black box, and it is only possible to obtain the profiles' ratings, e.g., through directly browsing profiles' ratings when shopping on Amazon.

Therefore, in our work, we consider a more practical scenario, where the attacker has the knowledge of all users' ratings, but knows nothing about the learning algorithms and parameters. In fact, our work does not depend on the type of the targeted RS deployed by the service provider and only needs the rating matrix, which exists in most RS, e.g., the neighborhood-based RS [39], the matrix-factorization-based RS [40], and the deep learning based RS [41], [42]. At times, some individuals may generate anonymous ratings for the sake of privacy, or some more user-specific attributes such as one's gender and age are provided. In these cases, we will show in the experiments that RecUP is still effective even when only a part of ratings can be access to (c.f. Section V-E.1), and becomes more powerful when extra attributes are available (c.f. Section V-E.2).

3) Attacker's Capability: Given the rating matrix of the RS, the attacker is capable to inject $\alpha \times m \in \mathbb{N}$ fake profiles into the training data matrix, each fake profile with at most β items regarding the rating score in the same range as that of a normal profile. Here, α is the *attack size* (a.k.a. the poisoning rate) that signifies the fraction of the training data controlled by the attacker, and β is the *profile size* that indicates the number of non-zero ratings (in other works [9], [25], β is also called the filler size, and is referred to as the number of items that one fake user could rate at most). The attacker's ability in our work is reasonable, since in many realistic scenarios, the attacker can take control of only a small fraction of profiles in the RS. For example, an attacker could register and falsify a number of profiles by leveraging compromised machines [16].

4) Attacker's Strategy: To achieve the goal, the attacker seeks for an optimal rating score vector for each fake profile to maximize the $HitRatio$ of the target item i^{tg}, which can be formulated as:

$$\max\ HitRatio^{tg} = \frac{\sum_{u \in U_{/tg}} H_u^{tg}}{\|U_{/tg}\|} \tag{9}$$

where H_u^{tg} is the hit-function defined as:

$$H_u^{tg} = \begin{cases} 1 & \text{if } i^{tg} \in I_u, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

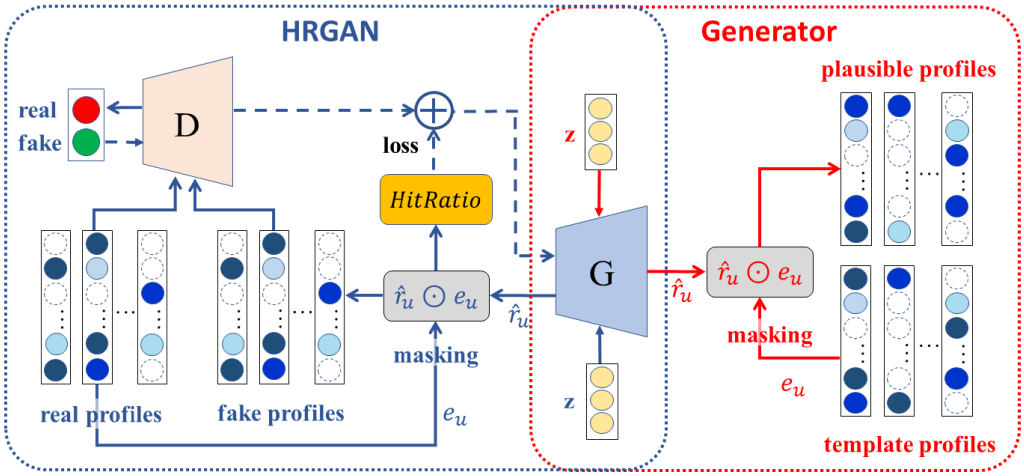


Fig. 2. The framework of RecUP.

and I_u denotes the set of top-N items that the RS recommend to the user u. $U_{/tg}$ denotes the set of profiles that exclude the target item i^{tg}, and $\|\cdot\|$ denotes the cardinality of the set.

It is pointed out that the above hit ratio maximization problem is NP-hard in general [8], [16], so it is quite challenging to find an optimal solution, let alone taking into account the variety of detection features discussed in Section III-A to escape the detection to the greatest extent. What makes it even harder is that the recommendation model is treated as a black box: neither the learning algorithm nor the parameters of the model are not fully informed. Therefore, we propose to generate plausible profiles with the idea of GAN.

IV. RecUP Design

Given the user-item rating matrix, RecUP aims to generate plausible profiles that can maximize the hit ratio of the target item at the greatest extent. To this end, RecUP mainly involves the following two steps (c.f. Fig. 2):

(1) *Constructing Generator with HRGAN*. RecUP develops a variant GAN dubbed HRGAN for poisoning RS, with a loss function to enhance the purpose of data poisoning attacks. The generator of HRGAN acts as the "attacker" to generate plausible profiles and the discriminator of HRGAN acts as the "defender" to recognize the fake profiles whose features are different from normal profiles.

(2) *Generating Plausible Profiles*. The well-trained generator with the masking operation can generate plausible profiles whose features are consistent with those of ground-truth, and potentially powerful profiles are selected as template profiles for masking to further enhance the attack effect.

A. Constructing Generator With HRGAN

1) GAN-Based RS: The original GAN [43] contains two neural networks: a generative model (shortly, G) and a discriminative model (shortly, D). GAN performs adversarial learning between G and D. In particular, G learns to capture the distribution of the real data $P_{data}(\boldsymbol{x})$ from a prior noise distribution $\boldsymbol{z}$, while D learns to distinguish the real data

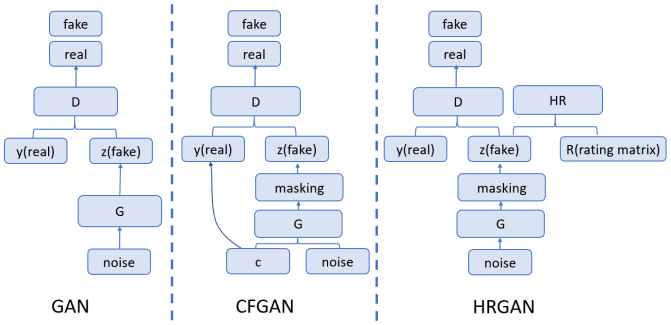


Fig. 3. A comparison of network architectures in GAN [43], CFGAN [46] and our HRGAN.

samples $\boldsymbol{x}$ from $G(\boldsymbol{z})$. Formally, G and D are playing the following min-max two-player game simultaneously:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim P_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim P_z(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))] \quad (11)$$

In the field of RS, IRGAN [44] and GraphGAN [45] are the earliest GAN-based recommendation methods. Their main idea is to let G generate discrete items a given user might purchase, and to let D discriminate a user's ground truth items from those generated by G. However, there are considerable limitations of the *discrete item index* generation approach in those GAN-based methods (see detailed explanation in [46]). CFGAN [46] takes better results by adopting *real-valued vector-wise* adversarial training, where after training, G generates a dense vector containing predicted rating scores for all items. Still, it is not normal for a user to have rated all items. Thus, CFGAN maskes G's output by multiplying with an n-dimensional purchase vector of the user (called the *template*), in order to meet the sparsity in ground-truth data that G tries to mimic.

2) HRGAN Design: Following CFGAN, we design HRGAN (Hit-Ratio-GAN) adopting some techniques such as real-valued vector-wise adversarial training as well as masking with the template to meet the peculiarity of the real RS

scenario. The difference is that we enhance the attack power by adding a well-designed loss function to guide the training of the generator (c.f. Fig. 3), so that the generated profiles are able to recommend the target item to as many normal profiles as possible.

In the design of HRGAN, the objective function of D, denoted as J^D, is as follows:

$$\begin{aligned} J^D &= -\mathbb{E}_{\boldsymbol{x}\sim P_{data}}[\log D(\boldsymbol{x})] - \mathbb{E}_{\hat{\boldsymbol{x}}\sim P_\phi}[\log(1 - D(\hat{\boldsymbol{x}}))] \\ &= -\sum_u \log D(r_u) - \sum_u \log(1 - D(\hat{r}_u \odot e_u)) \\ &= -\sum_u (\log D(r_u) + \log(1 - D(\hat{r}_u \odot e_u))) \end{aligned} \tag{12}$$

and the objective function of G is:

$$J^G = \sum_u \log(1 - D(\hat{r}_u \odot e_u)) \tag{13}$$

where e_u is an n-dimensional indicator vector specifying whether u has rated the item i ($e_u^i = 1$) or not ($e_u^i = 0$), and $\odot$ stands for element-wise multiplication.

We define both of the objective functions to be a minimization problem, and the main difference from the original GAN is that we mask G's output $\hat{r}_u$ by multiplying it with e_u, to simulate the sparsity of the real data that G tries to mimic. Here, random selection is employed in the template profile formation, as it is beneficial for learning more comprehensively about the distribution of real user ratings. This difference is consistent with CFGAN's framework [46], and only G's output on the rated items can contribute to the learning of G and D.

As such, in HRGAN the input of G is the random noise vector z, and the output is $\hat{r}_u$, an n-dimensional vector that represents a possible rating vector of the user u rated to all items. Then, the input of D is whether $\hat{r}_u \odot e_u$ from G or r_u from real data, and its output is a single scalar value in the range of [0, 1], which represents the probability that its input comes from the real data rather than G.

In addition, we expect the predicted rating scores $\hat{r}_u^{tg}$ that user u gives to the target item i^{tg} to go as large as possible, and the higher rank of i^{tg} in I_u, the better it would be. We notice that the concise sigmoid function satisfies our demands: a higher rank of i^{tg} in I_u would lead to a smaller loss. Since the range of J^G is only between 0 and 1, and the sigmoid function is differentiable, we thus use the sigmoid function to approximate this purpose as follows:

$$L_H = \sum_{u\in U} \sum_{i\in I_u} g(\hat{r}_u^i - \hat{r}_u^{tg}) \tag{14}$$

where $g(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. For each item $i \in I_u$, if $\hat{r}_u^i < \hat{r}_u^{tg}$, then the loss is smaller, and the rank of i_{tg} in I_u is higher. We thus add L_H into J^G as:

$$J^G = (1 - \lambda) \cdot \sum_u \log(1 - D(\hat{r}_u \odot e_u)) + \lambda \cdot L_H \tag{15}$$

where $0 < \lambda < 1$ is a coefficient that trades off the impact of different parts of the attack objective function.

We implement both G and D as multi-layer neural networks, the number of layer are $L^G >= 2$ and $L^D >= 2$, parameterized by ϕ and θ, respectively. We train our G and D networks by using the stochastic gradient descent (SGD) with minibatch and back-propagation, and alternately update the parameters of each network, ϕ and θ, keeping one fixed when the other being updated (c.f. Algorithm 1). After the adversarial training is completed, we can obtained the final generator (G).

Algorithm 1 Training Procedure of HRGAN

Input: User-item rating matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, learning rate for G and D: μ_G and μ_D, minibatch size for G and D: M_G and M_D, training step for G and D: $step_G$ and $step_D$, parameters c, λ.
Output: G's parameters θ.

Initial ϕ and θ;
for c **do**
  for $step_G$ **do**
    Noise sampling $\{z_1, z_2, \ldots, z_{M_G}\}$ from noise prior;
    Sample minibatch of M_G profiles from normal profiles as template profiles e_u;
    Calculate J^G by Equation (15);
    Update G by $\theta \leftarrow \theta - \frac{\mu_G}{M_G} \cdot \nabla_\theta J_G$;
  end for
  for $step_D$ **do**
    Noise sampling $\{z_1, z_2, \ldots, z_{M_D}\}$ from noise prior;
    Sample minibatch of M_D profiles from normal profiles as template profiles e_u;
    Calculate J^D by Equation (12);
    Update D by $\phi \leftarrow \phi - \frac{\mu_D}{M_D} \cdot \nabla_\phi J_D$;
  end for
end for
return θ

Algorithm 2 Selecting Potentially Powerful Profiles

Input: User-item rating matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, attack size α
Output: Potentially powerful profiles P.

Initial $P = \emptyset$;
for $i = 1$; $i < m$; $i++$ **do**
  Take Δ clone profiles of user u_i as S_u;
  Calculate the attack power of u by Equation (16);
end for
$P \leftarrow$ top-Δ profiles regarding to the attack power;
return P

B. Generating Plausible Profiles

Given the well-trained generator (G), along with the masking operation, we can generate $\lfloor \alpha \times m \rfloor$ plausible fake profiles based on the sampled noise and template profiles e_u that are sampled from normal profiles (see the right part in Fig. 2).

The template profiles e_u can be identified in a naive manner by randomly selected from all the real profiles to generate plausible profiles. This move ensures the sparsity of real profiles in the RS, but neglects the fact that the top-N recommendations results may be more influenced by some influential profiles [47], [48]. Therefore, instead of using random selection to build the template profiles, we attempt to select those profiles that have more attack power to enhance the effectiveness of our attack.

Considering our goal is to find an optimal rating score vector for fake profiles to maximize the $HitRatio$, we can evaluate the attack power of the user u by injecting the clone profile set S_u that contains a certain number of clone profiles of u into the RS and calculate the corresponding $HitRatio$:

$$\begin{aligned} &\mathcal{A}(u, tg) = HitRatio_{tg} \\ &s.t.\ \|S_u\| = \Delta \end{aligned} \tag{16}$$

where Δ is the set size (i.e., the number of clone profiles of user u in S_u) and $HitRatio_t$ is calculated by Equation (9). Normally, a small Δ may not be powerful enough to push the rank of the target item into top-N item list, while a large Δ may bring about extra overhead. In this sense, we set Δ based on the attack size:

$$\Delta = \lfloor \alpha \times m \rfloor \tag{17}$$

where $\lfloor * \rfloor$ means fractions are rounded down.

By injecting the subset S_u that include Δ clone profiles of u into the RS, we calculate the $HitRatio$ and take the result as the attack power of u's profile (c.f. Algorithm 2). After calculating the attack power of all profiles, we take the top-Δ profiles as the template profiles.

V. Performance Evaluation

A. Experiment Setup

1) Datasets: We evaluate RecUP typically on three RS datasets, namely ML-100K [38], FilmTrust (FL) [49] and CiaoDVD [49] (c.f. Table II). In ML-100K and CiaoDVD datasets, the rating scores of profiles are given on the scale $\{1, 2, 3, 4, 5\}$ with 1 the lowest and 5 the highest. The rating scores of FilmTrust are on the scale of $0.5 \sim 4$ with a step 0.5, and we scale up these ratings into $1 \sim 5$ for convenience. Incidentally in CiaoDVD dataset, some repetitive ratings are graded by the same user to the same item at different time. To capture the user's most recent interests, we merge these ratings to the latest timestamp.

To evaluate RecUP in scenarios where the attacker has knowledge of user-specific attributes (such as gender and age, etc.), we adopt ML-100K [38] and Restaurant & Consumer (R&C) [50] datasets. In particular, ML-100K involves 4 private attributes, namely gender, age, occupation, and zip code. R&C is used to predict consumer ratings given to different restaurants [50], and it includes 1,161 ratings (we use the rating of overall quality rather than food quality or service quality) by 138 profiles on 130 restaurants, with 21 attributes of each user, including drink level, birth year, personality, etc. Unlike ML-100K, the rating values in R&C is $\{0, 1, 2\}$ where 2 means the user likes this restaurant, 0 means unlike, and 1 means general. To distinguish with lack of ratings, we convert rating score 0 to 1, 1 to 3, and 2 to 5 (considering that 5 in ML-100K means the user likes this item and 1 means unlike). After preprocessing, these private attributes constitute the condition constraints for each user when we train the condition version of HRGAN (c.f. Section V-E.2).

2) Baseline Attacks and Detection Methods: We compare RecUP with attacks mentioned in Section II-A and Table I as baselines, including **Random** [13], **Average** [13], **Bandwagon** [28], **PGA** [14], **SGLD** [14], **S-TNA-Inf** [8], **U-TNA** [8], **DCGAN** [7] and **AUSH** [9]. In addition, we also consider the case when only the template profiles are added without masking with the generated profiles, which we refer to as **Poi-TP** (**poi**soning attack based on **t**emplate **p**rofiles) in the following experiments.

We evaluate our attack along with baselines using two well-known supervised detection methods: **SemiSAD** [18] and **PopSAD** [34]. In addition, we also adopt classic classification algorithms for detection, as commonly used in [18], [36], including the support vector machine (**SVM**), the random forest (**RF**), the decision tree (**DT**), the gradient boosting decision tree (**GBDT**), Adaptive Boosting (**AdaBoost**), and Gaussian Naive Bayes (**GaussianNB**). We calculate all the seven features in Section III-A of normal and fake profiles. Then, we split the results into training set (70%) and testing set (30%), learn a binary classifier via these detection methods based on the training set, and evaluate the binary classifier on the testing set.

3) Evaluation Metrics: We evaluate the performance of RecUP from two aspects: the attack effect and the undetectability. We take the widely used ***HR@N*** as the evaluation index of attack effect, where HR@N of a target item means the fraction of normal profiles whose top-N recommendation lists contain the target item after injecting the fake profiles. We used ***F1-Score*** to measure the undetectability, i.e., the performance of different attacks in face of poisoning detections. F1-Score is a standard metric with a combination of Precision and Recall. They are the fraction of the predicted true attackers to the predicted attackers and true attackers.

4) Experiment Settings: In the experiment, we take the matrix factorization based RS as our targeted recommender, for the convenience of comparing the performance of different attack methods, where we use the factorization based collaborative filtering algorithm [14] to calculate the loss function in the training process. To evaluate the performance of RecUP, we use a workstation equipped with an AMD Ryzen 7 3700X CPU and NVIDIA GeForce GTX 1080 Ti graphic card for executing deep learning applications. We use Pytorch 1.6.0 as the deep learning framework and define networks for HRGAN. We use Adam [51] for optimization. Both of generator and discriminator of HRGAN have 3 hidden layers, the output layer size is the number of all items for the generator.

We use the following default parameter settings unless otherwise specified: both the learning rate for G and D, μ_G and μ_D, are 0.0002; the minibatch size is 32; $step_G = 5$, $step_D = 2$; the number of adversarial training epochs is 50; attack size is set to 5%. We select target items randomly from all items whose average predicted ratings are around 0.8, as in [14], and $N = 10$ when computing HR@N (i.e., top-10 recommendation). Note that we do not set a certain value for the profile size (or the filler size), as we take the selected normal profiles as the template to mask our generated profiles. This operation ensures that the profile size of our final generated profile is of the same size as that of the normal user.

TABLE II
DATASET STATISTICS

Dataset	#Profile	#Item	#Rating	Density	#Attr.
ML-100K	943	1,682	100,000	6.30%	4
FilmTrust	1,508	2,071	35,497	1.14%	—
CiaoDVD	17,615	16,121	72,665	0.03%	—
R&C	138	130	1,161	6.47%	21

TABLE III
FILTERING COLD-START PROFILES IN CIAODVD

a	#Profile	#Item	#Rating	Density	HR@10
1	17,615	16,121	72,665	0.03%	0.0050
2	7,143	7,036	53,784	0.11%	0.2448
3	3,985	4,162	42,257	0.25%	0.2055
4	2,662	2,873	34,707	0.45%	0.2661
5	1,822	2,069	28,374	0.75%	0.3642
6	**1,345**	**1,598**	**23,787**	**1.11%**	**0.4257**

We set the training/test split as 7/3, which means 70% for training and the rest 30% for testing in all feature vectors. We report the average attack performance of three tests on datasets, and we highlight the best performance in each dataset for each attack size.

B. Warming-Up Experiments

1) Filtering Cold-Start Profiles: As pointed out in [9], [14], [25], cold-start profiles (i.e., those with few items) are too vulnerable, because there is usually insufficient information to produce reasonable recommendation to users. So we conduct experiments on CiaoDVD dataset, considering that its density is the lowest in Table II. The density here is obtained by dividing the total number of users' rating scores by the size of the rating matrix, and we increase the density by filter profiles with items less than a ratings. With the increment of the density, the performance of attack (HR@10) increases, as shown in Table III. This is because the higher density means fewer cold-start profiles, which leads to reasonable recommendation and stable attack effect. We observe that the satisfactory result got when the density approaches 1%. So, we take 1% as the threshold value of density to determine whether we need to filter a dataset to be attacked. To get better attack performance (e.g., get more reasonable recommendation to increase credibility and raise the $HitRatio$), we filter profiles of dataset CiaoDVD with items less than $a = 6$ ratings (including those without ratings), and other datasets are not processed.

It is also noticed that a denser dataset might be more robust against attacks under some circumstances [52]. The attack effect of different datasets cannot be determined simply by the density of datasets, considering that there are so many data characteristics that can affect the effectiveness of an attack, such as the data density, the shape of user-item matrix rating matrix, the rating mean and the rating variance. We believe that more work is needed to explore the relation between a sparser dataset and the stability of the attack effect.

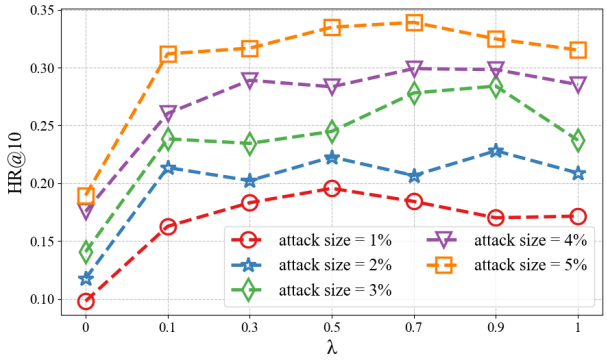


Fig. 4. Attack performance on ML-100K with different λ.

2) Choice of Coefficient λ: As shown in Equation (15), we employ the coefficient λ to trade off the impact of different parts of the objective function. When λ is set to zero, it means we do not care about the impact of the attacker's goal. On the other hand, if we set λ to one, we actually do not care whether the generated profiles can be recognized by the discriminator. To choose a proper value of λ, we generate fake profiles with different values of λ and test their performance on both attack effect and its undetectability. As can be seen in Fig. 4, different values of λ do not seem to make much difference to the results of the poisoning attack, except $\lambda = 0$. The principal reason is that when λ comes to zero, it means we do not include the loss function that approximates the purpose of poisoning attack in our framework (c.f. Fig. 2). As a result, the attack effect is far from satisfactory (c.f. Fig. 4) despite its good performance against detection methods (c.f. Fig. 5). When taking into consideration our attack purpose with the same weight as the need to avoid identification by discriminator ($\lambda = 0.5$), the performance is better on the whole. Therefore, we set $\lambda = 0.5$ in the following experiments unless otherwise stated.

C. Performance of RecUP

1) Attack Performance: Our goal is to promote the target item i^{tg} to as many normal profiles as possible and maximize the HR@10. We first investigate the attack performance of RecUP and compare it with other baselines on four datasets: ML-100K, FilmTrust, CiaoDVD and R&C. We select the target item as in [14] and consider the result of the top-10 recommendation list for each user.

We can see from the results in Table IV that, RecUP can effectively promote the target items with different attack sizes of fake profiles. Compared to other attacks, RecUP generally achieves better attack performance in ML-100K, FilmTrust and R&C. In CiaoDVD, RecUP's performance also maintains at a relatively high level. The main reason is that we consider the purpose of poisoning attack in our framework and enhance the attack performance by feeding template profiles with high attack power. It is worth mentioning that the results shown in Table IV are the mean values of three tests. Due to the space limit, we do not report the standard deviation of those results.

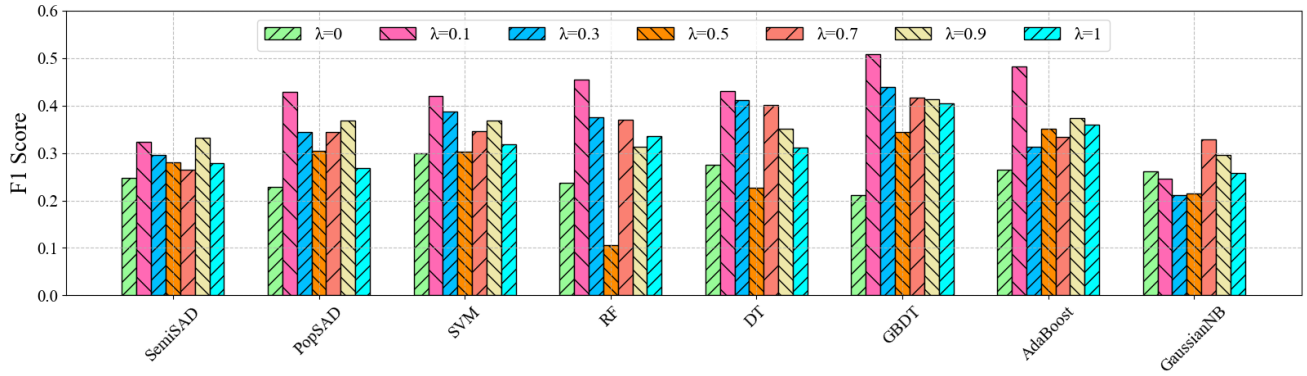


Fig. 5. Attack detection on ML-100K with different values of λ.

TABLE IV
ATTACK PERFORMANCE WITH DIFFERENT ATTACK SIZES

Dataset	Attack Size	Attack Method										
		Random	Average	Bandwagon	PGA	SGLD	S-TNA-Inf	U-TNA	AUSH	DCGAN	Poi-TP	**RecUP**
MovieLens -100K	1%	0.0659	0.0628	0.0687	0.0906	0.1014	0.1217	0.1167	0.0734	0.0131	0.1123	**0.1956**
	2%	0.0721	0.0750	0.0763	0.1353	0.1567	0.1624	0.1432	0.0844	0.0297	0.1509	**0.2227**
	3%	0.0735	0.0652	0.0842	0.1626	0.1883	0.2260	0.2037	0.1375	0.0898	0.1707	**0.2448**
	4%	0.0986	0.0802	0.0878	0.2043	0.2357	0.2752	0.2414	0.1760	0.0822	0.2030	**0.2834**
	5%	0.1229	0.1124	0.1212	0.2280	0.2464	0.3074	0.2954	0.1851	0.1367	0.2207	**0.3327**
FilmTrust	1%	0.0121	0.0121	0.0165	0.0263	0.0524	0.1031	0.0986	0.0723	0.0048	0.1027	**0.1418**
	2%	0.0134	0.0508	0.0508	0.0502	0.0661	0.1160	0.1210	0.0943	0.0072	0.1353	**0.1916**
	3%	0.0698	0.0565	0.0639	0.0775	0.1121	0.1121	0.1305	0.1559	0.0447	0.1500	**0.2217**
	4%	0.0732	0.0632	0.0689	0.0882	0.1281	0.1173	0.1364	0.1700	0.0529	0.1862	**0.2257**
	5%	0.0719	0.0722	0.0675	0.1209	0.1499	0.1222	0.1483	0.1905	0.0734	0.2177	**0.2662**
CiaoDVD	1%	0.1147	0.1829	0.2565	0.1254	0.1514	0.2217	0.1926	**0.2617**	0.0098	0.1704	0.2324
	2%	0.2473	0.2321	0.2757	0.1317	0.2223	0.2863	0.2216	**0.3063**	0.0144	0.2463	0.2874
	3%	0.2554	0.2438	0.3079	0.2231	0.2502	**0.3158**	0.2652	0.2585	0.0336	0.3051	0.3131
	4%	0.2846	0.3136	0.2981	0.2751	0.3157	0.3286	0.2927	0.3478	0.0744	0.3228	**0.3711**
	5%	0.3151	0.2914	0.3026	0.3326	0.3673	0.3784	0.3433	**0.4315**	0.0925	0.3942	0.4257
R&C	1%	0.0597	0.0606	0.0222	0.0682	0.0827	0.0672	**0.1103**	0.0584	0.0172	0.0584	0.0847
	2%	0.0606	0.0992	0.0370	0.1136	0.1053	0.1818	0.1324	0.0882	0.0365	0.0876	**0.2556**
	3%	0.0815	0.1395	0.0677	0.1231	0.1538	0.2348	0.1679	0.1037	0.0662	0.1103	**0.3876**
	4%	0.1061	0.2240	0.2097	0.2358	0.2031	0.3308	0.2593	0.1735	0.0882	0.2444	**0.4546**
	5%	0.2937	0.2441	0.3008	0.3333	0.3388	0.3817	0.3585	0.2217	0.1185	0.3504	**0.5148**

According to our experimental results, the standard deviation of the attack performance remains at an acceptable level, while the standard deviation increases with the increase of the attack size. There is no significant relationship between the standard deviation of different datasets or different attack methods.

2) Performance Against Attack Detection: Some service providers could arm the RS with certain fake-user detection methods to minimize the risk of the potential poison attack. Attackers achieve their goal by injecting malicious profiles, meanwhile, those malicious profiles faced with be detected by detection methods. We apply some detection methods on the injected profiles generated by different attack methods and report the fake profiles' detection results in Figs. 6∼8.

Specifically, we use seven features (c.f. Table I) extracted from each user's ratings to train the detection classifiers. Fig. 6 depicts the detection results w.r.t. F1 Score on ML-100K, while Figs. 7 and 8 depict the FPR (False Positive Rate, which means the fraction of normal profiles that are predicted to be malicious profiles) and FNR (False Negative Rate, which means the fraction of malicious profiles that are predicted to be normal profiles), respectively. We can observe that RecUP performs the best against all the detection methods. The reason is that, the features of the fake profiles generated by RecUP act in a plausible manner as the normal profiles; thus the generated fake profiles are difficult for the detectors to recognize.

As shown in Fig. 7 and Fig. 8, RecUP achieves significantly higher values compared to other attacks, indicating that the profiles generated by RecUP are more indistinguishable from normal profiles. It is also noted that some of results shown in Figs. 7 and 8 are zero, which means that the profiles generated are accurately detected by the particular detection method. This is because that the features of these generated profiles

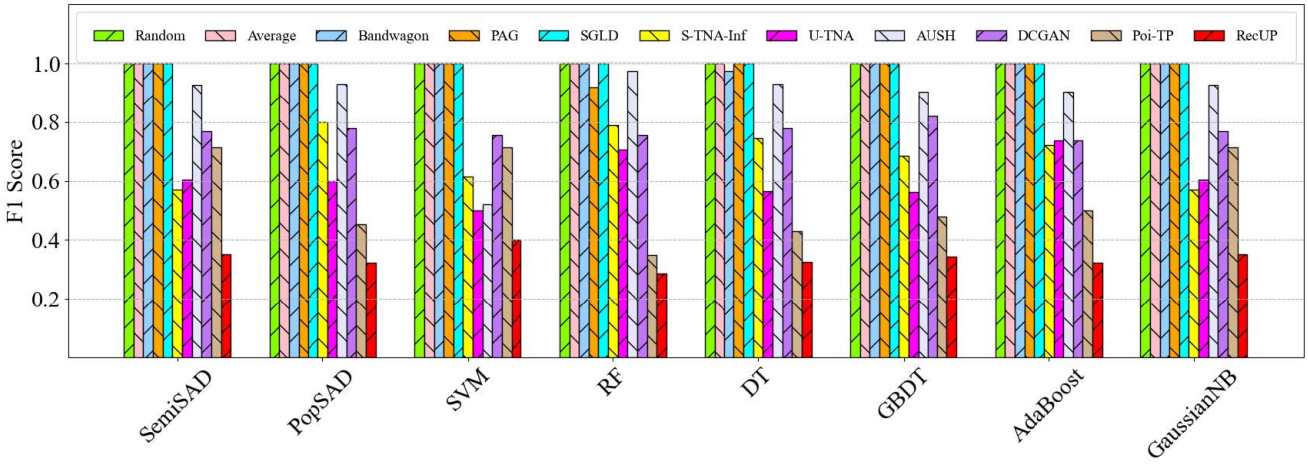


Fig. 6. Attack detection w.r.t. F1 Score on ML-100K with different attack methods.

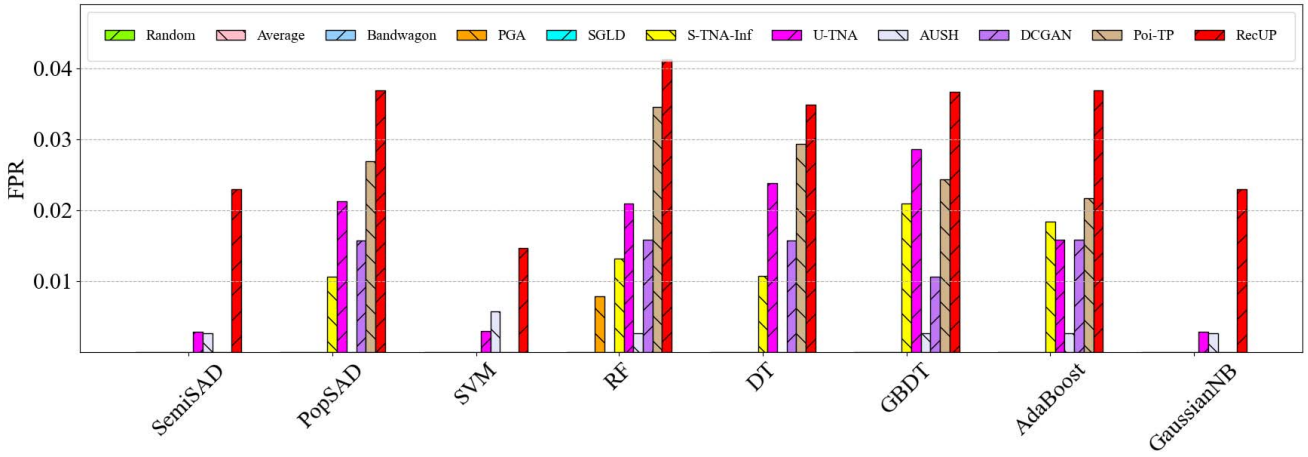


Fig. 7. Attack detection w.r.t. False Positive Rate (FPR) on ML-100K with different attack methods.

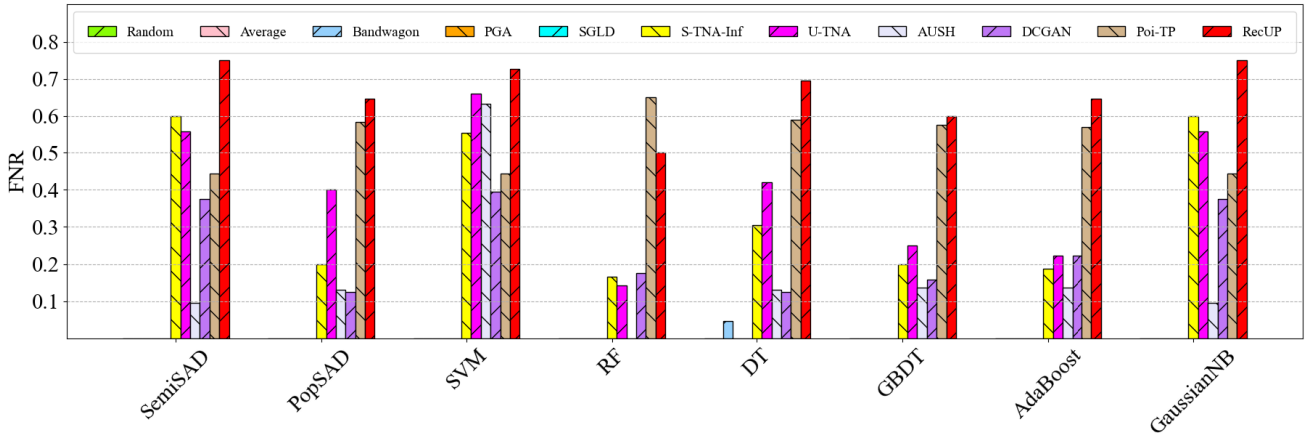


Fig. 8. Attack detection w.r.t. False Negative Rate (FNR) on ML-100K with different attack methods.

are significantly different from those of the normal users, yielding them being detected much easily. For other attacks, as revealed in Section III-B, they cannot cover all the features at the same time, so it is more detectable by the detectors. The results further verify the undetectability of RecUP to generate inconspicuous injections in poisoning attacks.

Compared with other methods, RecUP generally achieves satisfactory attack performance in most cases, and the attack

TABLE V
PERFORMANCE OF DIFFERENT COMPONENTS IN RECUP

Attack Method	HR@10	Detection Method							
		SemiSAD	PopSAD	SVM	RF	DT	GBDT	AdaBoost	GaussianNB
HRGAN	0.1776	0.290	0.360	0.362	0.218	0.258	0.394	0.448	0.288
GAN+Gen	0.1895	0.248	**0.228**	0.300	0.238	0.276	0.212	0.264	0.262
HRGAN+Gen	0.3327	**0.246**	0.292	**0.254**	**0.074**	0.314	**0.198**	**0.222**	**0.188**
HRCoT+Gen	0.2772	0.936	0.974	0.944	0.958	0.924	0.924	0.938	0.934
HRGAN_GP+Gen	**0.3452**	0.280	0.304	0.302	0.106	**0.226**	0.344	0.352	0.214

performance of RecUP is even far better than other attack methods when faced with elaborated detections.

D. Impact of Components in RecUP

In this section, we evaluate how much each component of RecUP can contribute to the attack effect. As shown in Fig. 2, we split our framework into two component (HRGAN+Gen): (1) The GAN-based module with the loss function that approximates $HitRatio$ (HRGAN); and (2) The generator module that generates plausible profiles based on selected potentially powerful profiles (Gen).

We remove or change some components of RecUP and investigate the performance changes as shown in Table V. We remove the loss function that approximates $HitRatio$ of HRGAN module (GAN+Gen) or generate plausible profiles just based on randomly selected normal profiles (HRGAN). The results indicate that both the potentially powerful profiles selection and the loss function contribute significantly to the attack effect. This is largely because that the loss function plays an important role in generating plausible profiles and the attack performance benefits greatly from the potentially powerful profiles selection.

To measure the contribution of the network structure of HRGAN, we evaluate this component by changing different network structure to train the generator. We select two successful cases as the template of our module to generate fake profiles, from recent literature [46], [53] about applying GAN to the RS. HRCoT means we add the loss function that approximates $HitRatio$ when we train CoT [53]. Consider a more robust version than general GAN, Wasserstein GAN gradient penalty [54] that is further adapted to learn the distribution of predictions of the augmented training data, we directly add the gradient penalty into the training (HRGAN_GP). All three variations of GAN adopt the masking when we generate the fake profiles. As we can see, the best performance when faced with detection methods just appears when we apply the HRGAN+Gen, while applying the HRGAN_GP+Gen gets the best results in HR@10. We believe that the main reason is that the gradient penalty helps the generator generate better profiles [54].

E. Impact of Attacker's Knowledge

1) Impact of Partial Knowledge: In this section, we consider the case where the attacker has access to only a subset of profiles. We construct the subset of profiles by randomly selected from all profiles one by one until we reach the size of available profiles. Fig. 9 shows the attack results when the attacker observes different amounts of normal profiles with different attack sizes. Note that in the partial-knowledge attack, we select potentially powerful profiles and generate fake profiles based on only the available profiles. Naturally, we observe that the attack performance improves when the attacker has access to more normal profiles. This is reasonable as more profiles lead to more desirable recommendations, which further yields better attack effect. When we only have access to a subset of profiles rather than all, we find that in some cases RecUP even outperforms baseline attacks that have all ratings, compared with the results in Table IV. As the detection performance is similar with the results in Fig. 6, we do not show the results here.

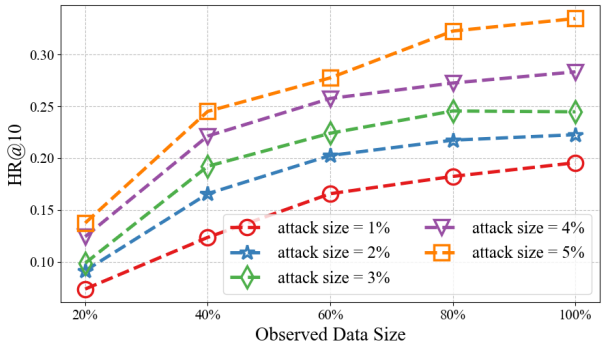


Fig. 9. Attack performance on ML-100K with different sizes of available profiles.

TABLE VI
ATTACK PERFORMANCE WITH CONDITIONS

DataSet	Attack Method			
	HRGAN +Gen	HRGAN_GP +Gen	cHRGAN +Gen	cHRGAN_GP +Gen
ML-100K	0.3327	**0.3452**	0.2945	0.3118
R&C	0.5148	**0.5314**	0.4314	0.4548

2) Impact of Extra Knowledge: In this section, we consider the case where the attacker knows extra user-specific attributes (such as gender and age, etc.). These attributes could help the generator to capture complex user-item associations better than without them. To add the attributes into our framework, we take it as the condition constraint when we train the generator networks (cHRGAN), in a similar manner as the condition GAN (cGAN) [55]. The attack and defense performance on ML-100K and R&C are shown in Table VI and Table VII, respectively.

TABLE VII
ATTACK DETECTION WITH CONDITIONS

DataSet	Detection Method								
	Attack Method	SemiSAD	PopSAD	SVM	RF	DT	GBDT	AdaBoost	GaussianNB
ML-100K	**HRGAN+Gen**	**0.246**	0.292	**0.254**	0.074	0.314	**0.198**	0.222	**0.188**
	HRGAN_GP+Gen	0.280	0.304	0.302	**0.106**	0.226	0.344	0.352	0.214
	cHRGAN+Gen	0.326	0.298	0.334	0.142	0.214	0.242	0.212	0.316
	cHRGAN_GP+Gen	0.248	**0.256**	0.324	0.114	**0.176**	0.236	**0.182**	0.250
R&C	**HRGAN+Gen**	0.398	**0.146**	0.376	0.234	0.146	**0.146**	0.300	0.334
	HRGAN_GP+Gen	0.514	0.460	0.646	0.280	0.214	0.214	0.374	0.420
	cHRGAN+Gen	**0.226**	0.278	0.330	**0.124**	**0.114**	0.260	0.280	0.332
	cHRGAN_GP+Gen	0.520	0.414	**0.266**	0.234	0.266	0.234	**0.272**	**0.314**

In general, we have better performance in Table VII and worse results in Table VI when we add the condition into the training. Compared with the results when we do not know user attributes, the additional condition leads to more plausible profiles. We believe that these attributes could help the generator to better capture complex user-item associations, thereby yielding better results in Table VII and though some reduction in attack effect. More 'normal' generated profiles are harder to detect, especially in the R&C dataset, whose profiles with 19 attributes far more than 4 attributes in ML-100K. That is why the difference made by the conditional results in the R&C dataset is more significant, as we can see in Table VII. The results suggest that better performance can be achieved if more knowledge (like users' attributes) of the target RS can be obtained.

VI. CONCLUSION

In this paper, we have presented RecUP, a novel poisoning attack framework that can craft plausible profiles to escape the detection. Our key idea is simulating the features of normal profiles based on a mini-max game with GAN. We additionally employ a shilling loss to achieve the attacker's desires. Instead of using all normal profiles to optimize ratings of fake profiles, we use a selected subset of potentially powerful profiles to further enhance the attack effect. We conduct extensive experimental studies to validate the effectiveness and superiority of RecUP, in terms of both the attack performance and the escape ability from detections. In the future, we plan to explore effective countermeasures against our attack.

REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, 2019.

[3] J. Jin *et al.*, "An efficient neighborhood-based interaction model for recommendation on heterogeneous graph," in *Proc. ACM KDD*, Aug. 2020, pp. 75–84.

[4] J.-T. Huang *et al.*, "Embedding-based retrieval in Facebook search," in *Proc. ACM KDD*, Aug. 2020, pp. 2553–2561.

[5] W. Krichene and S. Rendle, "On sampled metrics for item recommendation," in *Proc. ACM KDD*, Aug. 2020, pp. 1748–1757.

[6] A. Machanavajjhala, A. Korolova, and A. D. Sarma, "Personalized social recommendations: Accurate or private," *Proc. VLDB Endowment*, vol. 4, no. 7, pp. 440–450, Apr. 2011.

[7] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *Proc. ACM RecSys*, Sep. 2019, pp. 322–330.

[8] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-N recommender systems," in *Proc. WWW*, Apr. 2020, pp. 3019–3025.

[9] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, "Attacking recommender systems with augmented user profiles," in *Proc. ACM CIKM*, Oct. 2020, pp. 855–864.

[10] J. Song *et al.*, "PoisonRec: An adaptive data poisoning framework for attacking black-box recommender systems," in *Proc. IEEE ICDE*, Apr. 2020, pp. 157–168.

[11] H. Zhang, Y. Li, B. Ding, and J. Gao, "Practical data poisoning attack against next-item recommendation," in *Proc. WWW*, Apr. 2020, pp. 2458–2464.

[12] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *Proc. NDSS*, 2021, pp. 1–17.

[13] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proc. WWW*, 2004, pp. 393–402.

[14] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. NIPS*, 2016, pp. 1885–1893.

[15] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems," in *Proc. NDSS*, 2017, pp. 1–15.

[16] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proc. ACSAC*, Dec. 2018, pp. 381–392.

[17] Y. Deldjoo, T. Di Noia, and F. Antonio Merra, "Assessing the impact of a user-item collaborative attack on class of users," 2019, *arXiv:1908.07968*. [Online]. Available: http://arxiv.org/abs/1908.07968

[18] J. Cao, Z. Wu, B. Mao, and Y. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," *World Wide Web*, vol. 16, nos. 5–6, pp. 729–748, 2013.

[19] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, T.-S. Chua, and S. Ma, "Catch the black sheep: Unified framework for shilling attack detection based on fraudulent action propagation," in *Proc. IJCAI*, 2015, pp. 1–7.

[20] M. Aktukmak, Y. Yilmaz, and I. Uysal, "Quick and accurate attack detection in recommender systems through user attributes," in *Proc. ACM RecSys*, Sep. 2019, pp. 348–352.

[21] Z. Yang, Q. Sun, Y. Zhang, L. Zhu, and W. Ji, "Inference of suspicious co-visitation and co-rating behaviors and abnormality forensics for recommender systems," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2766–2781, 2020.

[22] Y. Zhang, J. Xiao, S. Hao, H. Wang, S. Zhu, and S. Jajodia, "Understanding the manipulation on recommender systems through web injection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3807–3818, 2020.

[23] R. A. Zayed, L. F. Ibrahim, H. A. Hefny, and H. A. Salman, "Shilling attacks detection in collaborative recommender system: Challenges and promise," in *Proc. IEEE AINA*, Apr. 2020, pp. 429–439.

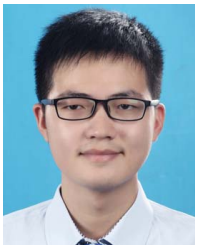
[24] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: A review," *Artif. Intell. Rev.*, vol. 53, no. 1, pp. 291–319, Jan. 2020.

[25] Y. Deldjoo, T. Di Noia, and F. A. Merra, "A survey on adversarial recommender systems: From attack/defense strategies to generative adversarial networks," 2020, *arXiv:2005.10322*. [Online]. Available: http://arxiv.org/abs/2005.10322

[26] B. Mehta and W. Nejdl, "Attack resistant collaborative filtering," in *Proc. ACM SIGIR*, 2008, pp. 75–82.

[27] C. E. Seminario and D. C. Wilson, "Attacking item-based recommender systems with power items," in *Proc. ACM RecSys*, 2014, pp. 57–64.

[28] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," *Beyond Pers.*, vol. 2005, no. 19, pp. 1–7, 2005.
[29] T. Dou, J. Yu, Q. Xiong, M. Gao, Y. Song, and Q. Fang, "Collaborative shilling detection bridging factorization and user embedding," in *Proc. CollaborateCom*, 2017, pp. 459–469.
[30] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proc. ACM KDD*, 2006, pp. 542–547.
[31] W. Zhou, Y. S. Koh, J. Wen, S. Alam, and G. Dobbie, "Detection of abnormal profiles on group attacks in recommender systems," in *Proc. ACM SIGIR*, Jul. 2014, pp. 955–958.
[32] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *User Model. User-Adapted Interact.*, vol. 19, nos. 1–2, pp. 65–97, 2009.
[33] H. Xia, B. Fang, M. Gao, H. Ma, Y. Tang, and J. Wen, "A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique," *Inf. Sci.*, vol. 306, pp. 150–165, Jun. 2015.
[34] W. Li, M. Gao, H. Li, J. Zeng, Q. Xiong, and S. Hirokawa, "Shilling attack detection in recommender systems via selecting patterns analysis," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 10, pp. 2600–2611, 2016.
[35] H. Cai and F. Zhang, "Detecting shilling attacks in recommender systems based on analysis of user rating behavior," *Knowl.-Based Syst.*, vol. 177, pp. 22–43, Aug. 2019.
[36] Z. Yang, L. Xu, Z. Cai, and Z. Xu, "Re-scale AdaBoost for attack detection in collaborative filtering recommender systems," *Knowl.-Based Syst.*, vol. 100, pp. 74–88, May 2016.
[37] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey," *Artif. Intell. Rev.*, vol. 42, no. 4, pp. 767–799, 2014.
[38] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.
[39] F. Alqadah, C. K. Reddy, J. Hu, and H. F. Alqadah, "Biclustering neighborhood-based collaborative filtering method for top-n recommender systems," *Knowl. Inf. Syst.*, vol. 44, no. 2, pp. 475–491, 2015.
[40] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, "FEXIPRO: Fast and exact inner product retrieval in recommender systems," in *Proc. ACM SIGMOD*, May 2017, pp. 835–850.
[41] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. WWW*, May 2015, pp. 111–112.
[42] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. WWW*, 2017, pp. 173–182.
[43] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.
[44] J. Wang *et al.*, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. ACM SIGIR*, Aug. 2017, pp. 515–524.
[45] H. Wang *et al.*, "Learning graph representation with generative adversarial nets," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3090–3103, Aug. 2021, doi: 10.1109/TKDE.2019.2961882.
[46] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J.-T. Lee, "CFGAN: A generic collaborative filtering framework based on generative adversarial networks," in *Proc. ACM CIKM*, Oct. 2018, pp. 137–146.
[47] P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang, "On the accuracy of influence functions for measuring group effects," in *Proc. NIPS*, 2019, pp. 5254–5264.
[48] T. Wang, J. Huan, and B. Li, "Data dropout: Optimizing training data for convolutional neural networks," in *Proc. IEEE ICTAI*, Nov. 2018, pp. 39–46.
[49] F. Yuan, L. Yao, and B. Benatallah, "Adversarial collaborative neural network for robust recommendation," in *Proc. ACM SIGIR*, Jul. 2019, pp. 1065–1068.
[50] H. Kanagawa, T. Suzuki, H. Kobayashi, N. Shimizu, and Y. Tagami, "Gaussian process nonparametric tensor estimator and its minimax optimality," in *Proc. ICML*, 2016, pp. 1632–1641.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.
[52] Y. Deldjoo, T. Di Noia, E. Di Sciascio, and F. A. Merra, "How dataset characteristics affect the robustness of collaborative recommendation models," in *Proc. ACM SIGIR*, Jul. 2020, pp. 951–960.
[53] S. Lu, L. Yu, S. Feng, Y. Zhu, and W. Zhang, "CoT: Cooperative training for generative modeling of discrete data," in *Proc. ICML*, 2019, pp. 4164–4172.
[54] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. NIPS*, 2017, pp. 5767–5777.
[55] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784



Xuxin Zhang received the B.E. degree from the University of Electronic Science and Technology of China, China, in 2019. He is currently pursuing the M.S. degree in electronics and information engineering with Huazhong University of Science and Technology, China. His research interests include data poisoning attacks and recommender systems.

Jian Chen (Student Member, IEEE) received the B.S. degree from Hubei University of Technology in 2014, and the M.S. degree from Huazhong University of Science and Technology, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Electronic Information and Communications. His research interests include machine learning and data privacy.

Rui Zhang (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Huazhong University of Science and Technology, China. From 2013 to 2014, she was a Visiting Scholar with the College of Computing, Georgia Institute of Technology, USA. She is currently an Associate Professor with the School of Computer Science and Technology, Wuhan University of Technology, China. Her research interests include machine learning, mobile computing, and data analytics.

Chen Wang (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Automation, Wuhan University, China, in 2008 and 2013, respectively. From 2013 to 2017, he was a Postdoctoral Research Fellow with the Networked and Communication Systems Research Laboratory, Huazhong University of Science and Technology, China. He joined the Faculty of Huazhong University of Science and Technology, where he is currently an Associate Professor. His research interests include wireless networking, the Internet of Things, and mobile computing, with a recent focus on privacy issues in wireless and mobile systems. He is a Senior Member of ACM.

Ling Liu (Fellow, IEEE) is currently a Professor with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA. She directs the research programs with the Distributed Data Intensive Systems Laboratory (DiSL). Her current research is primarily supported by USA National Science Foundation under CISE programs, IBM, and CISCO. She was a recipient of the IEEE Computer Society Technical Achievement Award in 2012 and the best paper award from numerous top venues. She served on editorial board of over a dozen international journals, including the Editor-in-Chief of IEEE TRANSACTIONS ON SERVICE COMPUTING from 2013 to 2016, and the Editor-in-Chief of *ACM Transactions on Internet Computing* since 2019.