IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 7, 2019, accepted February 24, 2019, date of publication March 1, 2019, date of current version March 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902042

Privacy-Aware Detection of Shilling Profiles on Arbitrarily Distributed Recommender Systems

BURCU YILMAZEL[1], ALPER BILGE[2], AND CIHAN KALELI[1], (Member, IEEE)
[1]Computer Engineering Department, Eskisehir Technical University, 26555 Eskisehir, Turkey
[2]Computer Research and Applications Center, Anadolu University, 26470 Eskisehir, Turkey

Corresponding author: Cihan Kaleli (ckaleli@eskisehir.edu.tr)

This work was supported by the Anadolu University under Grant 1403F069.

ABSTRACT Due to the mutual advantage of small-scale online service providers, they need to collaborate to deliver recommendations based on arbitrarily distributed preference data without jeopardizing their confidentiality. Besides privacy issues, parties also have concerns regarding the vulnerability against recommendation manipulation attempts, referred to as shilling attacks. Although there are methods for detecting these injected malicious profiles in central server-based configurations, they are not readily suitable for employing arbitrarily distributed data. In this paper, we present a novel classification-based shilling attack detection protocol enabling the recognition of malicious profiles in arbitrarily distributed configurations without compromising the privacy of collaborating parties. The analysis of the proposed protocol regarding confidentiality of parties reveals that the process is bound to collaboration by design, which does not allow parties to achieve detection by themselves. Furthermore, empirical evaluations using real-world preference data demonstrate that the protocol can achieve significantly high detection rates facilitating privacy-aware data collaboration.

INDEX TERMS Collaborative filtering, robustness, shilling, detection, arbitrary, partitioned data, privacy.

I. INTRODUCTION

Due to the explosive use of the Internet, the total amount of public digital information has been growing incessantly. According to business intelligence startup Domo's mind-blowing statistics for 2015, YouTube[1] users upload 300 hours of new video, Twitter[2] users send more than 347,000 tweets, Facebook[3] users like more than 4.1 million posts, Amazon[4] receives more than 4,310 unique visitors, and Vine[5] users play more than 1 million videos every minute. Besides proving the staggering increase in online data, these statistics also reveal the impossibility of human processing of mining data. Although such richness of data considered to be valuable, indeed, it puzzles individuals in discovering appropriate information and thus brings out the ''information overload problem'' [1]. In order to both cope with challenges of the explosion in the amount of data and simplify knowledge discovery, computer-aided personalization systems are developed, which consider tastes and demands of users [2]. Recommender systems are one of the most popular types of such systems that are studied extensively in research and industrial communities [1], [3]. During the last two decades, many techniques are developed to produce automated personalized referrals including collaborative, content-based, and knowledge-based filtering approaches [2].

Collaborative Filtering (CF) is the most popular recommendation technology that is employed by many e-commerce companies and online service providers such as Amazon[6] and Google News.[7] The basic ideas behind CF are (*i*) preferences of users' remain stable and consistent over time [4] and (*ii*) users who agreed in their past preferences will most probably have similar inclinations about new items in the future [5]–[7]. CF solely relies on preference data collected from customers on products and services. Therefore, to be able to provide accurate and dependable referrals, service providers need to collect a sufficient amount of data before launching recommendation services. Since the quality of

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Barhamgi.

[1]https://www.youtube.com/
[2]https://twitter.com
[3]https://www.facebook.com
[4]https://www.amazon.com
[5]https://vine.co
[6]https://www.amazon.com/
[7]https://news.google.com/

referrals is directly dependent on both the quantity and quality of collected data, e-commerce companies having lots of users are more likely to generate better recommendations compared to the ones with inadequate data. Moreover, poor recommendations that fail to meet users' expectations bring companies into disrepute and affect sales adversely. Thus, e-commerce companies aim to provide more dependable recommendations to disqualify their competitors. However, achieving such goal is not always possible for some e-companies, especially the newly established ones, due to lack of necessary and sufficient data, and it turns out to be a challenge [8], [9]. Companies overcome such shortage by collaborating with other companies with the aim of providing more meaningful recommendations based on the aggregated data [10]. However, there are risks and obstacles in establishing such cooperation. Since preference databases contain valuable preference information on consumer habits [4], collaborators hesitate to share the financial assets that enable them to satisfy their customers. In addition to these privacy issues, legal regulations complicate such collaboration, as well [11], [12].

Over the past few years, researchers have proposed Privacy-Preserving Distributed Collaborative Filtering (PPDCF) schemes that allow collaboration of two or more online recommendation services, even the competing ones [7], [13]–[15]. According to these studies, data collected for recommendation purposes might be distributed between two or more parties in three different data distribution scenarios, i.e., horizontal, vertical, or arbitrary. In horizontally distributed data (HDD) scenario, various parties hold disjoint sets of ratings for an identical set of items. Whereas, in vertically distributed data (VDD) scenario, different parties have disjoint sets of ratings collected from an identical set of users. Finally, in arbitrarily distributed data (ADD) scenario, which arises as the most probable to occur in the real world conditions, different parties hold distinct ratings for a shared set of users and items, and there is not a simple pattern of how ratings distribute between the parties [16].

A. PROBLEM DEFINITION

In addition to insufficient data, being subject to vicious attacks is yet another challenge faced by CF algorithms. Malicious users or competitor vendors, who compete unfairly by acting as a genuine user, might create fake user profiles and submit them to the central server aiming a bias in recommendations for their benefit [17], [18]. Such attempts are known as ''shilling'' or ''profile injection'' attacks [19], [20]. These attacks intend to manipulate recommendation outputs by either levitating or reducing (referred to as push and nuke attacks, respectively) popularity of targeted items depending on the aim of the attackers [18], [21]. Research orientation in this field includes generating appropriate attacking strategies and analyzing their effects on different CF algorithms, developing shilling attack detection strategies, and proposing robust algorithms resistant against shilling attacks [17].

The literature provides clear solutions regarding aforementioned fundamental problems of CF systems, i.e., (*i*) private protocol definitions allowing cooperation of data holders without jeopardizing corporate privacy toward insufficient data problem and (*ii*) centralized data-based shilling attack detecting solutions for improving the robustness of standalone CF systems. However, Yılmazel and Kaleli [22] show that PPDCF solutions are still vulnerable to shilling attacks since existing detection methods fall into abeyance when applied on distributed data. Despite protecting corporate privacy, authors demonstrated with experiments that state-of-the-art PPDCF schemes are defenseless against well-known shilling attack strategies. Existing solutions proposed for improving the robustness of centralized CF systems fail to be applied directly onto privacy-preserving ADD-based systems [22]. Therefore, mechanisms need to be developed that can enable detection of malicious profiles in ADD without compromising the privacy of collaborating companies in order for them to maintain cooperation.

B. CONTRIBUTIONS AND ORGANIZATION

In this study, we focus on detecting malicious user profiles aiming to manipulate the recommendations for particular items in PPDCF systems without jeopardizing the privacy of distributed parties. We propose the distributed version of a well-known classification-based attack detection method that can perform on ADD without sacrificing privacy, and we confirm that parties alone cannot successfully detect distributed attack profiles by themselves, especially having lower filler size. Main contributions of the study can be listed as follows:

1) Novel protocols that privately compute various classification attributes over ADD are proposed.
2) Private and distributed version of a well-known classification-based attack detection method for ADD is derived.
3) Established that employing the proposed approach, collaborating parties cannot successfully detect distributed shilling attack attempts on their own and are obliged to cooperate.

The rest of the paper is organized as follows: In Section II, the gap in the literature is manifested by reviewing PPDCF and shilling attack domains whereas, in Section III, related background information is given. Section IV presents proposed private protocols for the privacy-aware generation of classification attributes. Section V empirically evaluates the performance of proposed protocols through real-world data-based experiments. Finally, conclusions are given, and future works are discussed in Section VI.

II. RELATED WORK

Before diving in the distributed detection of shilling profiles in ADD scenarios, we provide a general overview of data distribution schemes, the robustness of recommendation algorithms, and detection of shilling profiles. It is very likely that a particular user's ratings to distribute among more than one parties due to different shopping histories. Such companies, particularly the ones having inadequate data, might collaborate to provide more qualified referrals.

However, corporate privacy is the biggest obstacle to such cooperation, since collected data is a valuable and confidential asset of a company. Besides confidentiality concerns based on legal regulations, companies might hesitate to collaborate due to financial fears [15], [23]. Various PPDCF schemes are proposed enabling data holders' collaboration without jeopardizing their privacy.

Polat and Du [9], [24] first discuss data distribution scenarios between two parties where authors introduce HDD- and VDD-based private binary top-N recommendation schemes to set an equilibrium among accuracy, privacy, and efficiency. They also focus on producing numerical predictions based on VDD [25]. Kaleli and Polat [26] discuss how to provide binary referrals on both HDD and VDD. Yakut and Polat [27] study a model-based distributed recommendation approach based on singular value decomposition.

Yakut and Polat [7], [28] are the first to emerge ADD-based numerical predictions. They also discuss binary ratings-based ADD scenarios and propose a privacy-preserving naïve Bayesian classifier-based solution [29]. In addition to the schemes based on two parties, researchers also motivate on enabling collaboration of more than two parties [8], [13], [14].

Mahony *et al.* [19] demonstrate the possibility of manipulating outputs of a CF system through the insertion of fake user profiles. Since then, several studies are focusing on attacking strategies, analyzing the robustness of existing CF systems, detection techniques to prevent attacks, and developing robust algorithms that are intrinsically resistant to attacks [17], [30]. In our previous work, we demonstrated that PPDCF algorithms running on ADD are severely vulnerable to shilling attacks [22]. Therefore, to employ ADD based solutions without robustness concerns, it is crucial to develop techniques for detecting shilling profiles on ADD.

Detection of shilling profiles can be performed using some methods, such as statistical techniques, classification, unsupervised clustering, and variable selection [18]. For the sake of clarity and relevance to the current study, we focus on detection by classification techniques. In these approaches, attack detection is considered as a classification/ranking problem trying to discriminate *Attack* profiles from *Authentic* ones based on detection attributes calculated for each profile. Burke *et al.* [31], [32] introduce a set of classification attributes based on expected characteristics of attack profiles, including generic and model-specific attributes, several of which is extended from attributes proposed initially in [33]. Mobasher *et al.* [34] introduce two more classification attributes that are particularly effective at detecting segment attacks. Also, Williams and Mobasher [35] examine the combined effects of proposed additional attributes with the existing ones. What efficacy of proposed classification attributes for attack detection is evaluated via well-known supervised classification algorithms, e.g., simple nearest-neighbor classification, decision-tree learning using C4.5, and Support Vector Machine (SVM) [21], [31], [32], [34]–[36]. Later, He *et al.* [37] classify and detect shilling attack profiles using rough set theory. Zhang and Zhou [38] apply ensemble learning approach in attack detection where the underlying idea is to improve predictive ability based on relearning existing knowledge. They proposed a meta-learning-based detection algorithm, which contains two training phases (i.e., base-level training and meta-level training), using SVM as the primary learning method. This method can efficiently detect profile injection attacks; however, it yields low precision for particularly small attack sizes. Zhang and Zhou [39] also propose an online method, namely HHT-SVM, for attack detection based on Hilbert-Huang transform (HHT) and SVM. The crucial point of the algorithm is HHT-based feature extraction method. To improve the success of attack detection, Morid *et al.* [40] applied a k-NN supervised classification method on influential users, instead of the whole user set. Zhang and Zhou [41] propose an ensemble detection model (EDM) by introducing backpropagation neural network and ensemble learning technique to detect profile injection attacks through selecting and integrating parts of the base classifiers using voting strategy. In another work, Zhang and Chen [42] illustrate the effectiveness of ensemble method for detecting shilling attacks based on ordered item sequences (EMDSA-OIS), which use simple majority voting strategy to combine the predictive results of multiple C4.5-based classifiers.

Although there are a few PPDCF algorithms on ADD that are proposed to enable collaboration of service providers to produce more accurate recommendations while preserving their privacy, they are shown to be too vulnerable to shilling attacks [22]. It is demonstrated that attackers being aware of the collaboration can insert distributed versions of various attack models, hence, manipulate outcomes of the state-of-the-art ADD-based PPCF schemes on their favor. On the other hand, proposed detection methods are designed for central data-based recommender systems and cannot be employed directly on ADD due to both structures of data and privacy concerns of data holders. In this study, we focus on computing classification attributes on ADD without jeopardizing data owners' privacy. Furthermore, we establish that even if parties run their own detection mechanisms, they are not able to detect distributed attacks, which makes collaboration mandatory.

III. BACKGROUND

A. ARBITRARILY DISTRIBUTED DATA

Jagannathan and Wright [16] introduce the idea of ADD and explain that there is not a straight-forward pattern of how preference data are distributed between two parties. In ADD model, two vendors combine their preference collections so that the aggregate holds the shared set of users and items in each of their datasets along with the ratings for these items. It is assumed that each rating belongs to either one of the parties and there are no overlapping votes [7]. Fig. 1 illustrates the ADD model set up between online vendors A and B, where black and striped cells in A's and B's datasets, respectively denote submitted votes and white cells the unrated items.

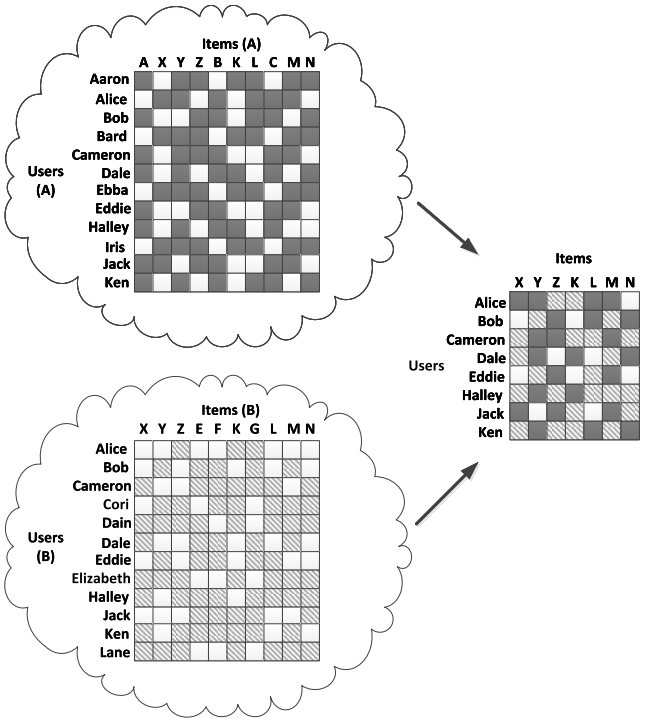

FIGURE 1. Arbitrarily distributed data model [22].

As can be followed in Fig. 1, votes given by same users in A and B to the set of common items form ADD between A and B under the assumption that there are no overlapping ratings between parties. The assumption implies a user can vote an item either in A or B, but not in both. Hence, ADD between A and B holds aggregated votes of intersecting users and items between datasets of A and B. Since it is not practical for a user to rate all available elements in a system, there also exist unrated cells. Indeed, a tiny fraction of cells in ADD contains a vote, which makes collaboration essential in the first place. ADD fits on real-world settings better than horizontal and vertical partitioning do. Produced protocols for ADD can be put into practical use for horizontal and vertical partitioning cases, as well [16].

B. SHILLING ATTACK MODELS

A profile injection attack against a recommender system aims to manipulate prediction outcomes of the system by inserting a set of malicious profiles about a single targeted item [31]. In general, the intent of shilling attacks is either to include an ordinary item in recommendation lists or to exclude an exceptional one. In other words, they either promote or demote a target item maliciously where promoting *push attacks* aim to increase the popularity of the target item and demoting *nuke attacks* aim the opposite [30].

An attack consists of several attack profiles to be injected into the system. The general form of an attack profile comprised of four disjoint sets is depicted in Fig. 2 [43]. I_S is the set of selected items with specific characteristics decided by the attacker to make the attack profile similar to those of users who prefer these particular group of products [35]. I_S receives

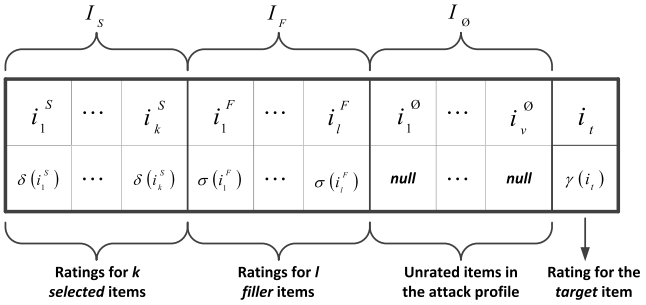

FIGURE 2. General form of an attack profile [17].

votes as determined by the rating function δ, whereas for some attack models it is empty. I_F is the set of filler items typically chosen randomly to complete the attack profile and to make it harder to detect. I_F receives votes provided by the rating function σ. i_t is the single target item whose popularity is manipulated by assigning a vote as determined by the rating function γ. Generally, such rating is either the maximum or minimum available vote in the system, i.e., r_{max} for *push* attacks and r_{min} for *nuke* attacks. The remaining items that have no rating compose the null portion of the attack profile indicated as $I_\emptyset$. Characteristics of an attack model are determined by the strategy used for identifying I_S and I_F, and the way the ratings are assigned to each of these sets including the target item, thus the rating functions δ, σ, γ.

In order to perform an attack, the attacker needs some information about the targeted system such as used recommendation algorithm and number of users/items [44]. While a *high-knowledge attack* requires a detailed information about the ratings distribution in the dataset, a *low-knowledge attack* requires system-independent information [21]. This study focuses detecting six popular and well-known shilling attack strategies; namely Random, Average, Bandwagon, Segment, Reverse Bandwagon, and Love/Hate attacks [30], [43], [45]–[47]. Attack profile generation guidelines for these attack types are summarized in Table 1 [22], [46].

C. CLASSIFICATION ATTRIBUTES FOR SHILLING ATTACK DETECTION

In classification-based shilling attack detection methods, classification attributes come in three diversities: generic, model-specific, and intra-profile. Generic attributes are the basic descriptive statistical features that try to capture the characteristics that make an attacker's profile look different from an official user profile. Attack model-specific attributes are generated to detect the features of a profile mainly associated with a specific attack model. Different from generic and model-specific attributes, intra-profile attributes concentrate on intra-profile statistics, in turn, they are designed to detect concentrations across profiles. The details of these classification attributes can be found in the following subsections based on the works in [21], [31], [32], [35], [36], [48], and [49]. Table 2 summarizes notations take part in the formulas of the attributes that were used throughout the paper.

TABLE 1. Attack profile summary [22].

Attack Type	I_S		I_F		I	i_t
	Items	Rating	Items	Rating		
Random	-	-	Randomly chosen	System mean	$I - I_F$	r_{max} / r_{min}
Average	-	-	Randomly chosen	Item mean	$I - I_F$	r_{max} / r_{min}
Bandwagon	Popular items	r_{max}	Randomly chosen	System mean	$I - (I_F \cup I_S)$	r_{max}
Segment	Segmented items	r_{max}	Randomly chosen	r_{min}	$I - (I_F \cup I_S)$	r_{max}
Reverse BW	Unpopular items	r_{min}	Randomly chosen	System mean	$I - (I_F \cup I_S)$	r_{min}
Love/Hate	-	-	Randomly chosen	r_{max}	$I - I_F$	r_{min}

TABLE 2. Notations.

Notation	Definition
U	Universe of all users
I	Universe of all items
n	Number of users in the system
m	Number of items in the system
$r_{u,i}$	Rating given by user u to item i
$\|I_u\|$	Number of items rated by user u
$\|U_i\|$	Number of users who rated item i
$\overline{r_i}$	Average rating of item i across all users
$W_{u,v}$	Similarity between users u and v calculated by Pearson's correlation coefficient
k	Number of nearest neighbors
$I_{u,v}$	The set of items co-rated items by users u and v
$\|I_{u,v}\|$	Size of the co-rated items between users u and v
d	Number of rating in common
$\overline{U}$	Average number of ratings per user

1) GENERIC ATTRIBUTES

The underlying assumption behind generic attributes is that the overall statistical signature of an attack profile will diverge significantly from that of an authentic profile. The rating assigned to the target item and the distribution of ratings among the filler items are the informers that cause such difference [21]. As many researchers have postulated [33], [44], [50], [51], it is improbable for an attacker to have complete knowledge of the ratings in a real system. Thus, profiles generated by malicious users often differ from rating patterns of authentic users. That is why an attribute that captures these irregularities can be descriptive in detecting attack profiles. There are a number of generic attributes [Rating Deviation from Mean Agreement (*RDMA*), Weighted Degree of Agreement (*WDA*), Weighted Deviation from Mean Agreement (*WDMA*), Degree of Similarity with Top Neighbors (*DegSim*), Degree of Similarity with Co-Rated Factor (*DegSim′*) and Length Variance (*LengthVar*)] proposed in the literature for shilling profile classification, whose equations are given in Table 3.

2) MODEL-SPECIFIC ATTRIBUTES

In [31] and [34], researchers have shown that when the profiles contain fewer filler items, generic attributes are insufficient to distinguish a true attack profile from eccentric, but authentic profiles, which results in significant false positive rate in classification. In order to reduce the success of the attacks, in addition to generic attributes, Williams *et al.* [49] proposed the use of model-specific attributes, which are based on partitioning each user profile somehow to maximize it's similarity to one created by a particular attack model. This partitioning can be modeled by splitting each user profile into three sets. The set $P_{u,T}$ contains all the items in user u's profile that is suspected to be targeted. According to the intent of the attack, alleged target items are the items that get either the profile's maximum rating (r_{max} for push attack), or the profile's minimum rating (r_{min} for nuke attack). The set $P_{u,F}$ contains all other ratings in user u's profile that is suspected to be filler items. The unrated items in user u's profile form the set $P_{u,\emptyset}$. The purpose is for $P_{u,T}$ to approximate $\{i_t\} \cup I_S$, for $P_{u,F}$ to approximate I_F, and $P_{u,\emptyset}$ will be equal to $I_\emptyset$. Hence, the statistical features

TABLE 3. Equations for generic attributes.

Generic Attributes

$$RDMA = \frac{\sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|}}{|I_u|} \quad (1)$$

$$WDA = \sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|} \quad (2)$$

$$WDMA = \frac{\sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|^2}}{|I_u|} \quad (3)$$

$$DegSim = \frac{\sum_{v \in neighbors(u)} w_{u,v}}{k} \quad (4)$$

$$DegSim' = \frac{\sum_{v \in neighbors(u)} w'_{u,v}}{k}$$
$$w'_{u,v} = w_{u,v} \times \frac{|I_{u,v}|}{d}, \quad \text{if } |I_{u,v}| < d \quad (5)$$

$$LengthVar = \frac{\big||I_u| - \overline{U}\big|}{\sum_{u=0}^{n} \big(|I_u| - \overline{U}\big)^2} \quad (6)$$

TABLE 4. Equations for average attack model-specific attributes.

Average Attack Model-Specific Attributes

$$P_{u,target} = \left\{ p_{target} \in P_u, such that r_{u,p_{target}} = r_{max} \right\}$$

$$MeanVar(u, p_{target}) = \frac{\sum_{i \in (P_u - (p_{target} \cup P_{u,}))} (r_{u,i} - \overline{r_i})^2}{|P_u - P_{u,}| - 1} = \frac{\sum_{i \in P_{u,F}} (r_{u,i} - \overline{r_i})^2}{|P_{u,F}|} \tag{7}$$

$$FMD = \frac{\sum_{i \in (P_u - (p_{target} \cup P_{u,}))} |r_{u,i} - \overline{r_i}|}{|P_u - P_{u,}| - 1} = \frac{\sum_{i \in P_{u,F}} |r_{u,i} - \overline{r_i}|}{|P_{u,F}|} \tag{8}$$

$$ProfileVar = \frac{\sum_{i \in P_{u,T} \cup P_{u,F}} (r_{u,i} - PMean)^2}{|P_{u,T} \cup P_{u,F}|}, \qquad where PMean = \frac{\sum_{i \in P_{u,T} \cup P_{u,F}} r_{u,i}}{|P_{u,T} \cup P_{u,F}|} \tag{9}$$

of these partitions can be used for generating the model-specific detection attributes. Reference [49] introduced several measures for detecting the distinctive signatures of attack models.

- *Average Attack Model-Specific Attributes* [35]:

According to the definition of average attack, each filler item will get ratings around the own mean of that filler item. In this case, optimal partitioning, which is the one where the mean-variance is minimized, is constituted through iterations over the items with extreme ratings. Let P_u be the profile of user u, and $P_{u,target}$ be the set of potential target items in P_u, which are given the rating r_{max} (or r_{min} according to the intend of the attack). *Mean Variance (MeanVar)* metric will be computed for each possible target item (p_{target}) from the set $P_{u,target}$ iteratively by taking one of them as suspected target, and assigning rest of the rated items as filler items. *MeanVar* metric is calculated as in Eq. 7, where $\left|P_u - P_{u,\emptyset}\right| - 1$ describes the number of rated items in user profile P_u, which is actually the size of the filler item set, $\left|P_{u,F}\right|$.

Among the calculated $MeanVar(u, p_{target})$ values, whichever yields the lowest value will be considered as the optimal partitioning, and p_{target} item that generates this minimum value will be selected as the target item t. Later, the $P_{u,T}$, and $P_{u,F}$ sets determined by this optimal partitioning will be used to create the following attributes: (1) *Filler Mean Variance (FMV)* is the minimum $MeanVar(u, p_{target})$ value that gives the optimal partitioning calculated in Eq. 7, (2) *Filler Mean Difference (FMD)* is the average of the absolute value of the difference between the user's rating, and the mean rating for the filler items calculated as in Eq. 8, and (3) *Profile Variance (ProfileVar)* that captures the within-profile variance calculated as in Eq. 9.

- *Random Attack Model-Specific Attributes* [32]:

Similar to average attack, the random attack is also a partitioning attack with the target partition being a single rating. However, unlike average attack, the filler items are assigned values generated randomly within the rating scale with a distribution centered around the mean for all user ratings across all items. Reference [35] proposed to use the correlation between a profile and the rating average for each item as a metric to discriminate random attackers. Hence, for random attack, the optimal partitioning is the one that gives the minimum correlation. The p_{target} item, which gives this minimum correlation is selected as the most likely target t, or as the partitioning set $P_{u,T}$, and all other rated items in P_u forms the set $P_{u,F}$. Obtained minimum correlation is called the *Filler Average Correlation (FAC)*, and used as a classification attribute in detecting random attackers. The other classification feature used in random attack detection is the *FMD* attribute [32], which is calculated as in Eq. 8 based on the discovered optimal partitioning sets $P_{u,T}$, and $P_{u,F}$ for random attack.

- *Group Attack Model-Specific Attributes* [49]:

Group attack attributes are proposed for detecting attacks that aim to increase the distinction of a targeted group of items ($i_t \cup I_S$), and the filler items (I_F), like the bandwagon and segment attack models. Therefore, the partitioning of the group attack model is done differently from the average and random attacks. In this model, all items in P_u that are given the maximum rating, r_{max}, (or the minimum rating, r_{min}, for nuke attack) in user u's profile are placed in the target partition, $P_{u,T}$, and all other rated items in P_u form the set $P_{u,F}$, which is the filler partition. Model-specific attributes for detecting both the bandwagon and segment attack models are calculated based on this partitioning.

$$\begin{aligned} P_{u,T} &= \left\{ i \in P_u, such that r_{u,i} = r_{max} \right\} \\ P_{u,F} &= P_u - (P_{u,T} \cup P_{u,\emptyset}) \end{aligned} \tag{10}$$

For bandwagon attacks, investigation of the filler ratings is identical to the random attack model. As in random attack, *FAC* and *FMD* attributes are generated, but in this case, the partitions given in Eq. 10 are used in calculations.

For segment attacks, *Filler Mean Target Difference (FMTD)* attribute, which intends to capture the difference between the average of the ratings in the target partition and the average of the ratings in the filler partition, is introduced. This feature is computed as follows:

$$FMTD = \left| \left(\frac{\sum_{i \in P_{u,T}} r_{u,i}}{\left|P_{u,T}\right|} \right) - \left(\frac{\sum_{k \in P_{u,F}} r_{u,k}}{\left|P_{u,F}\right|} \right) \right| \tag{11}$$

where i is from the set $P_{u,T}$, $r_{u,i}$ is the rating given by user u to item i, and $\left|P_{u,T}\right|$ is the number of ratings in the target partition $P_{u,T}$. Similarly, k is from the set $P_{u,F}$, $r_{u,k}$ is the rating given by user u to item k, and $\left|P_{u,F}\right|$ is the number of ratings in the filler partition $P_{u,F}$. The overall average $\overline{FMTD}$ calculated among all user profiles is then subtracted from $FMTD$ as a normalizing factor.

Group Filler Mean-Variance (GFMV) is another attribute used for detecting segment attacks. $GFMV$ is the variance of the filler items identified by the group attack partitioning, and is calculated as follows:

$$GFMV = \frac{\sum_{i \in P_{u,F}} \left(r_{u,i} - \overline{r_i}\right)^2}{\left|P_{u,F}\right|} \tag{12}$$

where $P_{u,F}$ is the set of items in the profile of user u that have been partitioned as filler items, $r_{u,i}$ is the rating user u has given to item i, $\overline{r_i}$ is the average rating of item i across all users, and $\left|P_{u,F}\right|$ is the number of ratings in the set $P_{u,F}$.

3) INTRA-PROFILE ATTRIBUTES

All of the classification attributes mentioned so far have concentrated on inter-profile statistics, which consider characteristics within a single profile [49]. In fact, a single profile cannot effect a recommender system significantly. Hence, the attackers need to inject multiple shilling profiles to cause a significant bias [35]. Williams *et al*. [36] introduced the *Target Model Focus (TMF)* attribute focusing on statistics across profiles, specifically concentrated on intra-profile statistics. Most probably, when a system is attacked, there will be several attack profiles that target the same item. Hence, TMF examines the density of target items. Since the partitioning associated with the model-specific attributes described above identifies the set of suspected targets for each user profile, using these partitions the TMF attribute calculates the degree to which the partitioning of a given user profile focuses on items common to other attack partitions, and thus measures a consensus of suspicion regarding each user profile [21].

In order to calculate TMF for a given user profile, F_i, the degree of focus on a given item is defined. Then, among the target set of a user profile, the item that has the highest focus is selected, and its focus value is used as a classification attribute. TMF of a user profile u is estimated as given in Eq. 13.

$$\begin{aligned} TMF &= \max_{j \in P_T} F_j, \quad \textit{where} \\ F_i &= \frac{\sum_{u \in U} \Theta_{u,i}}{\sum_{u \in U} \left|P_{u,T}\right|}, \quad \textit{and} \\ \Theta_{u,i} &= \begin{cases} 1, & \text{if } i \in P_{u,T} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

D. PRIVACY PROTECTION

Several privacy protection methods are introduced to achieve privacy in CF systems [15]. Among all, *Randomization* and *Cryptographic* techniques are the two most common approaches. In the following subsections, we first describe the methods utilized for preserving privacy, and then give the definition and analysis of privacy within the scope of this study.

1) UTILIZED METHODS

In this study, proposed private protocols utilize random filling (RF) method and Homomorphic Encryption (HE) [15]. Randomization is a privacy protection technique used for hiding or masking confidential data. Such methods aim at preserving privacy either by perturbing original ratings or by masking unrated items. Therefore, in order to disguise unrated items, RF method is employed [15] where collaborating companies selectively or uniformly randomly choose a number of their unrated items and fill them with unreal or default ratings. There are several options for generating such ratings, such as filling the unrated items with random numbers [52], or employing personalized ratings like user-mean and item-mean [7], [8].

Before initiating collaboration, the parties mask their part of data to protect the confidentiality of their data sets. For such purpose, parties make use of the method proposed by [7] for ADD, explained as follows:

1) Each part determines the density of users' in their own databases, which is the number of rated items in a user profile.
2) Parts uniformly randomly choose a value, θ, over the range $(1, \beta)$, where β is an experimental parameter indicating the density ratio.
3) Each part uniformly randomly selects θ percent of their empty cells, where θ is determined as stated by [7].
4) Parts fill such cells with unreal ratings and obtain their masked databases. Unreal ratings can be determined by following one of the strategies specified by [7].

Secure two-party computation enables two parties to jointly compute any function on their inputs without divulging to either party anything more than the correct output [53]. HE is a form of encryption that allows computation on ciphertexts while preserving the features of the function and format of the encrypted data [54]. As an example, for addition operation, suppose that ξ is an encryption function,

K is a public key, and plain-texts, x and y, are the private data values that will be encrypted. Hence, HE enables to compute $\xi_K(x+y)$ by using $\xi_K(x)$ and $\xi_K(y)$ without knowing x and y explicitly. In literature, most of the HE schemes operate on integer numbers. On the other hand, since there is a need for secure computations on floating-point numbers, there are a few HE schemes working on floating point numbers, as well. In this study, we employ Paillier cryptosystem [55] for addition operation of encrypted data and multiplication of an encrypted value with a constant. Also, for integer division a protocol proposed by Dahl *et al*. [56] is utilized. Finally, to perform floating-point-based calculations, floating-point homomorphic encryption (FPHE) scheme proposed by Cheon *et al*. [57] is used.

2) PRIVACY ANALYSIS

Privacy is a concept that is hard to define precisely, and it is complicated to specify its certain boundaries. In the context of CF, it has various meanings according to different viewpoints. In this study, privacy is defined as follows: *The parties should not be able to learn the actual rating values and the rated and/or unrated items held by each other during collaborative work*. Actual rating values are the opinions of the users on commercial products, and they can be used for profiling the customers. Hence, actual rating values and the list of rated items are considered confidential. E-companies are not willing to disclose their confidential data to others while performing collaborative tasks. Different from the actual rating values and the list of rated items held by a company, user and item IDs are regarded as public data. Thus sharing them would not contravene privacy.

Moreover, collaborative companies are assumed to be *semi-honest*, which means that they correctly apply the protocols, but they may try to gather as much private data as possible by resolving the inputs and outputs. Consequently, proposed protocols should enable cooperative work on ADD within boundaries of the specified confidentiality constraints.

In the RF method, data holders insert unreal ratings into their users' vectors. Thus, the privacy level of RF depends on how precisely the collaborating parties can guess the number of actual ratings and unreal ones in a user vector. Note that, in the proposed protocols, instead of a specific value in a user's vector, only aggregated computation results are exchanged between parties, which avoids occupation of actual rating values. On the other hand, if we assume that any particular user's masked rating vector is disclosed, attained privacy level can be estimated as stated by [7]. The probability of guessing the correct θ value is 1-out-of-β and probability of acquiring β is 1-out-of-100. The probability of guessing the exact positions of filled cells in a vector is $C_{r_a}^{r_f}$, i.e., $\frac{r_f!}{r_f!(r_a-r_f)!}$ where r_f represents the number of fake ratings and r_a shows all actual values send by the data holder. Consequently, the probability of guessing the set of rated items is $\left(100 \times \beta \times C_{r_a}^{r_f}\right)^{-1}$. The other utilized privacy method depends on HE, and according to proposed studies [55]–[57], the schemes are semantically secure for inference of input values.

IV. PRIVATE PROTOCOLS FOR ATTACK DETECTION ON ADD

In this section, we firstly describe the general data configuration structure, and then the steps in deriving the required classification attributes privately between two data holders. Then, we define the *Revised Private Mean Estimation* protocol that computes some essential primitive values, which are compulsory in the calculation of the classification attributes. Finally, we describe the *private protocols* that are built up to derive the required generic and model-specific classification attributes for each distributed profile collaboratively between two data holders.

A. DATA CONFIGURATION AND MODELLING

In ADD configuration, the ratings given by a user are indiscriminately distributed between two parties, which means while one party (PartA) has some fraction of the ratings, the other party (PartB) holds the remaining ones. In order to calculate basic primitives and classification attributes necessary for attack detection, they tend to agree on collaboration in certain conditions where they keep their confidentiality preserved. Since distributed computations require data sharing between the parties, the critical issue is to exchange as few as possible amounts of data. Therefore, such a collaboration scheme should not violate their data privacy and should force parties to keep the collaboration by depriving the other to obtain all the necessary information.

For this reason, parties agree to collaborate on specific terms where they each hold some part of the calculation results. These terms define which auxiliary part of any calculation will be held at any party, and should be in the form of sharing this information half-and-half. Therefore, such a sharing scheme allows each party to make calculations on their part and have their partial half of the result, which are then combined to construct the final result. We characterize this sharing scheme by representing partial calculations of these parties as even and odd parts. Thus, while one party holds partial calculation of a primitive or attribute obtained through even-indexed entities, the other one obtains for the odd-indexed ones. Note that, parties should agree on index of users while constructing ADD before collaboration. Also, remember that such representation of even- and odd-indexed partial calculations is arbitrary, and one could also use any other scheme to symbolize two mutually exclusive parts of a calculation.

B. REVISED PRIVATE MEAN ESTIMATION PROTOCOL

After masking their databases with unreal ratings, parts need to calculate the classification attributes necessary for shilling attack detection by collaborating. However, in order to calculate some of these classification attributes, a number of basic primitives, which are the crucial values in some essential calculations, namely the mean rating of a user and an item,

or the total number of ratings in the system provided by a user and for an item, need to be known.

In general, arithmetic mean can be computed as the sum of all the numbers in the series divided by the size of the series. Hence, $mean = sum/count$ is an example of algebraic measure that can be calculated by applying division operation to distributive measures *sum* and *count*. More specifically, *sum* and *count* can be computed by partitioning the data into smaller sets, computing partial measures for each subset, and finally merging them to obtain the final result [7].

Following this strategy, the mean rating provided for any item i, represented as $\overline{r_i}$, can be computed in a distributive manner between PartA and PartB as follows:

$$\overline{r_i} = \frac{\sum_{u \in U_i} r_{u,i}}{|U_i|} = \frac{\sum_{u \in U_i^A} r_{u,i} + \sum_{u \in U_i^B} r_{u,i}}{|U_i|^A + |U_i|^B} = \frac{ItemSum^A + ItemSum^B}{ItemCount^A + ItemCount^B} \tag{14}$$

where U_i^A and U_i^B represent user ratings residing in both parties, where $U_i^A \cup U_i^B = U_i$.

Similarly, the mean of votes provided by user u for all items, represented as $\overline{v_u}$, can be computed in a distributive manner between PartA and PartB as follows:

$$\overline{v_u} = \frac{\sum_{j \in I_u} v_{u,j}}{|I_u|} = \frac{\sum_{j \in I_u^A} v_{u,j} + \sum_{j \in I_u^B} v_{u,j}}{|I_u|^A + |I_u|^B} = \frac{UserSum^A + UserSum^B}{UserCount^A + UserCount^B} \tag{15}$$

where $v_{u,j}$ is the vote given by user u to item $j \in I_u$, I_u^A and I_u^B represent votes residing in both parts, where $I_u^A \cup I_u^B = I_u$.

As can be followed in Eq. 14 and Eq. 15, calculation of both user- and item-mean values are performed in a similar manner. Relying on this, we revise the *Private Mean Estimation* protocol [7] originally designed for estimating the user-average votes on ADD, to calculate both user- and item-mean values. Moreover, with some additional steps, it becomes possible to store some intermediate count values, which are necessary for the calculation of the classification attributes.

Employing the following protocol, the parties can estimate $\overline{r_i}(i = 1, 2, \ldots, n)$ and $\overline{v_u}(u = 1, 2, \ldots, m)$ values for all items and users in a distributive manner.

1) Both parties compute all partial *sum* and *count* values for all items and users based on their own databases.

TABLE 5. Distributed formulation of *RDMA*, *WDMA*, and *WDA* attributes.

RDMA Attribute

$$RDMA = \frac{\sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|}}{|I_u|} = \frac{\sum_{i=0}^{|I_u|^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|} + \sum_{i=0}^{|I_u|^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|}}{|I_u|} = \frac{WDA^A + WDA^B}{|I_u|} \tag{16}$$

WDA Attribute

$$WDA = \sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|} = \sum_{i=0}^{|I_u|^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|} + \sum_{i=0}^{|I_u|^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|} = WDA^A + WDA^B \tag{17}$$

$$= \sum_{i=0}^{|I_u|_{even}^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{even}} + \sum_{i=0}^{|I_u|_{odd}^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{odd}} + \sum_{i=0}^{|I_u|_{even}^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{even}} + \sum_{i=0}^{|I_u|_{odd}^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{odd}} = WDA_{even}^A + WDA_{odd}^A + WDA_{even}^B + WDA_{odd}^B \tag{18}$$

WDMA Attribute

$$WDMA = \frac{\sum_{i=0}^{|I_u|} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|^2}}{|I_u|} = \frac{\sum_{i=0}^{|I_u|^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|^2} + \sum_{i=0}^{|I_u|^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|^2}}{|I_u|} = \frac{WDMA^A + WDMA^B}{|I_u|} \tag{19}$$

$$= \frac{\sum_{i=0}^{|I_u|_{even}^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{even}^2} + \sum_{i=0}^{|I_u|_{odd}^A} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{odd}^2} + \sum_{i=0}^{|I_u|_{even}^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{even}^2} + \sum_{i=0}^{|I_u|_{odd}^B} \frac{|r_{u,i} - \overline{r_i}|}{|U_i|_{odd}^2}}{|I_u|} = \frac{WDMA_{even}^A + WDMA_{odd}^A + WDMA_{even}^B + WDMA_{odd}^B}{|I_u|} \tag{20}$$

2) PartA and PartB exchange odd- and even-indexed estimated sub-aggregates of their halves, respectively.
3) Both parties estimate user- and item-mean values for odd- or even-indexed users and items by performing the following steps:
   a) Each party calculates *sum* and *count* values for odd- or even-indexed users and items. Hence, PartA and PartB respectively holds even and odd partials of *UserSum*, *ItemSum*, *UserCount*, and *ItemCount* values.
   b) Both parties store partial *UserCount*, and *ItemCount* values to be used in the rest of the protocols.
   c) PartA estimates the user- and item-mean ratings with even indices, i.e., $\overline{v_{u_{even}}}$ and $\overline{r_{i_{even}}}$, while PartB with odd indices, i.e., $\overline{v_{u_{odd}}}$ and $\overline{r_{i_{odd}}}$.
4) Parts exchange the estimated user- and item-mean values. Hence, at the end of the protocol, each part ends up with $\overline{v_u}$ and $\overline{r_i}$ values.

Following the *Revised Private Mean Estimation* protocol, each part only sends computation results for even- or odd-indexed entities. Therefore, the companies cannot figure out the actual values and summation of the ratings during the protocol [7].

In the end, each part will have the following information: Both PartA and PartB learn $\overline{v_u}$ values for all users and $\overline{r_i}$ values for all items. In addition, PartA stores $|U_i|_{even}$ and $|I_u|_{even}$, while PartB stores $|U_i|_{odd}$ and $|I_u|_{odd}$ to be used in the calculation of the classification attributes necessary for shilling attack detection.

C. PRIVATE ESTIMATION OF RDMA, WDMA, AND WDA ATTRIBUTES

Examining the formulation of *RDMA*, *WDA*, and *WDMA* attributes given in Eq. 1, Eq. 2, and Eq. 3, note that calculation of *WDA* and *WDMA* attributes are very similar, and *RDMA* attribute can be calculated after obtaining *WDA* and dividing by $|I_u|$. Distributed formulations for these attributes are given in Eq. 16, Eq. 17, and Eq. 19. However, parties only hold partial data and share either even or odd parts of computations with each other. Keeping this in mind, parties can privately compute *RDMA*, *WDA*, and *WDMA* attributes in a distributed manner in a single protocol, as follows:

1) Each party calculates $|r_{u,i} - \overline{r_i}|$ values by themselves.
2) Having $|U_i|_{even}$ values, PartA computes WDA^A_{even} and $WDMA^A_{even}$ values for each user. Similarly, having $|U_i|_{odd}$ values, PartB computes WDA^B_{odd} and $WDMA^B_{odd}$ values.
3) Parties collaborate to calculate the remaining partial sum values as follows.
   a) PartA encrypts $|r_{u,i} - \overline{r_i}|$ values for the odd-indexed items on her side, and sends these encrypted values to PartB.
   b) PartB gets $1/|U_i|_{odd}$ and $1/|U_i|^2_{odd}$ exponent of the encrypted values as shown in Eq. 21 and Eq. 22.

$$\begin{aligned}
&\xi_K\left(|r_{u,i}-\overline{r_i}|\right)^{1/|U_i|_{odd}} \ mod\ n^2 \\
&\quad\equiv \xi_K\left(|r_{u,i}-\overline{r_i}| * \frac{1}{|U_i|_{odd}}\right) mod\ n \\
&\quad= \xi_K\left(\frac{|r_{u,i}-\overline{r_i}|}{|U_i|_{odd}}\right) \qquad (21)\\
&\xi_K\left(|r_{u,i}-\overline{r_i}|\right)^{1/|U_i|^2_{odd}} \ mod\ n^2 \\
&\quad\equiv \xi_K\left(|r_{u,i}-\overline{r_i}| * \frac{1}{|U_i|^2_{odd}}\right) mod\ n \\
&\quad= \xi_K\left(\frac{|r_{u,i}-\overline{r_i}|}{|U_i|^2_{odd}}\right) \qquad (22)
\end{aligned}$$

   c) Having the encrypted $\xi_K\left(\frac{|r_{u,i}-\overline{r_i}|}{|U_i|_{odd}}\right)$, and $\xi_K\left(\frac{|r_{u,i}-\overline{r_i}|}{|U_i|^2_{odd}}\right)$ values of each user, PartB multiplies these encrypted values to obtain their sum in encrypted form, which are $\xi_K\left(\sum_{i=0}^{|I_u|^A_{odd}} \frac{|r_{u,i}-\overline{r_i}|}{|U_i|_{odd}}\right)$ and $\xi_K\left(\sum_{i=0}^{|I_u|^A_{odd}} \frac{|r_{u,i}-\overline{r_i}|}{|U_i|^2_{odd}}\right)$ values, respectively, and sends them back to PartA.
   d) PartA decrypts them, and obtains WDA^A_{odd} and $WDMA^A_{odd}$ values. Besides, adds these values to the partial sum values obtained for the even-indexed items in Step 2. Hence, PartA computes WDA^A value by adding WDA^A_{even} and WDA^A_{odd}, and $WDMA^A$ value by adding $WDMA^A_{even}$ and $WDMA^A_{odd}$.
4) By switching roles between the parts, Steps 3a-3d are repeated once more to compute WDA^B and $WDMA^B$.
5) PartA sends WDA^A and $WDMA^A$ values of odd-indexed users to PartB, and PartB sends WDA^B and $WDMA^B$ values of even-indexed users to PartA.
6) By applying the followings, even-indexed users' *WDA* and *WDMA* attributes are known by PartA (WDA_{even}, $WDMA_{even}$), and odd-indexed users' *WDA* and *WDMA* attributes are known by PartB (WDA_{odd}, $WDMA_{odd}$).
   a) PartA sums up WDA^A values that she has for the even-indexed users with the WDA^B values that PartB sent in Step 5 to obtain *WDA* values of even-indexed users. Similarly, PartB obtains *WDA* values of odd-indexed users.
   b) The numerator of the *WDMA* attribute is computed in the same manner. PartA sums up $WDMA^A$ values she has for the even-indexed users with the $WDMA^B$ values PartB sent to obtain the numerator of the *WDMA* attribute of even-indexed users. In a similar manner, PartB obtains the numerator of the *WDMA* attribute of odd-indexed users. According to the formula of the *WDMA* attribute, the numerator should be divided by $|I_u|$. Thus, each party can compute

the *WDMA* attribute for their part. By knowing the numerator of the *WDMA* attribute of the even-indexed users, and the $|I_u|_{even}$ value, PartA computes $WDMA_{even}$. Similarly, PartB computes $WDMA_{odd}$.

7) Each part computes *RDMA* attribute for half of the users by themselves. PartA calculates $RDMA_{even}$ for the even-indexed users by dividing WDA_{even} by $|I_u|_{even}$. Likewise, PartB calculates $RDMA_{odd}$. Thus, at the end of this step, even-indexed users' *RDMA* attributes are known by PartA, and odd indexed users' *RDMA* attributes are known by PartB.

D. PRIVATE ESTIMATION OF DEGSIM ATTRIBUTE

DegSim attribute is based on the average similarity of a profile's top-k nearest neighbors. In order to calculate this attribute, as a first step, similarity values between a user and the remaining users need to be calculated. Then, from these similarity values, the highest k of them are chosen. For similarity computation, the *Private Similarity Computation (PrivateSims)* protocol proposed by [58], which is originally designed for distributed calculation of Pearson' correlation coefficient on ADD between two parties without jeopardizing privacy, is employed.

The proposed protocol for distributed calculation of *DegSim* attribute is as follows:

1) Each party computes their half of the similarities by applying the *PrivateSims* protocol, and among these similarity values locates the partial highest k of them.
2) Parties initiate collaboration to calculate *DegSim* attribute. For even-indexed users:
   a) PartB sends partial k highest similarity values to PartA.
   b) PartA merges these similarity values with the ones she owns.
   c) Among all, PartA chooses the highest k, adds them up, and divides the sum by k to obtain the *DegSim* attribute.
3) By switching roles, Steps 2a-2c are repeated for the remaining users. Hence, *DegSim* attribute of the odd-indexed users are obtained by PartB. At the end of the protocol, each part will have *DegSim* attribute for half of the users.

Different from *DegSim* attribute, $DegSim'$ attribute given in Eq. 5, takes into account the number of co-rated items between two users. However, finding co-rated items of two users whose ratings are distributed between two parts is costly. Besides, such calculation might disrupt privacy, too. Therefore, we do not utilize this attribute since privacy is the main concern.

E. PRIVATE ESTIMATION OF LENGTHVAR ATTRIBUTE

At the end of the *Revised Private Mean Estimation* protocol, PartA and PartB learns the number of ratings given by the even- and odd-indexed users, respectively. In order to calculate the *LengthVar* attribute, the average number of ratings per user, i.e., $\overline{U}$) value in Eq. 6 should be distributively computed as given in Eq. 23.

$$\overline{U} = \frac{\sum_{u=0}^{n} |I_u|}{n} = \frac{\sum_{u\in U_{even}} |I_u| + \sum_{u\in U_{odd}} |I_u|}{n} = \frac{count_{even} + count_{odd}}{n} \tag{23}$$

1) Each party calculates total number of ratings in their databases for even- or odd-indexed users by adding up rating counts.
2) PartA sends the total number of ratings for even-indexed users to PartB, and PartB sends the count for odd-indexed users to PartA.
3) Adding the count values, $count_{even} + count_{odd}$, and dividing this sum to n, each part computes the $\overline{U}$ value.
4) PartA calculates $\left||I_u|_{even} - \overline{U}\right|$ difference for even-indexed users, likewise, PartB calculates $\left||I_u|_{odd} - \overline{U}\right|$ for odd-indexed users.
5) According to Eq. 6, $\left||I_u| - \overline{U}\right|$ difference needs to be divided by $\sum_{u=0}^{n} \left(|I_u| - \overline{U}\right)^2$. This denominator can be calculated partially by each part. Thus, PartA calculates the partial sum for even-indexed users ($\sum_{u=0}^{U_{even}} \left(|I_u| - \overline{U}\right)^2$), and PartB calculates the partial sum for odd-indexed users.
6) Parts exchange the partial sum values. By adding them to their partial sum values, each part finds the denominator of the *LengthVar* attribute ($\sum_{u=0}^{n} \left(|I_u| - \overline{U}\right)^2$).
7) By dividing the absolute differences calculated in Step 4 to the denominator of the *LengthVar* attribute, PartA computes the *LengthVar* attribute of the even-indexed users, and PartB for the odd-indexed users.

F. PRIVATE ESTIMATION OF AVERAGE ATTACK MODEL-SPECIFIC ATTRIBUTES

To find the average attack model-specific attributes, namely *FMV*, *FMD*, and *ProfileVar*, described in Section III-C2, the optimal partitioning where the mean-variance is minimized needs to be discovered first.

Since each user profile is distributed between the parties, each part can construct the set of possible target items ($P^A_{u,T}$ for PartA and $P^B_{u,T}$ for PartB) by checking voted items rated by either maximum rating, r_{max}, or minimum rating, r_{min}, according to the intent of the attack. Then, in order to find the optimal partitioning, each part needs to calculate the $MeanVar\left(u, p_{target}\right)$ attribute iteratively for each of the items in their possible target item sets. Through *Revised Private Mean Estimation* protocol, both parts know the average rating for each item, $\overline{r_i}$ values. According to Eq. 7, parts need to calculate the square of the difference of the filler ratings, however, these filler ratings are distributed between parts. Hence, Eq. 7 can be calculated in a distributed manner as shown in Eq. 6.

TABLE 6. Distributed formulation of average attack model-specific attributes.

Average Attack Model-Specific Attributes
$$MeanVar\,(u, p_{target}) = \frac{\sum_{i\in P^A_{u,F}} (r^A_{u,i} - \overline{r_i})^2 + \sum_{i\in P^B_{u,F}} (r^B_{u,i} - \overline{r_i})^2}{\left\|P^A_{u,F}\right\| + \left\|P^B_{u,F}\right\|} \quad (24)$$
$$FMD = \frac{\sum_{i^A\in P^A_{u,F}} \left\|r^A_{u,i} - \overline{r_i}\right\| + \sum_{i^B\in P^B_{u,F}} \left\|r^B_{u,i} - \overline{r_i}\right\|}{\left\|P^A_{u,F}\right\| + \left\|P^B_{u,F}\right\|} = \frac{\sum_{i^A\in P^A_{u,F}} \left\|r^A_{u,i} - \overline{r_i}\right\| + \sum_{i^B\in P^B_{u,F}} \left\|r^B_{u,i} - \overline{r_i}\right\|}{|I_u| - 1} \quad (25)$$
$$ProfileVar = \frac{\sum_{i^A\in P^A_u - P^A_{u,}} \left(r^A_{u,i} - \overline{v_u}\right)^2 + \sum_{i^B\in P^B_u - P^B_{u,}} \left(r^B_{u,i} - \overline{v_u}\right)^2}{|I_u|} \quad (26)$$

More specifically, for calculating the *MeanVar* values of the possible target items in $P^A_{u,T}$, PartA can consider all the rated items in PartB along with the ones that were rated in her side, except the chosen target item, as her filler item set, $P^A_{u,F}$; and similarly PartB can consider all the rated items in PartA along with her $P^B_{u,F}$ set as her filler items in calculating the *MeanVar* values of the possible target items in $P^B_{u,T}$. Hence, by looking at all of their rated items, parts can compute $\sum_{i\in P_{u,F}}(r_{u,i} - \overline{r_i})^2$ value, and send this partial value to the other part. Iterating through the possible target item sets, each part now can calculate the numerator of the $MeanVar\,(u, p_{target})$ values for their part. Although only the numerator of the $MeanVar\,(u, p_{target})$ values are known, since the denominator is same for all, the minimum numerator value will also be the minimum *MeanVar* value for that part. Among the two values computed for each part, the minimum one will give the optimal partitioning for user u.

So far, the part that optimal partitioning relies is known. However, actual *MeanVar* attribute is not calculated yet, only the numerator is obtained. To calculate the *MeanVar* value, this numerator should be divided by the size of the filler items, $\left\|P^A_{u,F}\right\| + \left\|P^B_{u,F}\right\|$, which is one less than $|I_u|$ value calculated by the *Private Mean Estimation* protocol, indicating the target item. However, at the end of the *Private Mean Estimation* protocol, PartA knows $|I_u|_{even}$, and PartB knows $|I_u|_{odd}$. Depending on whether the part having the optimal partitioning has the corresponding $|I_u|$ value of the active user or not, either part can calculate the *FMV* attribute for each user by herself, or send the numerator for letting the other part to compute the attribute.

FMD and *ProfileVar* are the other attributes that need to be computed based on the optimal partitioning. These attributes can also be computed in a distributed manner while calculating the *FMV* attribute of any user. Eq. 8 can be calculated in a distributed manner as shown in Eq. 25. In calculating *ProfileVar*, unlike *FMD* and *MeanVar*, the possible target item is also taken into consideration. Therefore, $i \in P_{u,T} \cup P_{u,F}$ is the same as $i \in P_u - P_{u,\emptyset}$. Hereafter, $\left\|P_{u,T} \cup P_{u,F}\right\|$ in the denominator is actually equals to $\left\|P_{u,F}\right\| + \left\|P_{u,T}\right\|$. Since, $\left\|P_{u,T}\right\|$ is 1, it turns out to be $\left\|P_{u,F}\right\| + 1$, which is equal to $|I_u|$ value. Hence, Eq. 9 can be written in a distributed manner as shown in Eq. 26.

Parties can compute *FMV*, *FMD* and *ProfileVar* attributes privately and in a distributed manner with a single protocol. For each user u the following steps will be performed:

1) Each part creates their possible target item set by examining user u's partial profile that they have. Hence, PartA constructs $P^A_{u,T}$, and PartB constructs $P^B_{u,T}$.
2) Parts compute $\sum_{i\in P_{u,F}}(r_{u,i} - \overline{r_i})^2$ value by considering all the rated items they have for their part, and exchange these values with each other. Thus, PartA computes $\sum_{i\in P^A_{u,F}}(r^A_{u,i} - \overline{r_i})^2$ value, let us name it as $MeanVarNumerator^A$, and sends it to PartB. Similarly, PartB computes $MeanVarNumerator^B$, and sends to PartA.
3) Iterating through the possible target item set, each part calculates the numerator of the $MeanVar\,(u, p_{target})$ values in their part.
   a) For each element in $P^A_{u,T}$, PartA treats it as a suspected target item, and finds the numerator of the $MeanVar\,(u, p_{target})$ values.
      i) Construct the $P^A_{u,F}$, which is the rest of the rated items in user u's profile owned by PartA plus the ones in PartB.
      ii) Calculate $\sum_{i\in P^A_{u,F}}(r^A_{u,i} - \overline{r_i})^2$, which is the $MeanVarNumerator^A$ value for the corresponding suspected target item.
      iii) Add the obtained $MeanVarNumerator^A$ to $MeanVarNumerator^B$ value that PartB sent in Step 2 to calculate the numerator of the $MeanVar\,(u, p_{target})$ value of the suspected target item.
   b) Among the calculated values choose the minimum one as the partially optimal partitioning according to PartA.

4) Similarly, by switching sides and following Steps 3a-3b, PartB learns the partially optimal partitioning for her part.
5) Parts exchange these minimum values, and both learn whether the optimal partitioning is on their part or not. The part that has the optimal partitioning increases the TMF value of the target item, which gives this partitioning, by 1.
6) *FMV*, *FMD* and *ProfileVar* attributes are computed.
   - If optimal partitioning is in the part who knows the $|I_u|$ value of the user:
     a) Part calculates the *FMV* attribute by herself.
     b) For calculating *FMD* attribute, the other part calculates $\sum_{i \in P_{u,F}} \left| r_{u,i} - \overline{r_i} \right|$ partial value for her filler item set, which include all the items that have rating on that particular part, and sends such value. The part having optimal partitioning, calculates her partial sum value for her filler item set, which include all the items that have rating on that particular part, except the target item. Then, adds up these partial sums, and divides the sum by $|I_u| - 1$ to obtain the *FMD* attribute.
     c) For calculating *ProfileVar* attribute, the other part calculates $\sum_{i \in P_{u,T} \cup P_{u,F}} (r_{u,i} - \overline{v_u})^2$ partial value for all the rated items that she has, and sends this value. The part having the optimal partitioning similarly calculates a partial sum value on all the rated items in her part, including the target item. Then, adds up these partial sums, and then divides by $|I_u|$ to obtain the *ProfileVar* attribute.
   - If the optimal partitioning is in one part, but the $|I_u|$ value is known by the other part:
     a) Part having the optimal partitioning sends the numerator value she has to the other part. Knowing the $|I_u|$ value of the user, other part calculates the *FMV* attribute for that user. This part cannot know which item is the possible target, but can only learn the value of the attribute.
     b) For calculating *FMD* and *ProfileVar* attributes, Steps 6b-6c are repeated by switching the roles between the parts.

At the end of this protocol, as well as learning the *FMV* attribute, parts also learn the value of *FMD*, and *ProfileVar* attribute for half of the users. This protocol needs to be computed twice; once for push attack, where the set $P_{u,target}$ contains items rated by r_{max}, and once for nuke attacks, where the set $P_{u,target}$ contains items rated by r_{min}. Hence, the corresponding attributes for nuke attacks can be computed with the above protocol by choosing the minimum ratings of the profile as the target set.

G. PRIVATE ESTIMATION OF BANDWAGON ATTACK MODEL-SPECIFIC ATTRIBUTES

For bandwagon attack, *FAC*, and *FMD* attributes need to be generated, as in random attack. However, different from random attack, these attributes need to be computed based on the partitioning shown in Eq. 10. Hence, all items in P_u that are given the maximum rating, r_{max}, (or the minimum rating, r_{min}, for nuke attack) in each user u's profile are placed in the target partition, $P_{u,T}$, and all other rated items form the set $P_{u,F}$, which is the filler partition. After forming these partitions, *FAC* and *FMD* attributes need to be calculated privately in collaboration of both parts.

FAC attribute captures the correlation between the ratings given to filler items and the average ratings of these items. Since both parts know average ratings of items, this correlation value can be calculated distributively between two parts, as shown in Eq. 27.

$$\begin{aligned} FAC &= \frac{\mathbf{x} \cdot \mathbf{y}'}{\sqrt{\mathbf{x} \cdot \mathbf{x}'} \cdot \sqrt{\mathbf{y} \cdot \mathbf{y}'}} \\ &= \frac{\mathbf{x^A} \cdot \mathbf{y^{A'}} + \mathbf{x^B} \cdot \mathbf{y^{B'}}}{\sqrt{\mathbf{x^A} \cdot \mathbf{x^{A'}} + \mathbf{x^B} \cdot \mathbf{x^{B'}}} \cdot \sqrt{\mathbf{y^A} \cdot \mathbf{y^{A'}} + \mathbf{y^B} \cdot \mathbf{y^{B'}}}} \end{aligned} \quad (27)$$

where, $\mathbf{x^A}$ and $\mathbf{x^B}$ represent the vectors having ratings given to filler items in PartA and PartB, respectively. Also, $\mathbf{y^A}$ and $\mathbf{y^B}$ are the vectors containing average ratings of the corresponding items, in order for PartA and PartB.

Distributed calculation of the *FAC* attribute given in Eq. 27 can be obtained privately for bandwagon attack model.

For each user u the following steps will be performed:

1) PartA constructs the filler item and target item sets of user u based on her own data by herself.
   a) PartA forms the set $P^A_{u,T}$ and increases the TMF value of these items by 1.
   b) PartA forms the set $P^A_{u,F}$ which is actually the $\mathbf{x^A}$ vector.
2) PartA forms $\mathbf{y^A}$ vector, which contains the corresponding item mean values of the filler items.
3) PartA calculates $\mathbf{x^A} \cdot \mathbf{y^A}^T$, $\mathbf{x^A} \cdot \mathbf{x^A}^T$, and $\mathbf{y^A} \cdot \mathbf{y^A}^T$.
4) Steps 1-3 are repeated by PartB.
5) For half of the users (if the user is odd-indexed), the following steps are performed:
   a) PartA sends the value of $\mathbf{y^A} \cdot \mathbf{y^A}^T$ to PartB. By using $\mathbf{y^B} \cdot \mathbf{y^B}^T$ value that she has, PartB calculates $\sqrt{\mathbf{y^A} \cdot \mathbf{y^A}^T + \mathbf{y^B} \cdot \mathbf{y^B}^T}$.
   b) PartB sends the value of $\mathbf{x^B} \cdot \mathbf{x^B}^T$ to PartA. By using $\mathbf{x^A} \cdot \mathbf{x^A}^T$ value that she has, PartA calculates $\sqrt{\mathbf{x^A} \cdot \mathbf{x^A}^T + \mathbf{x^B} \cdot \mathbf{x^B}^T}$. Then, encrypts it with her key and sends encrypted result to PartA.
   c) PartA gets the $\sqrt{\mathbf{x^A} \cdot \mathbf{x^A}^T + \mathbf{x^B} \cdot \mathbf{x^B}^T}$ exponent of the encrypted value sent by PartB in Step 5a to obtain the encrypted value of their product, which is the denominator of *FAC*.

d) According to Eq. 27, the numerator of *FAC* is the sum of $\mathbf{x^A} \cdot \mathbf{y^{A'}}$ and $\mathbf{x^B} \cdot \mathbf{y^{B'}}$. However, the former is known by PartA while latter is known by PartB. Then, PartB sends $\xi_K\left(\mathbf{x^B} \cdot \mathbf{y^{B'}}\right)$ to PartA.
e) PartA computes $\xi_K\left(\mathbf{x^A} \cdot \mathbf{y^{A'}} + \mathbf{x^B} \cdot \mathbf{y^{B'}}\right)$.
f) Now, PartA has both the numerator and the denominator value of the *FAC* attribute in encrypted form. PartA performs the division, and sends the result to PartB.
g) PartB decrypts the result and obtains the *FAC* attribute of the user.

6) By switching roles in Step 5, PartA obtains the *FAC* attribute of the remaining users.

Distributed calculation of *FMD* attribute given in Eq. 25 can be obtained privately for bandwagon attack model as follows:

For each user u the following steps will be performed:

1) Each part constructs the filler item and target item sets of user u according to their data.
   a) By constructing the filler sets, each part also learns its size. Thus, PartA forms the set $P^A_{u,F}$ and learns $\left|P^A_{u,F}\right|$ and PartB forms the set $P^B_{u,F}$ and learns $\left|P^B_{u,F}\right|$.
   b) Each part increases the TMF value of the items in their target item list by 1.
2) Parts calculate the partial sum values $\sum_{i^A \in P^A_{u,F}} \left|r^A_{u,i} - \overline{r_i}\right|$ and $\sum_{i^B \in P^B_{u,F}} \left|r^B_{u,i} - \overline{r_i}\right|$ based on their own filler item sets.
3) For half of the users (if the user is even-indexed), the following steps are performed:
   a) PartA encrypts her partial sum value and filler item set size, $\left|P^A_{u,F}\right|$, and sends these encrypted values to PartB.
   b) PartB encrypts her partial sum value and filler item set size, $\left|P^B_{u,F}\right|$.
   c) PartB multiplies these encrypted partial sum values and obtains sum of them in encrypted form. Similarly, by multiplying the encrypted size values, PartB obtains the sum of the sizes (count) in encrypted form.
   d) PartB divides encrypted sum to encrypted count, and sends the result to PartA.
   e) PartA decrypts the result and gets the *FMD* attribute of the user.
4) By switching roles in Step 3, PartB obtains the *FMD* attribute of the remaining users.

By choosing the minimum ratings of the profile as the target set, *FAC* and *FMD* attributes can be computed for nuke intent attacks with the above protocols. Moreover, these protocols can also be used in calculating the required random attack model-specific attributes. The only difference is the operated target and filler item sets. Therefore, these protocols need to be computed in a loop and the partitioning that gives the minimum *FAC* value is chosen as the optimal partitioning. Then, *FAC* and *FMD* attributes calculated based on the corresponding partitioning will be taken as the random attack model-specific attributes.

H. PRIVATE ESTIMATION OF SEGMENT ATTACK MODEL-SPECIFIC ATTRIBUTES

For segment attacks, *FMTD* and *GFMV* attributes need to be generated based on the partitioning shown in Eq. 10. Both the *GFMV* attribute and the *FMTD* attribute, which intend to capture the difference between average of the ratings in target partition and average of the ratings in filler partition, can be calculated in a distributed manner between PartA and PartB as shown in Eq. 28 and Eq. 29, respectively.

$$GFMV = \frac{\sum_{i^A \in P^A_{u,F}} \left(r^A_{u,i} - \overline{r_i}\right)^2 + \sum_{i^B \in P^B_{u,F}} \left(r^B_{u,i} - \overline{r_i}\right)^2}{\left|P^A_{u,F}\right| + \left|P^B_{u,F}\right|} \tag{28}$$

$$FMTD = \left| \frac{\sum_{k^A \in P^A_{u,T}} r^A_{u,k} + \sum_{k^B \in P^B_{u,T}} r^B_{u,k}}{|P^A_{u,T}| + |P^B_{u,T}|} - \frac{\sum_{i^A \in P^A_{u,F}} r^A_{u,i} + \sum_{i^B \in P^B_{u,F}} r^B_{u,i}}{|P^A_{u,F}| + |P^B_{u,F}|} \right| \tag{29}$$

For each user u the following steps will be performed:

1) Each part constructs her filler item and target item sets according to their data and increases the TMF value of the items in their target item list by 1.
2) Each part calculates some fundamental partial sum values based on their data. PartA calculates $\sum_{k^A \in P^A_{u,T}} r^A_{u,k}$, $\left|P^A_{u,T}\right|$, $\sum_{i^A \in P^A_{u,F}} r^A_{u,i}$, $\left|P^A_{u,F}\right|$, and $\sum_{i^A \in P^A_{u,F}} \left(r^A_{u,i} - \overline{r_i}\right)^2$. Similarly, PartB calculates these values based on her data.
3) For half of the users (if the user is even-indexed), the following steps are performed:
   a) PartA encrypts her fundamental partial sum values obtained in Step 2 and sends them to PartB.
   b) PartB encrypts her fundamental partial sum values.
   c) PartB multiplies $\xi_K\left(\sum_{k^A \in P^A_{u,T}} r^A_{u,k}\right)$ with $\xi_K\left(\sum_{k^B \in P^B_{u,T}} r^B_{u,k}\right)$ and obtains $\xi_K\left(\sum_{k^A \in P^A_{u,T}} r^A_{u,k} + \sum_{k^B \in P^B_{u,T}} r^B_{u,k}\right)$.
   d) PartB multiplies $\xi_K\left(\left|P^A_{u,T}\right|\right)$ with $\xi_K\left(\left|P^B_{u,T}\right|\right)$ and obtains $\xi_K\left(\left|P^A_{u,T}\right| + \left|P^B_{u,T}\right|\right)$.
   e) PartB multiplies $\xi_K\left(\sum_{i^A \in P^A_{u,F}} r^A_{u,i}\right)$ with $\xi_K\left(\sum_{i^B \in P^B_{u,F}} r^B_{u,i}\right)$ and obtains $\xi_K\left(\sum_{i^A \in P^A_{u,F}} r^A_{u,i} + \sum_{i^B \in P^B_{u,F}} r^B_{u,i}\right)$.

f) PartB multiplies $\xi_K\left(\left|P^A_{u,F}\right|\right)$ with $\xi_K\left(\left|P^B_{u,F}\right|\right)$ and obtains $\xi_K\left(\left|P^A_{u,F}\right| + \left|P^B_{u,F}\right|\right)$.

g) PartB divides encrypted sum value obtained in Step 3c to encrypted count value found in Step 3d.

h) PartB divides encrypted sum value obtained in Step 3e to encrypted count value found in Step 3f.

i) PartB subtracts the value obtained in Step 3h from the value obtained in Step 3g by first multiplying the result in Step 3g with -1, and then multiplying with the result in Step 3h to get their sum [59], and sends the result to PartA.

j) PartB multiplies $\xi_K\left(\sum_{i^A \in P^A_{u,F}} \left(r^A_{u,i} - \overline{r_i}\right)^2\right)$ with $\xi_K\left(\sum_{i^B \in P^B_{u,F}} \left(r^B_{u,i} - \overline{r_i}\right)^2\right)$ and obtains their encrypted sum.

k) PartB divides encrypted sum value obtained in Step 3j to encrypted count value found in Step 3f, and sends the result to PartA.

l) PartA decrypts the results and obtains *FMTD* and *GFMV* attributes of the user.

4) By switching roles in Step 3, PartB obtains the *FMTD* and *GFMV* attributes of the remaining users.

V. EXPERIMENTAL EVALUATION

In this section, we present experiments performed on real-data to demonstrate the detection performance of distributed classification approach.

A. DATA SET AND EVALUATION CRITERIA

In the experiments, we employed publicly available MovieLens 100K[8] (MLP) dataset, which consists of 100,000 5-star ratings on 1,682 movies by 943 genuine users where each user rates at least 20 movies. In order to simulate an ADD configuration, ratings are randomly distributed over two parties with the following strategy: For each user, a random distribution rate, $r \in (0, 100)$, is chosen and randomly selected $r\%$ of their existing ratings are transferred to PartA and the remaining ones to PartB. This way, the dataset is split into two sets so that genuine ratings of each user are randomly shared between parties with varying rates. Hence, an ADD configuration is obtained.

We measure classification performance by *Recall* and *Precision* metrics given in Eq. 30 and Eq. 31, where *Recall*, also known as *Sensitivity*, measures the number of attack profiles correctly classified as a fraction of the total attack profiles and *Precision*, also known as *Positive Predictive Value* (PPV), indicates the number of attack profiles correctly classified as a fraction of the number of profiles labeled as an attack [30].

$$Recall \equiv Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative} \tag{30}$$

$$Precision \equiv PPV = \frac{TruePositive}{TruePositive + FalsePositive} \tag{31}$$

B. EXPERIMENTAL METHODOLOGY

In this section, we present methodologies for two categories of experiments performed to demonstrate the motivation behind collaboration regarding attack detection and how collaborating parties can achieve decent detection rates by employing the proposed private protocols.

The general experimentation methodology for both types of experiments includes (*i*) formation of the training and test sets, (*ii*) creation of attack profiles for both training and tests individually, (*iii*) construct a classifier model based on the attacked training set, and (*iv*) testing the accuracy of the classifier using attacked test set [49]. For all experiments, one-third of the dataset is reserved as the test set and the remaining as the training set. Then, attack profiles for varying attack types and attacking parameters are generated based on the training and test sets separately. Following the profile creation, classification attributes are calculated based on the aggregation of the training set and corresponding attack profiles for each attack type. Therefore, a classifier model is created comprising of features for genuine users and each attack profile types. Finally, the success of the created model against each attack type is assessed by inserting corresponding fake profiles into the test set. For compatibility with reported results in the literature, we employed a k-NN classifier with a neighborhood size of 9 [31].

During the experiments, filler size parameter, I_F, is varied from 3 to 60%, whereas attack size parameter, the number of inserted fake profiles, is kept constant at 1% since it is unreasonable to insert more in real-world cases [40]. Also, the attacked movies, i_t, in the training set is chosen at random among the ones which have 80 to 100 ratings. Note that, training and testing processes for segment attack are performed through attack profiles based on movies starring Harrison Ford (Harrison Ford segment) and favorite horror movies (Horror segment), respectively.

1) EXP1: HOW SUCCESSFUL CAN THE PARTIES ALONE BE IN IDENTIFYING DISTRIBUTED ATTACK PROFILES?

In this experiment, it is assumed that PartA and PartB employ any PPDCF scheme on ADD to provide predictions, but they do not collaborate for attack detection. Instead, as a defense to shilling attacks, each part utilizes the classification-based attack detection method [49] to their partial data. However, the attack profiles are generated in a distributed manner and inserted into both parties datasets.

Under these circumstances, the question is whether PartA and PartB can detect the distributed attack profiles on their datasets individually. More specifically, if distributed attack profiles are injected into the system by a

[8] http://www.grouplens.org/datasets/movielens

malicious user who is aware of the cooperated recommendation scheme, how successful is the detection algorithm in identifying the distributed attack profiles for both parties independently?

In order to make inferences on independent success rates, we compare obtained results with outcomes of the classification-based attack detection method introduced by Williams *et al*. [49], which is referred to as *Baseline* in the experiments. For the case of *Baseline*, general experimentation methodology is employed on entire MLP, and results reported in [49] are reproduced.

To investigate the research question, we also employed the same general experimentation methodology separately on each individual part of ADD, i.e., PartA and PartB. Fig. 3 demonstrates train and test processes carried out by each party, where *MLP-A* and *MLP-B* represent an arbitrarily distributed case of MLP between PartA and PartB, respectively. In formation of the train and test sets, each party employs the same users with the ones in *Baseline* experiments. Also note that, similar to *Baseline*, attack profiles are generated based on entire MLP and injected into *MLP-A* and *MLP-B* disjoint datasets in distributed manner.

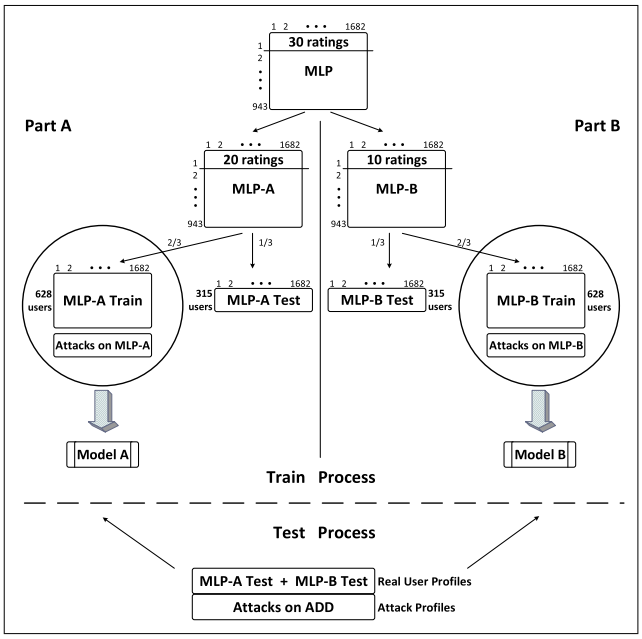

FIGURE 3. Methodology for experiment 1.

The classification attributes utilized in training classifiers are as follows [49]: Generic attributes (WDMA-RDMA-WDA-LengthVar-DegSim-DegSim'), average attack model-specific attributes (FMV-FMD-ProfileVar), random attack model-specific attributes (FMD-FAC), group attack model-specific attributes for Bandwagon attack (FMD-FAC), group attack model-specific attributes for Segment attack (FMTD-GFMV), and a target detection model-specific attribute (TMF).

2) EXP2: HOW SUCCESSFUL CAN COLLABORATING PARTIES BE IN IDENTIFYING DISTRIBUTED ATTACK PROFILES?

In this experiment, it is assumed that PartA and PartB cooperate to provide predictions by employing any PPDCF scheme on ADD, and they further collaborate for attack detection by employing the proposed distributed classification-based detection algorithm on ADD. Fig. 4 demonstrates data masking, collaborative training, and test processes carried out by parties, where *MLP-A* and *MLP-B* represent an arbitrarily distributed case of MLP, and *MLP-A′* and *MLP-B′* respectively denote their masked versions.

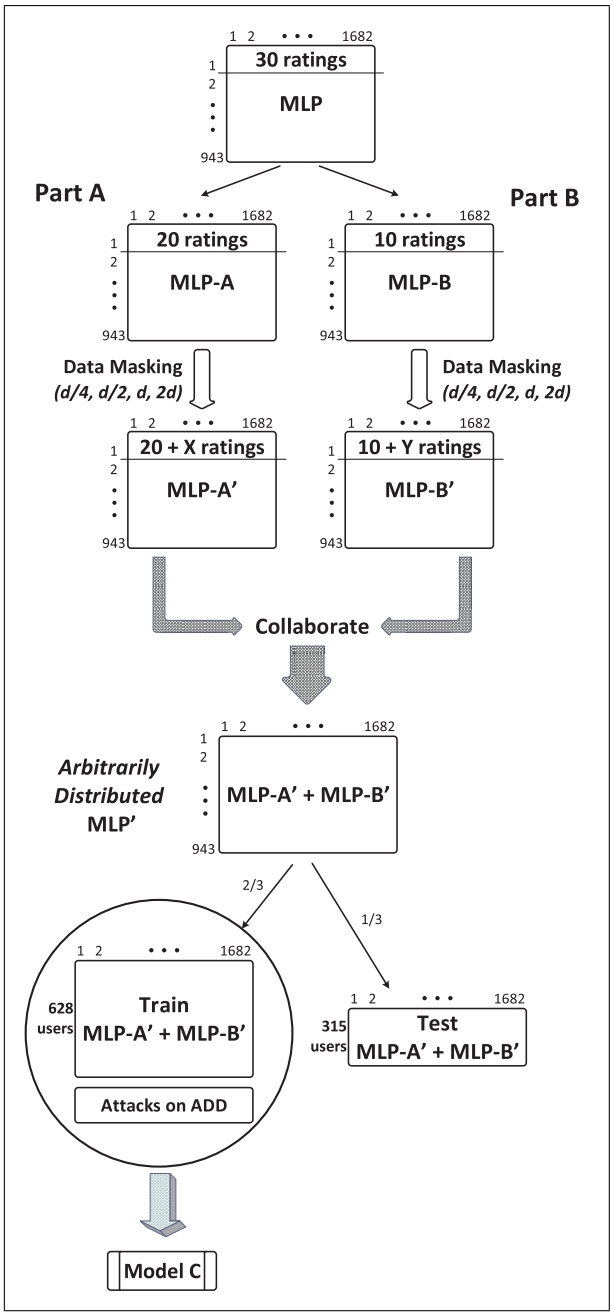

FIGURE 4. Methodology for experiment 2.

Before collaboration, each part masks their individual data by applying Random Filling. Then, the general experimentation methodology is followed to obtain train and test sets. Distributed user profiles in train sets constitute the training data for the proposed attack detection algorithm. For each trial, test profiles are injected and then run through the classifier, namely *Model C* in Fig. 4.

C. EMPIRICAL RESULTS

Two groups of experiments are conducted to find answers to the research questions that are stated in subsection V-B. Therefore, the first group of experiments is aimed to show how successful can the parties alone be in identifying distributed attack profiles, while the second group of experiments is demonstrated how successful can collaborating parties be in identifying distributed attack profiles.

1) EXP1: HOW SUCCESSFUL CAN THE PARTIES ALONE BE IN IDENTIFYING DISTRIBUTED ATTACK PROFILES?

By employing the classification-based attack detection method for centralized data [35], each part builds her own classifier model based on her own dataset as explained in subsection V-B1. In order to demonstrate performance of three classifiers, attack size parameter is set to 1% and effects of Recall versus Filler Size are examined.

Firstly, detection success of Average Nuke and Average Push attack models are analyzed, and the results are shown in Fig. 5 and Fig. 6. According to the results, while it is possible to detect whole attacks with filler size values equal to or greater than 10% by employing *Baseline* method, Recall values of *Baseline* algorithm is 72 and 86 for filler size 3% and 5%, respectively. On the other hand, when we analyze detection success of PartA and PartB, it is concluded that parties can reach the success of *Baseline* at filler size 10%, if the attacks are performed with filler size values greater than or equal to 30%. Also, Recall values for filler sizes 3% and 5% are less than 30, which means that the parties cannot detect those attacks. Note that, it is possible to manipulate a recommender system with a small number of filler sizes [35]. Therefore, we conclude the parties alone cannot be successful in identifying distributed attack profiles, especially, for smaller filler size values.

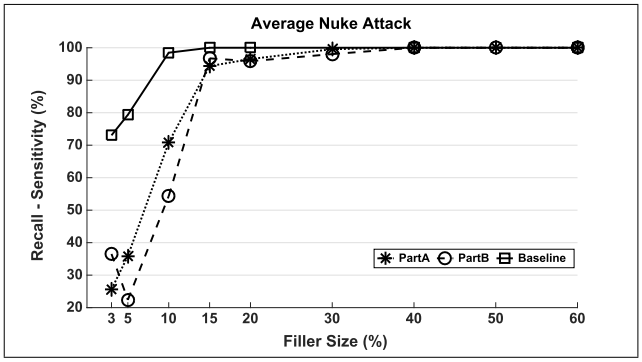

FIGURE 5. Recall vs. filler size for average nuke attacks.

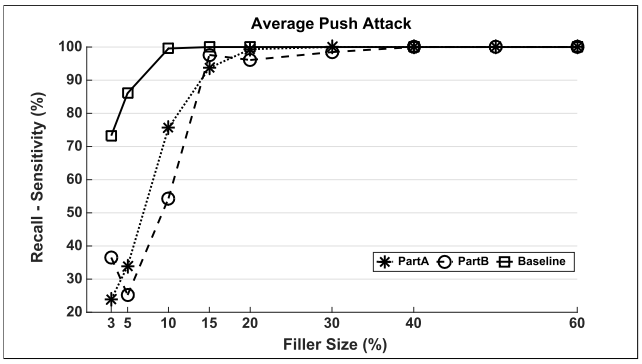

FIGURE 6. Recall vs. filler size for average push attacks.

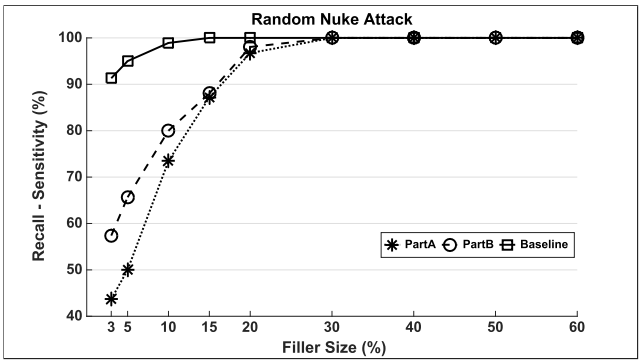

FIGURE 7. Recall vs. filler size for random nuke attacks.

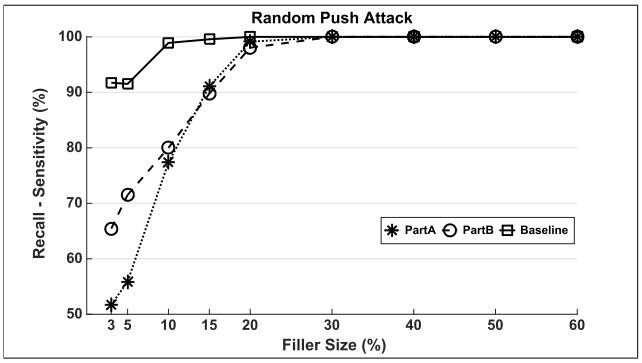

FIGURE 8. Recall vs. filler size for random push attacks.

Fig. 7 and Fig. 8 show the classification performances against Random Nuke and Random Push attack models. Even for small filler sizes *Baseline* classifier detects random nuke and push attacks with 90 Recall value. As seen from the results, 10% is a critical filler size value for random nuke and push attack models for the *Baseline*. *Baseline* classifier can detect whole random nuke and push attacks with filler size values equal to or higher than 10%. However, it is not possible to reach the same outcomes for PartA and PartB. As can be seen from the figures, it is difficult for PartA and PartB to detect distributed random nuke and push attacks, if they are generated with filler size less than 15%. For example, if an attacker employs 5% filler size, PartA can detect the

shilling profiles with Recall value less than 50. Although PartB appears to be more successful than PartA, since Recall value is less than 60, we can conclude that parties cannot successfully detect random attacks on their own.

The classification performance of PartA, PartB, and *Baseline* for Reverse Bandwagon Nuke and Bandwagon Push attack models are presented in Fig. 9 and Fig. 10. Compared to Average and Random attack models, PartA and PartB achieve better Recall results for small filler size values, when Reverse Bandwagon Nuke and Bandwagon Push attack models are employed. According to results, it is obtained that the *Baseline* method can reach 91 Recall value for filler size 3%. However, it can be stated that parties can obtain similar results for filler size 10% and in order to detect whole attacks, PartA and PartB need the filler size is equal to or greater than 15%. Again, the parties fail to decrease the effect of bandwagon attacks individually.

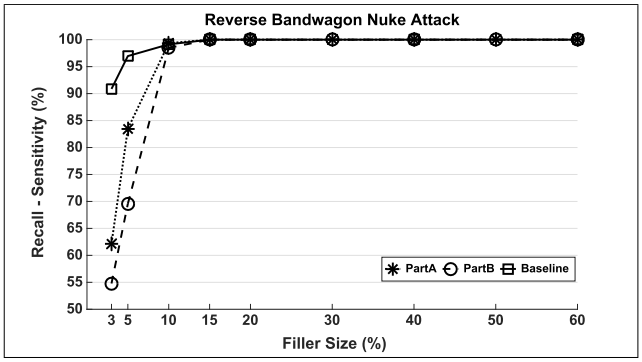

FIGURE 9. Recall vs. filler size for reverse bandwagon Attacks.

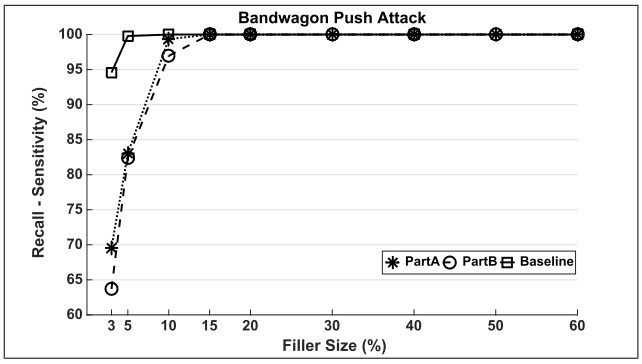

FIGURE 10. Recall vs. filler size for bandwagon attacks.

Attack detection performance of PartA, PartB, and *Baseline* for Love/Hate nuke attack model is given in Fig. 11. Recall value of the *Baseline* classifier starts with 55 for 3% filler size, increases to 88 for 5% filler size, and reach 100 for filler sizes equal to or higher than 10%. When the Recall results obtained for PartA and PartB are examined, it can be stated that, while PartB detects whole distributed Love/Hate attack profiles more successfully than the *Baseline*, especially for small filler sizes, the other part, which is PartA in the figure, cannot detect them as successfully as PartB and *Baseline* up to filler size of 10%. Hence, for distributed Love/Hate

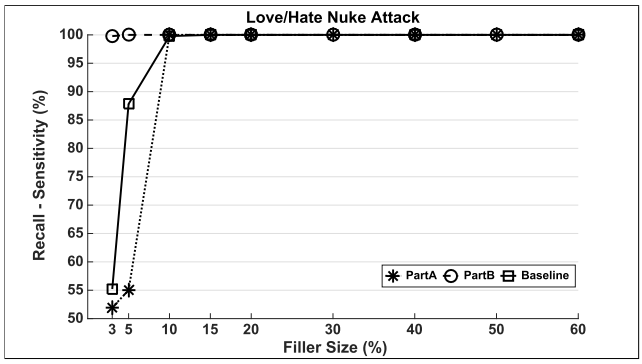

FIGURE 11. Recall vs. filler size for love/hate attacks.

attacks on ADD, while one part might find the attack profiles, the other part might not find them for same filler size values. Such phenomenon occurs due to the random distribution of the generated Love/Hate attack profiles between the parts.

Fig. 12 shows the classification performance of PartA, PartB, and *Baseline* for Segment Attack model, which aims to push an item to a targeted group of users with a specific interest. The results obtained for the Segment Attack model are significantly different from the Recall results obtained for other attack models. When Recall results obtained by the *Baseline* classifier for various filler sizes are analyzed, it can be stated that while 82 Recall is achieved for 3% filler size, for the values equal to or higher than 5% 100 Recall value is obtained. PartA and PartB to achieve such a Recall value, filler size value has to be equal to or higher than 40%, which is too high for a reasonable attack strategy. For smaller filler size values, while one part can detect the distributed attack profiles, the other part almost cannot detect whole attacks. There are three critical filler size values for PartA and PartB, which are 10%, 20%, and 30%. Up to 20% filler size, while Recall of PartA shows a decreasing behavior, PartB's Recall results are increasing. Between 20% and 30% filler sizes Recall of both PartA and PartB are increasing, and these results reach *Baseline* classifier when filler size is 40%.

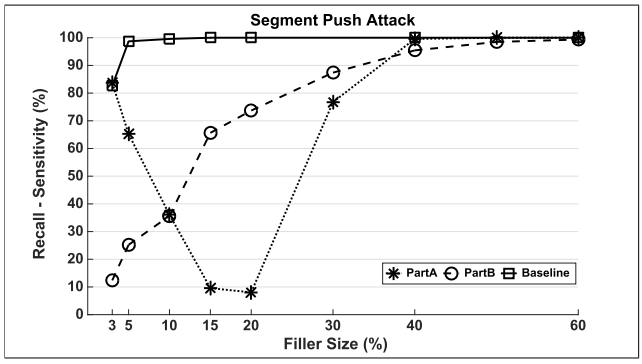

FIGURE 12. Recall vs. filler size for segment attacks.

When Recall results are compared among each other, results obtained for PartA and PartB are seen as successful as

the *Baseline* for Bandwagon and Love/Hate Attack models. However, Precision is particularly a problem for PartA and PartB classifiers, especially for these attack models. Fig. 13 compares Precision results for PartA, PartB, and *Baseline* for Bandwagon Push Attack on various filler size values. As the figure indicates, Precision values obtained for PartA and PartB are far below the *Baseline* case even for large filler sizes. Hence, many false-positive identifications are made by PartA and PartB compared to *Baseline*, which means that classifiers on PartA and PartB label significant amounts of authentic profiles as an attack. This outcome is also valid for Love/Hate Attack model. In Fig. 14 Precision results obtained for PartA, PartB and *Baseline* for Love/Hate Attack model on different filler size values are shown. Precision results of PartB are far below the *Baseline*, and even though the Precision values obtained by PartA are better than PartB, these results still cannot reach the *Baseline*. Therefore, PartA and PartB classifiers misclassify some authentic profiles and label them as an attack.

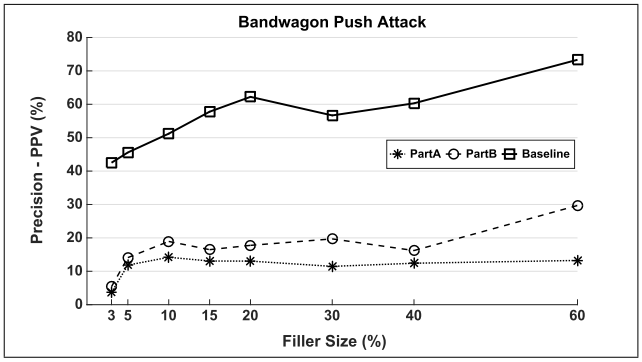

FIGURE 13. Precision - PPV vs. filler size for bandwagon attacks.

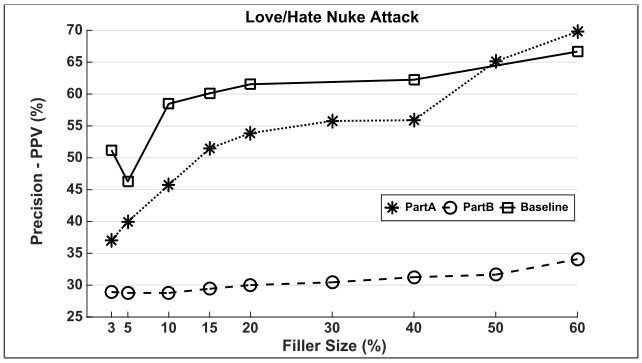

FIGURE 14. Precision - PPV vs. filler size for love/hate attacks.

Classification results obtained for several push and nuke attack models indicate that classifiers built by parts alone are not as successful as the *Baseline*. Even if collaborating parts have their detection mechanisms, since these classifiers are trained based on partial data, instead of on all data, they cannot detect distributed shilling attacks on small filler sizes. Therefore, collaboration of parts is necessary for detecting distributed shilling attacks on ADD without jeopardizing privacy.

2) EXP2: HOW SUCCESSFUL CAN COLLABORATING PARTIES BE IN IDENTIFYING DISTRIBUTED ATTACK PROFILES?

The first part of the experimental results states that the parties cannot successfully detect malicious users with small filler size values. Although it is possible to detect the attacks with larger filler size values, such an attack profile is not reasonable in real-life scenarios. Therefore, the parties must collaborate in order to produce reliable recommendations while operating on ADD. In this subsection, the classification success of collaboration is analyzed. For shilling attack detection, the proposed classification-based shilling attack detection method for ADD is employed between PartA and PartB. A variety of experiments are conducted to investigate whether both confidentiality and classification accuracy can be achieved simultaneously by utilizing the proposed attack detection method for ADD.

For detecting distributed shilling attacks on ADD without revealing privacy, parts need to mask their data by inserting unreal ratings into the uniformly randomly selected empty cells. In order to generate unreal ratings, there are two possible methods [7], which are personalized or non-personalized rating generation. Yakut and Polat [7] experimentally show that user-mean gives slightly better results compared to other methods. Therefore, in the experiments conducted in this section, user-mean votes are assigned to empty cells for random filling. To capture the balance between privacy and classification accuracy, the number of filled cells is an important parameter. Note that, privacy and classification accuracy are conflicting goals. As a result, it is expected that when the number of unreal ratings increases, the privacy of each party also enhances, while the accuracy of classification deteriorates. On the other hand, assigned user-mean votes might have a positive effects on accuracy, since datasets are too sparse [60].

In order to explore the effects of the number of unrated cells to be filled on classification accuracy, trials are performed by various perturbation levels, i.e., $d/4, d/2, d, 2d$, where d represents the density of any user. Note that, the values for $d = 0$, which indicates no perturbation, demonstrates baseline results in the figures. During the experiments filler size is set to 5% and attack size to 1%. The results are shown in Fig. 15 for nuke (AVNuke: Average Nuke - LH: Love/Hate - RNuke: Random Nuke - RBW: Reverse Bandwagon) and Fig. 16 for push (AVPush: Average Push - BW: Bandwagon - RPush: Random Push - SG: Segment) attack models. According to Fig. 15, it can be seen that for nuke attack models, adding a small amount of noise by inserting unreal ratings into $d/4$ number of empty cells, improves Recall results of the classifier compared to the Baseline. When the level of perturbation increases to $d/2$, the results are still better than the base case except for the Love/Hate attack model, but slightly worse than the results obtained with $d/4$. The adverse effect of noise on the classification results is beginning to appear slightly after $d/2$, and precisely after d. It can be concluded from these results that, adding small amounts of unreal ratings, introduce

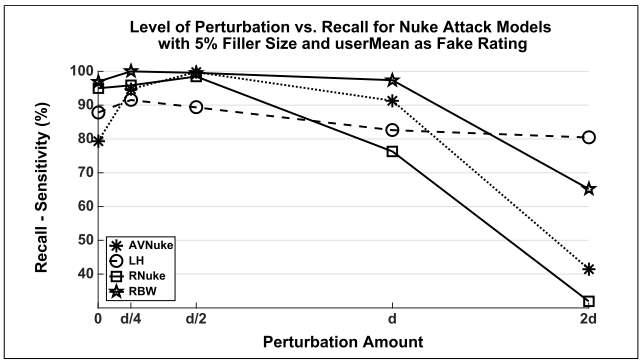

FIGURE 15. Perturbation amount vs. recall for nuke attack models with 5% filler size.

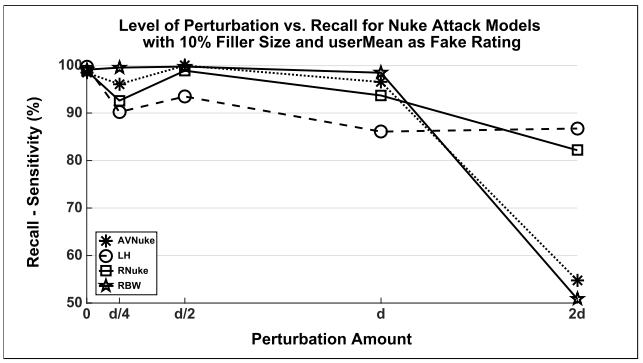

FIGURE 17. Perturbation amount vs. recall for nuke attack models with 10% filler size.

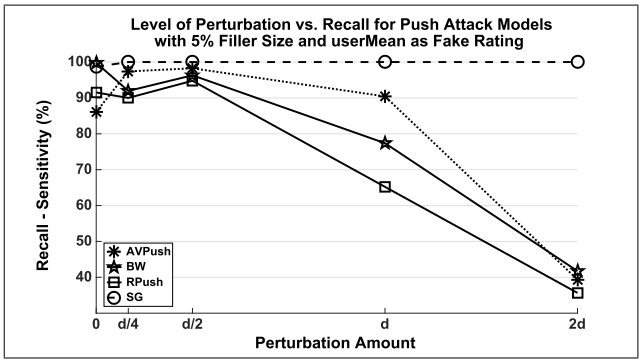

FIGURE 16. Perturbation amount vs. recall for push attack models with 5% filler size.

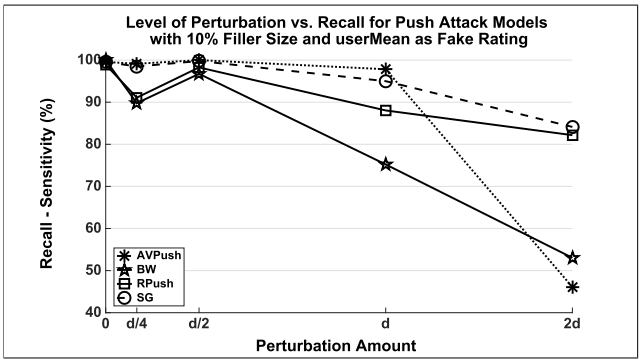

FIGURE 18. Perturbation amount vs. recall for push attack models with 10% filler size.

an improvement at classification accuracy. Since MLP dataset is too sparse, such positive effect of random filling is not surprising as stated in [60]. However, when the perturbation amount grows from $d/2$ to $2d$, the classification performance decreases dramatically compared to the base case, although the perturbation amount of d still achieves acceptable Recall values for nuke attack models. However, selecting a perturbation amount larger than d completely worsens results.

Similar outcomes can be inference for push attack models as shown in Fig. 16. When the perturbation amount is set to $d/4$, Recall results of the classifier for Segment and Average Push attack models become slightly better, whereas for Bandwagon and Random Push attack models results become slightly worse compared to Baseline. For all of the push attack models, the best Recall results are achieved when the level of perturbation increases to $d/2$. Like the nuke attack models, the adverse effect of adding noise on obtained Recall values for push attack models appears to be seen after $d/2$ and becomes even worse at $2d$. Although Recall results of push attack models are not as good as the ones of nuke attacks for perturbation amount of d, depending on the need of the privacy level, it can be chosen. Results established from nuke and push attack models with 5% filler size present that it is possible to achieve both confidentiality and classification accuracy by utilizing $d/4$ or $d/2$ as the level of perturbation.

In order to understand the relation between random filling and filler size, we rerun the previous trials with filler size of 10% and the results are given in Fig. 17 and Fig. 18, respectively for nuke and push attacks. It can be seen from Fig. 17 that adding a small amount of noise slightly deteriorates Recall results of the classifier compared to the Baseline, as in the case of $d/4$, yet these results are still accurate for nuke attack models with larger filler size. The best Recall values are obtained when the level of perturbation is $d/2$ for nuke attack models. Even though increasing the level of perturbation from $d/2$ to d harms Recall of the classifier for nuke attack models, they are still acceptable compared to Baseline, especially if a higher level of privacy is required. On the other hand, after d, increase in perturbation amount worsens the Recall results, especially for Reverse Bandwagon and Average Nuke attack models. Recall versus perturbation amount results for push attack models indicates similar outcomes with the nuke attack models with 10% filler size.

As seen in Fig. 18, initial Recall value for all nuke attack models is 100%. Adding some level of noise slightly harms the initial results, such as in $d/4$. While Segment and Average Push attack models are not affected by this noise, Bandwagon and Random Push attacks are slightly affected. As the case for nuke attack models, when the level of perturbation is set to $d/2$, the best Recall results for push attacks are obtained, which are very close to Baseline. The adverse effect of adding noise on obtained Recall values for push attack models appears to be seen after $d/2$, and becomes worse at $2d$. Even though Recall results of push attack models are not as

good as the ones of nuke attacks for perturbation amount of d, depending on the need of the privacy level, d can also be chosen. As Recall results obtained at $2d$ indicate, increase in perturbation amount worsens the classification specifically for Bandwagon and Average Push attack models. Results established from nuke and push attack models with 10% filler size present that it is possible to achieve both confidentiality and classification accuracy by utilizing $d/4$ or $d/2$ as the level of perturbation, which is identical to the results obtained for attack models with 5% filler size.

These empirical results demonstrate that a balance between privacy and classification accuracy can be assured for well-known shilling attack models. Consequently, utilizing the proposed classification-based attack detection method, shilling attacks on ADD can be detected with decent accuracy without jeopardizing data owners' privacy.

VI. CONCLUSIONS AND FUTURE WORK

In this study, to protect privacy-preserving distributed collaborative filtering algorithms against shilling attacks, and to keep up the collaboration of online vendors on arbitrary data, distributed version of a classification-based attack detection method is presented. Private protocols are proposed to derive the required generic, and model-specific classification attributes for each distributed profile collaboratively between two parts. For secure computations, homomorphic encryption and random filling techniques are utilized in the protocols for protecting the confidentiality of data holders. By operating the proposed private attribute estimation protocols consecutively, classification attributes are derived off-line among the parts. In the end, half of the derived attributes are known by each part. Then, by exchanging the attributes, the classification model is constructed with the k-NN algorithm. At this point, even though both parts have the model, a collaboration between parties is required for testing and classifying a new instance. In order to generate the required classification attributes for the new instance, parts need to apply the same process collaboratively. Empirical analyzes show that with the proposed method it is still possible to detect attacks on arbitrarily distributed data efficiently, without jeopardizing data owners's privacy.

The study has many future directions.

1) To keep collaboration of the online vendors on arbitrarily distributed binary data, new detection methods are required. Indeed, there is no binary shilling attack detection algorithm in the literature. Hence, shilling attack detection methods on binary data both for central server-based systems, and distributed systems need to be investigated.
2) There are numerous shilling attack detection algorithms proposed in the literature, hence distributed versions of these methods, which can operate on arbitrarily distributed data while preserving privacy, can be presented. For some clustering based detection algorithms, private protocols provided in this study can be utilized in deriving the required attributes.
3) In this study, it is assumed that there are no overlapping ratings. However, in real life scenarios, overlapping ratings are inevitable. Therefore the effects of overlapping ratings should be studied.
4) Developing robust privacy-preserving distributed collaborative filtering algorithms for arbitrarily distributed data, which are intrinsically resistant to attacks, might also be an exciting research topic.
5) This work can be employed with improvements if researchers propose the collaboration of multiple parties on arbitrarily distributed data. If multiple data holders are considered, it must be noted that parties may coalesce for capturing a target party's data, therefore proposed protocols must be revised to prevent such data holders' attacks.

REFERENCES

[1] S. Berkovsky and J. Freyne, "Web personalization and recommender systems," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. Sydney, NSW, Australia, Aug. 2015, pp. 2307–2308.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[3] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Found. Trends Hum.-Comput. Interact.*, vol. 4, no. 2, pp. 81–173, Feb. 2010.

[4] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. New York, NY, USA: Cambridge Univ. Press, 2010.

[5] M. P. O'Mahony, "Towards robust and efficient automated collaborative filtering," Ph.D. dissertation, Dept. Comput. Sci., Univ. College Dublin, Dublin, Ireland, Dec. 2004.

[6] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, 2001.

[7] I. Yakut and H. Polat, "Arbitrarily distributed data-based recommendations with privacy," *Data Knowl. Eng.*, vol. 72, pp. 239–256, Feb. 2012.

[8] C. Kaleli and H. Polat, "SOM-based recommendations with privacy on multi-party vertically distributed data," *Oper. Res. Soc.*, vol. 63, no. 4, pp. 826–838, 2012.

[9] H. Polat and W. Du, "Privacy-preserving top-N recommendation on distributed data," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 59, no. 7, pp. 1093–1108, 2008.

[10] A. J. P. Jeckmans, "Cryptographically-enhanced privacy for recommender systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Twente, Enschede, The Netherlands, 2014.

[11] *OECD Guidelines for Consumer Protection in the Context of Electronic Commerce*, OECD Publishing, Paris, France, 2000.

[12] *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*, OECD Publishing, Paris, France, 2005.

[13] C. Kaleli and H. Polat, "Privacy-preserving SOM-based recommendations on horizontally distributed data," *Knowl.-Based Syst.*, vol. 33, pp. 124–135, Sep. 2012.

[14] C. Kaleli and H. Polat, "Privacy-preserving Naïve Bayesian classifier–based recommendations on distributed data," *Comput. Intell.*, vol. 31, pp. 47–68, Feb. 2015.

[15] A. Bilge, C. Kaleli, I. Yakut, and I. Güneş, and H. Polat, "A survey of privacy-preserving collaborative filtering schemes," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 23, no. 8, pp. 1085–1108, 2013.

[16] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. Chicago, IL, USA, Aug. 2005, pp. 593–599.

[17] I. Güneş, 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.

[18] I. Güneş, A. Bilge, C. Kaleli, and H. Polat, "Shilling attacks against privacy-preserving collaborative filtering," *J. Adv. Manage. Sci.*, vol. 1, no. 1, pp. 54–60, 2013.

[19] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Promoting recommendations: An attack on collaborative filtering," in *Proc. 13th Int. Conf. Database Expert Syst. Appl. (DEXA)*. Aix-en-Provence, France. Berlin, Germany: Springer, Sep. 2002, pp. 494–503.

[20] B. Mehta and T. Hofmann, "A survey of attack-resistant collaborative filtering algorithms," *IEEE Data Eng. Bull.*, vol. 31, no. 2, pp. 14–22, Jun. 2008.

[21] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Trans. Internet Technol.*, vol. 7, no. 4, pp. 23–60, 2007.

[22] B. Y. Yılmazel and C. Kaleli, "Robustness analysis of arbitrarily distributed data-based recommendation methods," *Expert Syst. Appl.*, vol. 44, pp. 217–229, Feb. 2016.

[23] A. Jeckmans, Q. Tang, and P. Hartel, "Privacy-preserving collaborative filtering based on horizontally partitioned dataset," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*. Denver, CO, USA. Washington, DC, USA: IEEE Computer Society, May 2012, pp. 439–446.

[24] H. Polat and W. Du, "Privacy-preserving top-*N* recommendation on horizontally partitioned data," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*. Washington, DC, USA: IEEE Computer Society, Sep. 2005, pp. 725–731.

[25] H. Polat and W. Du, "Privacy-preserving collaborative filtering on vertically partitioned data," in *Knowledge Discovery in Databases: PKDD 2005* (Lecture Notes in Computer Science), vol. 3721, A. M. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, Eds. Berlin, Germany: Springer, 2005.

[26] C. Kaleli and H. Polat, "Providing Naïve Bayesian classifier-based private recommendations on partitioned data," in *Knowledge Discovery in Databases: PKDD 2007* (Lecture Notes in Computer Science), vol. 4702, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Germany: Springer, 2007.

[27] I. Yakut and H. Polat, "Privacy-preserving SVD-based collaborative filtering on partitioned data," *Int. J. Inf. Technol. Decis. Making*, vol. 09, no. 3, pp. 473–502, 2010.

[28] I. Yakut and H. Polat, "Privacy-preserving hybrid collaborative filtering on cross distributed data," *Knowl. Inf. Syst.*, vol. 30, no. 2, pp. 405–433, 2012.

[29] I. Yakut and H. Polat, "Estimating NBC-based recommendations on arbitrarily partitioned data with privacy," *Knowl.-Based Syst.*, vol. 36, pp. 353–362, Dec. 2012.

[30] R. Burke, M. P. O'Mahony, and N. J. Hurley, "Robust collaborative recommendation," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA, USA: Springer, 2015.

[31] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. Philadelphia, PA, USA: ACM, New York, NY, USA, Aug. 2006, pp. 542–547.

[32] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Detecting profile injection attacks in collaborative recommender systems," in *Proc. 8th IEEE Int. Conf. E-Commerce Technol. 3rd IEEE Int. Conf. Enterprise Comput., E-Commerce, E-Services (CEC/EEE)*. San Francisco, CA, USA: IEEE Computer Society, Washington, DC, USA, Jun. 2006, pp. 23–30.

[33] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proc. 7th Annu. ACM Int. Workshop Web Inf. Data Manage. (WIDM)*. Bremen, Germany: ACM, New York, NY, USA, Nov. 2005, pp. 67–74.

[34] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik, "Analysis and detection of Segment-focused attacks against collaborative recommendation," in *Proc. 7th Int. Workshop Knowl. Discovery Web (WebKDD), Adv. Web Mining Web Usage Anal.*. Chicago, IL, USA. Berlin, Germany: Springer, Aug. 2006, pp. 96–118.

[35] C. Williams and B. Mobasher, "Profile injection attack detection for securing collaborative recommender systems," M.S. thesis, Dept. Comput. Sci., DePaul Univ., Chicago, IL, USA, Jun. 2006.

[36] C. Williams, R. Bhaumik, R. Burke, and B. Mobasher, "The impact of attack profile classification on the robustness of collaborative recommendation," in *Proc. KDD Workshop Web Mining Web Usage Anal. (WebKDD), 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. Philadelphia, PA, USA: ACM, New York, NY, USA, Aug. 2006, pp. 1–10.

[37] F. He, X. Wang, and B. Liu, "Attack detection by rough set theory in recommendation system," in *Proc. IEEE Int. Conf. Granular Comput. (GrC)*. San Jose, CA, USA: IEEE Computer Society, Washington, DC, USA, Aug. 2010, pp. 692–695.

[38] F. Zhang and Q. Zhou, "A meta-learning-based approach for detecting profile injection attacks in collaborative recommender systems," *J. Comput.*, vol. 7, no. 1, pp. 226–234, 2012.

[39] F. Zhang and Q. Zhou, "HHT–SVM: An online method for detecting profile injection attacks in collaborative recommender systems," *Knowl.-Based Syst.*, vol. 65, pp. 96–105, Jul. 2014.

[40] M. A. Morid, M. Shajari, and A. R. Hashemi, "Defending recommender systems by influence analysis," *Inf. Retr.*, vol. 17, no. 2, pp. 137–152, Apr. 2014.

[41] F. Zhang and Q. Zhou, "Ensemble detection model for profile injection attacks in collaborative recommender systems based on BP neural network," *IET Inf. Secur.*, vol. 9, no. 1, pp. 24–31, 2015.

[42] F. Zhang and H. Chen, "An ensemble method for detecting shilling attacks based on ordered item sequences," *Secur. Commun. Netw.*, vol. 9, no. 7, pp. 680–696, 2016.

[43] R. Bhaumik, C. Williams, B. Mobasher, and R. Burke, "Securing collaborative filtering against malicious attacks through anomaly detection," in *Proc. 4th Workshop Intell. Techn. Web Personalization (ITWP)*. Boston, MA, USA: AAAI Press, Jul. 2006, pp. 10–20.

[44] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proc. of the 13th Int. Conf. World Wide Web (WWW)*. New York, NY, USA, May 2004, pp. 393–402

[45] B. Mobasher, R. Burke, R. Bhaumik, and J. J. Sandvig, "Attacks and remedies in collaborative recommendation," *IEEE Intell. Syst.*, vol. 22, no. 3, pp. 56–63, May 2007.

[46] A. Bilge, I. Güneş, and H. Polat, "Robustness analysis of privacy-preserving model-based recommendation schemes," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3671–3681, 2014.

[47] C. C. Aggarwal, "Attack-resistant recommender systems," in *Recommender Systems*. Cham, Switzerland: Springer, 2016.

[48] C. A. Williams, B. Mobasher, R. Burke, and R. Bhaumik, "Detecting profile injection attacks in collaborative filtering: A classification-based approach," in *Adv. Web Mining Web Usage Anal., 8th Int. Workshop Knowl. Discovery Web (WebKDD)*, in Lecture Notes in Computer Science, vol. 4811, Philadelphia, PA, USA, O. Nasraoui, M. Spiliopoulou, J. Srivastava, B. Mobasher, and B. M. Masand, Eds. Berlin, Germany: Springer, Aug. 2007, pp. 167–186.

[49] C. A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: Detection of profile injection attacks," *Service Oriented Comput. Appl.*, vol. 1, no. 3, pp. 157–170, Nov. 2007.

[50] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Trans. Internet Technol.*, vol. 4, no. 4, pp. 344–377, 2004.

[51] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Effective attack models for shilling item-based collaborative filtering systems," in *Proc. WebKDD Workshop, Held Conjuction ACM SIGKDD*. Chicago, IL, USA: ACM, New York, NY, USA, Aug. 2005, pp. 13–23.

[52] H. Polat and W. Du, "Privacy-preserving collaborative filtering," *Int. J. Electron. Commerce*, vol. 9, no. 4, pp. 9–35, 2005.

[53] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*. Chicago, IL, USA: IEEE Computer Society, Washington, DC, USA, Nov. 1982, pp. 160–164.

[54] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 79-1–79-35, 2018.

[55] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. 17th Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Prague, Czech Republic. Berlin, Germany: Springer-Verlag, May 1999, pp. 223–238.

[56] M. Dahl, C. Ning, and T. Toft, "On secure two-party integer division," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 7397, A. D. Keromytis, Ed. Ed. Berlin, Germany: Springer, 2012, pp. 164–178.

[57] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Floating-point homomorphic encryption," in *Proc. IACR Cryptol. ePrint Arch.*, 2016, p. 421.

[58] B. Memiş and I. Yakut, "Privacy-preserving two-party collaborative filtering on overlapped ratings," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 8, pp. 2948–2966, 2014.

[59] D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. Thorpe, "Practical secrecy-preserving, verifiably correct and trustworthy auctions," *Electron. Commerce Res. Appl.*, vol. 7, no. 3, pp. 294–312, 2008.
[60] A. Bilge and H. Polat, "A comparison of clustering-based privacy-preserving collaborative filtering schemes," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2478–2489, 2013.

BURCU YILMAZEL was born in Izmir, Turkey, in 1981. She received the B.S., M.S., and Ph.D. degrees in computer engineering from Anadolu University, Eskisehir, Turkey, in 2005, 2008, and 2016, respectively. Her research interests include recommender systems and privacy-preserving information filtering.

ALPER BILGE was born in Bursa, Turkey, in 1983. He received the B.S. degree in electrical and electronics engineering and the M.S. and Ph.D. degrees in computer engineering from Anadolu University, Eskisehir, Turkey, in 2005, 2008, and 2013, respectively. His research interests include recommender systems, information filtering, and privacy-preserving information processing.

CIHAN KALELI (M'19) was born in Istanbul, Turkey, in 1982. He received the master's and Ph.D. degrees from the Computer Engineering Department, Anadolu University, in 2008 and 2012, respectively. He is currently with the Computer Engineering Department, Eskisehir Technical University, Turkey. His research interest is privacy-preserving data mining in general and specifically studies on distributed data-based collaborative filtering with privacy.