# DEVELOPING TECHNIQUES FOR ROBUSTNESS OF PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING

**A dissertation submitted for the degree of**
***Doctor of Philosophy***

**Burcu YILMAZEL**
**Eskişehir, 2016**

# DEVELOPING TECHNIQUES FOR ROBUSTNESS OF PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING

**Burcu YILMAZEL**

**A dissertation submitted for the degree of**
***Doctor of Philosophy***

**Department of Computer Engineering**
**Supervisor: Assoc. Prof. Dr. Cihan KALELİ**

**Eskişehir**
**Anadolu University**
**Graduate School of Science**
**October, 2016**

## FINAL APPROVAL FOR THESIS

This thesis titled "**Developing Techniques for Robustness of Privacy Preserving Distributed Collaborative Filtering**" has been prepared and submitted by **Burcu Yılmazel** in partial fullfillment of the requirements in "Anadolu University Directive on Graduate Education and Examination" for the Degree of PhD in Computer Engineering Department has been examined and approved on 07/10/2016.

| **Committee Members** | | **Signature** |
|---|---|---|
| Member (Supervisor) | : Assoc. Prof. Dr. Cihan KALELİ | .............................. |
| Member | : Assoc. Prof. Dr. Gürkan ÖZTÜRK | .............................. |
| Member | : Assist. Prof. Dr. Alper BİLGE | .............................. |
| Member | : Assist. Prof. Dr. Efnan ŞORA GÜNAL | .............................. |
| Member | : Assist. Prof. Dr. Mehmet KOÇ | .............................. |

Director
Graduate School of Science

## ABSTRACT

DEVELOPING TECHNIQUES FOR ROBUSTNESS OF PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING

Burcu YILMAZEL

Department of Computer Engineering
Anadolu University, Graduate School of Science, October, 2016

Supervisor: Assoc. Prof. Dr. Cihan KALELİ

Success of collaborative filtering systems strongly depend on having adequate data. Due to customers' shopping habits and increasing number of e-commerce sites, data collected for referral purposes might be distributed among various sites. Therefore, especially for newly established companies, offering recommendation services might turn out to be a trouble, due to lack of qualified data. To overcome this challenge, collaboration of online vendors on distributed data while preserving privacy has become an important topic. Researchers have proposed several privacy-preserving distributed collaborative filtering schemes, which enable collaboration of online vendors, even the competing ones, on distributed data without jeopardizing privacy. However, such schemes have not been evaluated in terms of robustness against attacks. If manipulating the outcomes of privacy-preserving distributed collaborative filtering algorithms by injecting fake profiles is possible, shilling attacks might be an obstacle for collaboration. Online vendors, who are unsure of being subject to shilling attacks, might refrain from cooperation, even if they need it for offering more useful recommendation services to their customers.

In this dissertation, robustness of state-of-the-art privacy-preserving distributed collaborative filtering schemes proposed for arbitrarily distributed data are analyzed against shilling attacks. A new attack strategy that can be applied on arbitrarily distributed data, and used in generation of distributed adaptations of formerly proposed attack models is outlined. Empirical studies show that attacks generated by the proposed strategy are effective in manipulating predicted outcomes, hence, despite privacy, these schemes are not resistant to attacks. The reasons of why existing shilling attack detection methods cannot be directly employed on arbitrarily distributed data are discussed. To protect these algorithms against attacks, distributed version of a well-known classification-based attack detection method is proposed, which can operate on arbitrarily distributed data. Real data-based experiments demonstrate that the proposed detection method is able to identify distributed attack profiles on arbitrary data with privacy. Moreover, the need for collaboration in detection of distributed attacks is exposed with experimental analyzes.

**Keywords:** Robustness, Shilling Attacks, Profile Injection Attacks, Detection, Distributed Data, Arbitrarily, Recommendation, Collaborative Filtering, Privacy, E-commerce.

## ÖZET

GİZLİLİĞİ KORUYAN DAĞITIK VERİ TABANLI ORTAK FİLTRELEME METOTLARININ GÜRBÜZLÜĞÜ İÇİN TEKNİKLER GELİŞTİRİLMESİ

Burcu YILMAZEL

Bilgisayar Mühendisliği Anabilim Dalı
Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Ekim, 2016

Danışman: Doç. Dr. Cihan KALELİ

Ortak filtreleme sistemlerinde başarıya ulaşabilmek için yeterli ve uygun veriye sahip olmak gerekir. Internet alışverişlerindeki kullanıcı tercihleri ve e-ticaret firmalarındaki çeşitlilikten dolayı, analiz için kullanılabilecek veri birçok farklı kaynağa dağılmış durumdadır. Nitelikli verinin azlığı, özellikle yeni kurulan firmalar için, öneri hizmetlerinin sağlanmasında önemli bir sorun teşkil eder. Bu sorunun çözümü için dağıtık veri üzerinde işbirliği yapılması, bu işbirliği esnasında da gizliliğin korunması önemli bir araştırma konusu haline gelmiştir. Dağıtık ortak filtreleme üzerine yapılmış birçok çalışma bulunmaktadır. Bu çalışmalar ile online sağlayıcıların gizlilik ilkelerini ihlal etmeden işbirliğinde bulunmaları sağlanmıştır. Fakat bu çalışmalar, ataklara karşı gürbüzlük açısından değerlendirilmemiştir. Sahte profil enjeksiyonu ile gizlilik koruyan, dağıtık ortak filtreleme algoritmalarının sonuçlarına müdahale edilebilirse, shilling ataklar işbirliğine engel oluşturabilir. Shilling ataklarına karşı sistemine güven duymayan bir online sağlayıcı, daha iyi öneri hizmeti sunabilme fırsatına karşın işbirliğinden kaçınabilir.

Bu tezde, gizlilik koruyan dağıtık ortak filtreleme yöntemlerinin, gelişigüzel dağıtılmış veride shilling ataklara karşı gürbüzlüğü incelenmiştir. Gelişigüzel dağıtılmış veri üzerine uygulanabilecek yeni bir atak stratejisi belirlenmiş ve bu strateji daha önceki atak modellerinin dağıtık uyarlamalarının oluşturulmasında kullanılmıştır. Deneysel çalışmalar, öne sürülen strateji ile oluşturulan atakların tahmin edilen sonuçları değiştirmede etkili olduğunu, dolayısıyla da bu sistemlerin gizliliğe rağmen saldırılara açık olduğunu göstermiştir. Mevcut shilling atak tespit yöntemlerinin, gelişigüzel dağıtık veride uygulanamamasının nedenleri açıklanmıştır. Bu algoritmaları ataklardan korumak için, çok iyi bilinen sınıflandırma tabanlı bir tespit yönteminin dağıtık versiyonu öne sürülmüştür. Gerçek veri kullanılarak yapılan deneyler, öne sürülen yöntemin atak profillerini gizlilik kuralları çerçevesinde tespit edebildiğini göstermiştir. Ayrıca, dağıtık atakların tespiti için tarafların işbirliğinin gerekliliği deneysel analizlerle kanıtlanmıştır.

**Anahtar Sözcükler:** Gürbüzlük, Shilling Atakları, Profil Enjeksiyon Atakları, Bulma, Dağıtık Veri, Rastgele, Öneri, Ortak Filtreleme, Gizlilik, E-ticaret.

## ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my supervisor, Assoc. Prof. Dr. Cihan KALELİ, who has supported me throughout my dissertation with his patience and unsurpassed knowledge. I thank him for his guidance and encouragement, where good advice, support, and friendship has been invaluable on both academic and personal levels.

Also, I would like to thank Assoc. Prof. Dr. Gürkan ÖZTÜRK, Assist. Prof. Dr. Alper BİLGE, Assist. Prof. Dr. Mehmet KOÇ and Assist. Prof. Dr. Efnan ŞORA GÜNAL on my dissertation committee for their valuable contributions.

A special gratitude and love goes to my parents, my brother Ali YÜREKLİ and my sister Müge TOYDEMİR for all the support and encouragement that I received throughout my studies and before.

I am especially thankful to my lovely son, Barış, for his sweet smile and understanding.

Finally, I want to express my deepest love and thanks to my dear husband, İ. Barış YILMAZEL, for his love, support, understanding, patience, and inspiration at all times.

Burcu YILMAZEL

October, 2016

07/10/2016

## STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Burcu YILMAZEL

# CONTENTS

Page

# LIST OF TABLES

# LIST OF FIGURES

# GLOSSARY OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| $\delta$ | Delta |
| $\gamma$ | Gamma |
| $\sigma$ | Sigma |
| ADD | Arbitrarily Distributed Data |
| AS | Attack Size |
| CDD | Cross Distributed Data |
| CF | Collaborative Filtering |
| $\mathfrak{D}$ | Decryption Function |
| $d$ | Density |
| DegSim | Degree of Similarity with Top Neighbours |
| DegSim’ | Degree of Similarity with Co-Rated Factor |
| $\xi$ | Encryption Function |
| $\xi_K(x)$ | Encrypted Value of $x$ with Key $K$ |
| EDM | Ensemble Detection Model |
| FAC | Filler Average Correlation |
| FMD | Filler Mean Difference |
| FMTD | Filler Mean Target Difference |
| FMV | Filler Mean Variance |
| FN | False Negative |
| FP | False Positive |

| | |
|---|---|
| FS | Filler Size |
| GFMV | Group Filler Mean Variance |
| HDD | Horizontally Distributed Data |
| HE | Homomorphic Encryption |
| HySAD | Hybrid Shilling Attack Detector |
| $I_F$ | Set of Filler Items |
| $I_\emptyset$ | Set of Unrated Items |
| $I_S$ | Set of Selected Items |
| $i_t$ | Single Target Item |
| $K$ | Public Key |
| $k$-NN | $k$-Nearest Neighbors |
| LengthVar | Length Variance |
| $m$ | Total Number of Items in the System |
| MeanVar | Mean Variance |
| $n$ | Total Number of Users in the System |
| NBC | Naïve Bayesian Classifier |
| OECD | Organization for Economic Co-operation and Development |
| $p_{target}$ | Possible Target Item |
| $p_{aq}$ | Prediction on Item $q$ for User $a$ |
| PC | Pearson Correlation |
| PCA | Principle Component Analysis |
| PPCF | Privacy Preserving Collaborative Filtering |
| PPCFADD | Privacy Preserving Collaborative Filtering on Arbitrarily Distribute Data |

| | |
|---|---|
| PPDCF | Privacy Preserving Distributed Collaborative Filtering |
| PPV | Positive Predictive Value |
| PrivateSims | Private Similarity Computation Protocol |
| PLSA | Probabilistic Latent Semantic Analysis |
| $q$ | Target Item |
| $r_{max}$ | Maximum Possible Rating |
| $r_{min}$ | Minimum Possible Rating |
| RDMA | Rating Deviation from Mean Agreement |
| RF | Random Filling |
| RPT | Randomized Perturbation Techniques |
| RRT | Randomized Response Techniques |
| SVD | Singular Value Decomposition |
| SVM | Support Vector Machine |
| TMF | Target Model Focus |
| TN | True Negative |
| TP | True Positive |
| $u_a$ | Active User |
| VDD | Vertically Distributed Data |
| WDA | Weighted Degree of Agreement |
| WDMA | Weighted Deviation from Mean Agreement |

*Dedicated to my dear son Barış*

*and my loving parents, Mesut and Nurten . . .*

# 1. INTRODUCTION

**The truth is incontrovertible. Malice may attack it, ignorance may deride it, but in the end, there it is.**

***–Winston Churchill***

Due to the explosive use of the Internet, the amount of accessible digital information has been growing incessantly. According to business intelligence startup Domo's mind-blowing statistics for 2015; every minute of every day 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 (Kim, 2015). While these statistics prove the staggering increase in online data, they also reveal impossibility of human processing. Although richness of data may be perceived as valuable, actually it puzzles individuals while making decisions, and discovering appropriate information, and thus brings the "information overload problem" with itself (Berkovsky and Freyne, 2015). In order to cope with information overload, and to simplify information discovery, computer-based personalized applications, which take into account the preferences and demands of individuals, are required (Bobadilla et al., 2013). One type of such applications that has become very popular both in research community, and industry is recommender systems (Ekstrand et al., 2011; Berkovsky and Freyne, 2015).

Recommender systems are emerging software tools, which provide users with the ability to find interesting items among many alternatives (Burke, 2002; Ricci et al., 2015). These systems are beneficial for both customers and online-vendors.

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

Recommender systems assist customers to determine the most appropriate products that best meet their needs among the potentially huge range of goods and services available (O'Mahony, 2004). Many customers prefer online shopping, however, not to lose money, waste time, and not to be unhappy by buying something they might not like, customers ask recommendations before deciding on what to buy. In order to help customers to identify the most relevant products with little effort, and hence to achieve high level of customer satisfaction, to promote their sales and profits, and to attract customers's attention, various e-commerce companies integrate recommender systems into their solutions (Schafer et al., 2001). In the past two decades, there has been a lot of research about recommender systems, and a number of ways have been proposed to produce automated referrals including collaborative, content-based, knowledge-based, demographic, or hybrid techniques (Adomavicius and Tuzhilin, 2005; Ekstrand et al., 2011; Bobadilla et al., 2013).

## 1.1. Collaborative Filtering

Collaborative filtering (CF) is the most successful and widely used recommender technology that has been integrated into many e-commerce and online systems for recommendation purposes. The term CF takes place in the literature for the first time in 1992 by the designers of Tapestry project (Goldberg et al., 1992), which is one of the first recommender systems used for e-mail filtering. Through CF, users can get recommendations about a film or music that they wish to buy, or a restaurant, cafe or hotel that they want to go, or on any every day activity that a human can provide a preference (Canny, 2002).

On the basis of the user preferences given to a variety of products, CF algorithms produce desired referrals. The representation of users' preferences on different products may vary according to the data collection method of the system. For example, a company may wish customers to express their preferences with *numerical values* within a rating interval (e.g., 1-5 stars). Some companies consider *ordinal values* as user preferences, where ordinal categories represent the possible level of user opinion (e.g., Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree). Sometimes it is preferable to know whether a customer likes an item or not more clearly, instead of knowing how much the item is liked. In this case, companies ask for *binary values* (e.g., like/dislike, interested/not interested, or good/bad), where positive opinion is represented by 1, and negative opinion is represented by 0. After creating the preference vectors with the desired data collection method, customers send them to service providers. For example, let's $V_A$ be the preference vector of customer $A$, and let's assume that the service provider asks for customer $A$'s preferences on six different products in a 5-point rating scale. In this case, $V_A$ can be as $\{-, 2, -, 4, 3, -\}$, where "-" represents that the user has no preferences on that product. According to $V_A$ it can be stated that customer $A$ did not much like the second product, but she really liked the fourth one. Now, customer $A$ can send $V_A$ to online shopping site to get a recommendation on whether she will like or dislike the first product.

The basic ideas behind CF are that users who like (or dislike) similar items in the past most probably will like (or dislike) other items similarly in the future (O'Mahony, 2004; Goldberg et al., 2001; Yakut and Polat, 2012a), and the preferences of users remain stable and consistent over time (Jannach et al., 2010). Under these assumptions, the main goal of CF systems is to provide referrals for a particular user $u_a$, called the *active user*, based on the preferences of other like-minded users (Sarwar et al., 2001). The general CF process is shown in Figure 1.1. Traditional CF systems first construct a $n \times m$ user-item matrix from the collected data, where $n$ represents the number of users and $m$ represents the number of items in the system. Similarly, $U = \{u_1, u_2, ..., u_n\}$ and $I = \{i_1, i_2, ..., i_m\}$ represent the list of users and items in the system, respectively. Each cell of the user-item matrix is filled with the corresponding user rating if the user has expressed an opinion about that item. The preferences of users can be gathered in one of two ways. Preferences may be explicitly entered by the users of the system through ratings/reviews (*ex-*

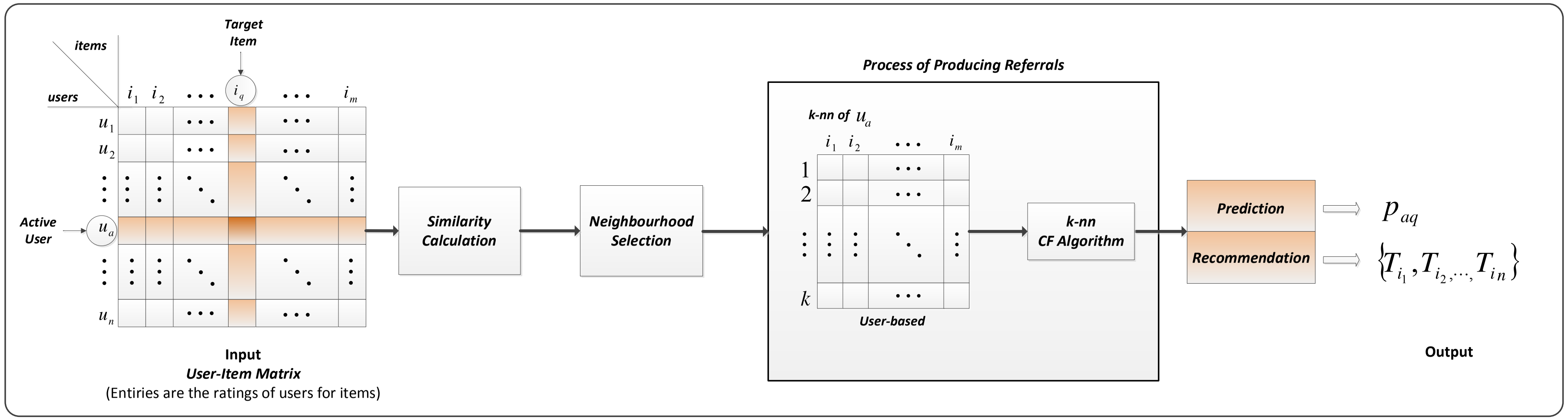


**Figure 1.1.** Collaborative Filtering Process

*plicit rating*), or implicitly inferred from user activities, such as purchase history, web logs, or URL references (*implicit rating*) (O'Mahony, 2004; Ricci et al., 2015; Herlocker et al., 1999). When the active user $u_a$ requests a referral for a target item ($q$), the most similar $k$ users in the system are determined, and then based on the preferences of these neighbours, CF algorithm provides one of the following two basic functionalities (Sarwar et al., 2001; Polat and Du, 2008):

(i) Prediction - is a numerical value $p_{aq}$ showing the estimated likeliness of a target item $q$ for $u_a$.

(ii) Recommendation - is a sorted list of items that will be liked by $u_a$, denoted as *top-N* recommendations.

There are three classes of approaches that are commonly used in CF: *memory-based algorithms*, *model-based algorithms*, and *hybrid approaches.* Memory-based algorithms estimate referrals over the entire user-item matrix (Herlocker et al., 1999; Sarwar et al., 2001, 2000). On the other hand, model-based algorithms first create a model from user-item matrix off-line, and then by using that model generate referrals online (Basu et al., 1998; Breese et al., 1998; O'Connor and Herlocker, 1999; Goldberg et al., 2001). Since model-based algorithms operate over a pre-generated model, they are faster than memory-based algorithms. However, the accuracy of memory-based algorithms is better. Though it is difficult to insert new data into model-based algorithms, it is an easier task in memory-based ones'. Hybrid approaches are proposed to avoid the limitations of memory-based and model-based algorithms, and to improve performance (Su and Khoshgoftaar, 2009).

Memory-based algorithms are among the earliest, and most practically applied CF algorithms, which are also referred to as *neighborhood-based* (Breese et al., 1998), *heuristic-based* (Adomavicius and Tuzhilin, 2005) or *traditional* CF algorithms (Aggarwal, 2016b; Ning et al., 2015). In order to make referrals, memory-based algorithms use the stored user-item matrix directly, and compute similarities between neighbours by using a variety of similarity metrics, such as Pearson Correlation (PC) (Resnick et al., 1994), Cosine Similarity (Breese et al., 1998), Spearman Rank Correlation (Herlocker et al., 1999), Mean Squared Difference (Shardanand and Maes, 1995), or entropy-based uncertainty (Herlocker et al., 2002). In this point, neighbours can be defined either based on users (*user-based*), or on items (*item-based*). In user-based systems, ratings provided by similar users of a target user are used in recommendation. Hence, to detect similar users, similarity is computed between the rows of the user-item matrix (Aggarwal, 2016b). On the other hand, in item-based systems, ratings of the user for similar items of a target item are used in predicting the rating of that user for that target item (Ning et al., 2015). Therefore, to detect similar items, similarity is computed between the columns of the user-item matrix (Aggarwal, 2016b). For large scale online-vendors with huge amounts of users and items, such as Amazon.com (Linden et al., 2003), item-based methods are more preferable due to response speed, on the contrary, others, such as GroupLens (Konstan et al., 1997), and Ringo (Shardanand and Maes, 1995), prefer user-based methods due to accuracy.

Model-based algorithms are proposed to overcome the shortcomings of memory-based approaches such as sparseness and scalability. Although model-based algorithms are faster than memory-based algorithms as they run over a prototype of the original data, they usually suffer in terms of accuracy and it is difficult to fine-tune their parameters. The main idea of these algorithms is to create a summarized model of the data off-line, and then generate referrals online by using that model. The models of user preferences are constructed by utilizing the databases. During the model building process, various data mining and machine learning techniques such as clustering, dimensionality reduction, Bayesian classifiers, decision trees, and association rule mining are applied (Adomavicius and Tuzhilin, 2005). The clustering model approaches CF as a classification problem (Basu et al., 1998; Breese et al., 1998; Ungar and Foster, 1998) to group similar users or items. Dimensionality reduction techniques including Principle Component Analysis (PCA) and Singular Value Decomposition (SVD), are applied to project data into a reduced dimensional space (Sarwar et al., 2000; Goldberg et al., 2001); however, they generally result in loss of meaningful information (Russell and Yoon, 2008). The Bayesian networks are exploited to create a probabilistic model for CF problems (Heckerman et al., 2000). Association rule mining algorithms are used to explore association between co-purchased items (Sarwar et al., 2000). The items are recommended based on the strength of the association between items.

In addition to memory-based and model-based algorithms, researchers also propose hybrid schemes to avoid the limitations of former approaches and improve the performance. Different techniques can be combined in many ways. The Ripper machine learning system is trained with a combination of content data and training data to produce better recommendations (Basu et al., 1998). Content-boosted CF algorithms generate more accurate results in recommendation (Melville et al., 2002). Despite the fact that utilizing content-based filtering with CF increases recommendation accuracy, these hybrid approaches have complexities at the implementation phases (Pazzani, 1999; Burke, 2002). Some researchers propose methods that combine memory and model-based CF algorithms. In a hybrid probabilistic approach, memory and model-based techniques are combined to reduce complexity with decent accuracy (Yu et al., 2004). Some combinations of memory-based and model-based methods provide very good results in terms of accuracy (Koren, 2008).

## 1.2. Challenges of Collaborative Filtering

Although CF has been very successful in information filtering and take an important role in e-commerce applications, this type of algorithms also have a number of limitations. Main problems of CF, as outlined in O'Mahony (2004) and Su and Khoshgoftaar (2009) are as follows:

### 1.2.1. Insufficient data

Providing accurate and reliable referrals is one of the main purposes of CF. Since CF algorithms work entirely on ratings collected from customers on many goods and products, in order to offer precise and dependable referrals, online vendors need to have sufficient amount of data. Seeing that the quality of referrals rely on the quantity of data, e-commerce companies, which have a lot of users, are more likely to generate accurate recommendations than the companies with inadequate number of users. Moreover, poor qualified results that fail to meet user expectations adversely affect sales. Customers may feel uncomfortable, and may decide not to purchase from that vendor anymore. In order to be ahead of their competitors and to be more preferable by users, e-commerce companies should generate more truthful recommendations. However, for some e-companies, especially for the newly established ones, this is not always possible due to lack of qualified data, and it turns out to be a challenge (Kaleli and Polat, 2012b; Polat and Du, 2008).

### 1.2.2. Privacy of data holders

Since shopping habits of customers vary from one another, user preferences might be distributed among two or more companies. For companies particularly having inadequate data, collaborating with other companies might be a solution to provide better referrals, and to relieve the challenges caused by insufficient data. However, privacy is the main obstacle in this cooperation. First of all, rating databases of companies contain detailed information about buying habits and preferences of the customers (Jannach et al., 2010). This personal information is especially valuable in the recommender system domain, and so, companies are afraid of losing their confidential data and capital. Privacy and legal reasons are the other issues that hesitate such collaboration. According to reports published by the Organization for Economic Co-operation and Development (OECD), personal data should be protected by the companies against risks such as unauthorized access, use, modification or disclosure of data (OECD, 2000, 2005).

Due to privacy concerns and other issues induced from these concerns, such as financial fears, legal problems, and data control, companies might hesitate to share their own data with each other (Bilge et al., 2013; Jeckmans et al., 2012). Therefore, privacy is an important challenge, which must be ensured for the cooperation of the companies.

### 1.2.3. Shilling attacks

The main motivation behind CF methods depends on user preferences, and therefore, user preference vectors are very essential components for recommender systems. However, collecting user preferences causes the main weakness that CF algorithms might face, *shilling* or *profile injection attacks* (O'Mahony et al., 2002; Burke et al., 2006b).

Online vendors employ CF algorithms to provide referrals to their customers, so that they can increase their sales and profits (Güneş et al., 2013b). While online vendors make use of CF algorithms to get ahead of their competitors, malicious users, vendors, or rivals might try to insert fake profiles into their user-item matrices in order to affect the predicted ratings, or reduce the performance of the system on behalf of their advantages (O'Mahony et al., 2002). For instance, a malicious vendor who wants to increase her own products' popularity, might create fake user profiles and send them to the central server. Also, with the same method, a vendor might try to decrease popularity of a competitive merchant's products. Due to the inserted fake profiles; quality of the data decreases, while amount of available data increases. Moreover, low qualified or noisy data make accuracy worse, while augmented data, due to inserted fake profiles, make online performance worse (Güneş et al., 2014). Consequently, CF systems might be defenceless against shilling attacks and for the overall success, it is compulsory to handle them (Burke et al., 2005a).

## 1.3. Solutions for Eliminating Challenges

CF is one of the promising research fields in literature, and there are several studies effort to overcome the common limitations of CF algorithms. In this section, proposed solutions for eliminating the aforementioned challenges are explained.

### 1.3.1. Collaboration

Due to need for adequate user data during CF processes, collaboration of online vendors on distributed data while preserving their privacy has become an important topic. In order to make the cooperation of online vendors, even the competing ones, possible on distributed data without disclosing privacy, several Privacy-Preserving Distributed Collaborative Filtering (PPDCF) schemes have been proposed. In the proposed studies, researchers consider collaboration of two or more parties on three different data distribution scenarios, i.e., horizontal, vertical, and arbitrary. In horizontally distributed data, different data holders gather the preferences of different set of users for common set of items. Conversely, in vertically distributed data, different data holders gather the preferences of common set of users for different set of items. Unlike horizontal and vertical data distribution scenarios, in arbitrarily distributed data (ADD), which is combination of multiple horizontal and vertical partitioning, both the set of users and items are common for data holders.

### 1.3.2. Shilling attacks

There are many Internet users and these users may not be all good intentions in terms of feedback and cooperation. For online vendors, it is not easy to distinguish this kind of malicious users. However, for the overall success of CF algorithms, it is crucial to cope with shilling attacks. Researchers present several studies on shilling attacks, which can be grouped into four major classes (Güneş et al., 2014).

One group of research focuses on shilling attack strategies and generating profile injection attacks against CF algorithms. The potentiality of influencing the consequences of CF algorithms by inserting shilling attacks into the system, and the concept of *system robustness* is introduced by O'Mahony et al. (2002). While the amount of knowledge required for the attacker to generate shilling attacks is studied by O'Mahony et al. (2005), Lam and Riedl (2004) discuss the effects of shilling attacks on different aspects, such as utilized algorithm as being recommendation or prediction, and as preferring user-based or item-based algorithms. Moreover, several shilling attack strategies for both memory-based and model-based approaches are identified and designed to manipulate the systems (O'Mahony, 2004; Mobasher et al., 2005, 2006b). Some of these include random attack, average attack, bandwagon attack, segment attack, reverse-bandwagon attack, and love/hate attack.

Other group of research focuses on detecting shilling attacks. According to some researchers, preventing attacks is impossible. However, effects of known attack models can be diminished by detecting attack profiles before generating recommendations or predictions, and hence, by applying attack detection methods, robust CF algorithms can be obtained (Güneş et al., 2014). With this idea, many shilling attack detection algorithms based on classification (Burke et al., 2006b; Williams and Mobasher, 2006; Williams et al., 2007; Zhang and Zhou, 2012), clustering (O'Mahony, 2004; Bhaumik et al., 2011; Chakraborty and Karforma, 2013), variable selection (Mehta et al., 2007a), statistical approaches (Zhang et al., 2006c; Li and Luo, 2011; Gao et al., 2014) are proposed. Among these, classification-based methods, which make use of generic attributes, model-specific attributes, and intra-profile attributes, are very effective, since shilling attack models are designed with respect to certain patterns (Burke et al., 2006a; Mobasher et al., 2007b; Williams et al., 2006a).

Another group of research focuses on analyzing the robustness of various CF algorithms with respect to shilling attacks. Since robustness analysis of CF algorithms indicates how much the algorithm is resistant to shilling attacks, researchers examine the robustness of most commonly used CF algorithms with respect to attacks (O'Mahony et al., 2004a; Lam and Riedl, 2004; Burke et al., 2005a; Mobasher et al., 2005, 2007b). In addition to robustness analysis of numeric ratings-based CF schemes against shilling attacks, a few researcher also analyze the robustness of binary ratings-based CF algorithms (Long and Hu, 2010; Kaleli and Polat, 2013).

The others focus on developing robust CF algorithms against shilling attacks, which directly incorporate the attack-profile removal step absorb into the system. O'Mahony et al. (2004b) propose to avoid attacks through intelligent neighbour selection. Ji et al. (2007) propose a robust architecture, which considers the trust relationships between users in web of trust. Mehta and Nejdl (2008) demonstrate that SVD method can be used to increase the robustness of CF algorithms. Van Roy and Yan (2009, 2010) propose linear and asymptotically linear CF algorithms, which are robust against attacks.

## 1.4. Problem Definition

CF algorithms are one of the indispensable components of e-commerce companies. For these algorithms to produce truthful referrals, sufficient amount of data is required. However, especially for the young companies, it is difficult and time consuming to obtain enough data. Due to shopping habits, user preferences on various products might be distributed between two or more e-commerce sites. Companies holding distributed or inadequate data might think of collaboration to diminish the challenges caused by insufficient data. However, privacy, legal, and financial concerns of the companies might obstruct this collaboration. As stated before, researchers propose several PPDCF solutions for collaboration of online vendors on distributed data while guaranteeing privacy. These studies make the collaboration of online vendors with insufficient data possible, benefitting them by increasing the quality of CF services without compromising privacy. Even though accuracy problems caused by insufficient data are avoided and privacy expectations that obstacles collaboration are established with these studies, existing PPDCF algorithms might be vulnerable to shilling attacks.

In this dissertation, robustness of PPDCF algorithms against shilling attacks is questioned. Although there are numerous works on shilling attacks, so far there is no work that examines PPDCF algorithms in terms of robustness against shilling attacks. If manipulating the outcomes of PPDCF algorithms by injecting fake profiles is possible, then shilling attacks might be an obstacle for collaboration, in spite of the benefits. While PPDCF solutions privately handle accuracy problems caused by inadequate data, shilling attacks might take away the advantages of cooperation. If these systems are vulnerable to attacks, malicious users, who know the collaboration of online vendors, might easily effect the results of PPDCF solutions on behalf of their benefits. Due to the inserted fake profiles, the quality of referrals might be decreased. Also, augmented data might reduce online performance of the PPDCF algorithms. Therefore, online vendors, who are unsure of being subject to shilling attacks, might refrain from cooperation, even they need it to offer more useful recommendation services to customers. Hence, the robustness of PPDCF algorithms proposed on ADD is analyzed in this dissertation.

Moreover, if the proposed PPDCF solutions are defenceless against shilling attacks, in order to make the cooperation of online vendors on distributed data possible, detection of shilling attacks is compulsory. As briefly stated, there are many detection algorithms proposed for enhancing robustness of CF algorithms against profile injection attacks. However, all the existing methods are centralized solutions, which work on whole data. In other words, these methods make use of knowledge about all of the user-item matrix, and whole attack detection process is in control of a single data owner. Due to privacy constraints, in PPDCF solutions collaborating companies can only operate on their own data, yet not on the integrated data. Hence, existing attack detection methods cannot be directly applied to PPDCF algorithms proposed on ADD schemes. Therefore, shilling attack profiles on ADD cannot be identified by employing existing detection methods. Hence, providing solutions for detecting shilling profiles on ADD without jeopardizing data owners' privacy is another point that this dissertation focuses on.

## 1.5. Contributions

Main contributions of the dissertation can be summarized in the following.

A new research problem, which is robustness of PPDCF schemes against shilling attacks, is pointed out. PPDCF schemes, which enable collaboration of online vendors holding insufficient data to provide accurate referrals with privacy, might be subject to shilling attacks. Existing PPDCF solutions have not yet been evaluated in terms of robustness against attacks. Most of the research in shilling attack literature focus on robustness analysis of central server-based systems against shilling attacks, and in all the solutions centralized data is considered. Although, there are some studies that analyze the robustness of privacy-preserving collaborative filtering (PPCF) algorithms against attacks, the privacy of individuals is considered in these solutions (Güneş et al., 2013a,b; Bilge et al., 2014a). However, in PPDCF schemes, confidentiality of data holders is essential. It is obvious that, if online vendors are not sure about being robust against malicious users, they might hesitate to collaborate, even if they need it. Necessity of analyzing PPDCF algorithms in terms of robustness against shilling attacks is testified in this dissertation.

ADD is chosen as the focus, hence the robustness of PPDCF algorithms proposed on ADD are examined. Since shilling attack strategies described in previous studies are designed to be mounted against central server-based systems, in order to manipulate the results of PPDCF algorithms proposed on ADD, new attack strategies that can be applied on ADD are required. Strategies for designing shilling attacks against ADD are discussed. A method that injects attacks into the system by partitioning the attack profile between data holders, which can be employed both for numeric and binary data, is outlined. Empirical studies show that attacks generated by the proposed method are capable of biasing the outcomes of the PPDCF algorithms proposed on ADD.

Robustness analyzes of three state-of-the PPDCF schemes proposed on ADD are performed by means of shilling attacks. In the empirical studies, the proposed shilling attack design methodology for ADD is employed while inserting the well-known shilling attack models into these schemes. Proposed attack design strategy prove to be successful in altering the predictions of the PPDCF algorithms on ADD for cases of both numeric and binary data. Thus, in spite of privacy protection, vulnerability of the PPDCF schemes proposed for ADD against shilling attacks is disclosed in this dissertation (Yılmazel and Kaleli, 2016).

Robustness analyzes of PPDCF algorithms on ADD, also reveal the need for defending mechanisms against attacks. Looking at the results, online vendors who are unsure of being subject to shilling attacks might hesitate to collaborate, even if collaboration is beneficial for them to offer more accurate CF services. Hence, in order to make the cooperation of online vendors on distributed data possible, detection of shilling attacks is compulsory. This dissertation shows that, in order to make online vendors collaborate on distributed data, solutions to detect attacks on ADD without jeopardizing data owners' privacy are required as PPDCF solutions are defenceless against shilling attacks.

In this dissertation, the reasons of why existing shilling attack detection methods cannot be directly employed in PPDCF algorithms on ADD are discussed in details. Moreover, the need of new or distributed versions of the existing attack detection solutions, which can be employed on ADD without jeopardizing data owners' privacy, is explained. Even though there are many solutions proposed for enhancing robustness of CF algorithms against attacks, all the existing methods are for the case where data is collected in one center. Hence, existing attack detection methods work on centralized data, and whole detection process is controlled by a single data holder, who owns all data. Unlike existing methods, when data is arbitrarily distributed between two parties, each party can only operate on her own data, due to privacy constraints. Therefore, existing detection methods cannot be directly applied on distributed data, and shilling attack profiles on ADD cannot be identified.

New detection algorithms, or distributed versions of the existing attack detection solutions are required.

To protect PPDCF algorithms proposed on ADD against attacks, distributed version of a well-known classification-based attack detection method, which can operate on ADD while preserving privacy, is provided through this dissertation. Private protocols are developed to calculate the desired classification attributes collaboratively between two parties. To achieve confidentiality, homomorphic encryption, and random filling techniques are utilized. Empirical analyzes show that with the proposed method, it is still possible to detect attacks on ADD effectively without jeopardizing data owners' privacy.

The need for collaboration in detection of distributed attacks is also exposed. With empirical analyzes it is shown that even if collaborative parts have their own detection mechanisms working on their own sides based on their own data, distributed attacks, which are injected by the proposed attack generation strategy for ADD, cannot be detected. Hence, in order to identify distributed shilling attacks profiles on ADD, collaboration is required.

## 1.6. Outline of the Dissertation

The remainder of this dissertation is organized in the following manner:

- In **Chapter 2** relevant background and preliminaries about shilling attacks are presented. A comprehensive review of shilling attack studies on major research fields are described. Basic shilling attack models subject to many research are briefly explained. In particular, a classification-based shilling attack detection method, and descriptions of required classification attributes are described, which are used in later chapters.

- In **Chapter 3** relevant background on PPDCF algorithms is reviewed. Data partitioning scenarios in PPDCF schemes are described. The definition of privacy, considered in this dissertation, is explained. Privacy protection methods utilized to ensure the declared privacy constraints are discussed.

- In **Chapter 4** state-of-the-art PPDCF schemes proposed for ADD are described.
- In **Chapter 5** a new research problem, which is robustness of PPDCF schemes against shilling attacks, is introduced. Despite privacy, vulnerability of the schemes described in Chapter 4 against shilling attacks is discussed. For injecting attack profiles on ADD, an attack strategy that can be used in generation of distributed adaptations of formerly proposed attack models both for numeric and binary data is proposed. Effects of the attacks injected by applying the proposed attack strategy on PPDCF schemes proposed for ADD are analyzed.
- In **Chapter 6** distributed version of the classification-based shilling attack detection method described in Chapter 2 is developed to defend PPDCF schemes proposed for ADD against shilling attacks injected by the attack strategy introduced in Chapter 5. The need for collaboration in attack detection on ADD is empirically exposed.
- In **Chapter 7** concluding remarks and recommendations for further research are discussed.

# 2. SHILLING ATTACKS

In this chapter, a comprehensive review of shilling attack studies on major research directions are given. Then, the general form of attack profiles are discussed. The most popular and well-known shilling attack models subject to many research, are briefly explained. Finally, a classification-based shilling attack detection method, which is chosen to work on, is described. Descriptions and formulas of all the required classification attributes, which need to be computed, are presented.

## 2.1. Shilling Attack Studies

The possibility of manipulating the outcomes of CF algorithms by injecting malicious profiles into the system database is introduced by O'Mahony et al. (2002). Since then, a lot of research in different aspects are studied on "shilling" or "profile injection attacks", which are grouped as shilling attack strategies and generating profile injection attacks against CF algorithms, detecting shilling attacks, analyzing the robustness of existing CF algorithms, and developing robust algorithms that are intrinsically resistant to attacks (Güneş et al., 2014; Burke et al., 2015; Aggarwal, 2016a).

***Shilling Attack Strategies*** & ***Generating Profile Injection Attacks against CF Algorithms:*** Initially, O'Mahony describes several shilling attack strategies toward CF systems in his doctoral dissertation, in which attacks are generated by inputting bogus data through the normal system interface, and no direct access to database is assumed (O'Mahony, 2004). The experimental results obtained in his dissertation show up the noteworthy vulnerability of the commonly used class of CF algorithms, and the necessity of some domain knowledge for generating successful attacks. Then, O'Mahony et al. (2005) examine the extent of such domain knowledge and discover that even when a small amount of such knowledge is available, it is possible to implement successful attacks. Lam and Riedl (2004, 2005) introduce two basic attack models, namely random and average attacks, and try to explore the answers of four open questions related to the effectiveness of shilling attacks. Burke et al. (2005d) outline major issues in building secure CF systems, and introduce several other attack models, including the consistency, segmented, and bandwagon attacks. Moreover, Burke et al. (2005a) examine the success of bandwagon and popular item attacks against CF systems. Later, Burke et al. (2005b,c) propose the segment attack model, which concentrates on a targeted set of users with similar tastes. In addition to these, reverse bandwagon and love/hate attack models are introduced by Mobasher et al. (2007b) as nuke attacks. Attack models described up to this point are for memory-based CF algorithms. Cheng and Hurley (2009) discuss the design of effective attack strategies targeting model-based CF algorithms, and propose diverse and obfuscated attacks. For reputation-based recommender systems copied-item injection attack is presented by Oostendorp and Sami (2009).

***Detecting Shilling Attacks:*** Since bogus data injected into the system reduce the quality of CF data, and cause to produce unsatisfactory referrals, which will lead user dissatisfaction; for the overall success of the CF system, it is important to cope with attacks. Recently, detection of attack profiles has become a very popular research topic, and several attack detection algorithms have been proposed, which can be categorised as *supervised*, *unsupervised*, or *semi-supervised* techniques.

Some researchers model attack detection as a classification problem, and apply supervised learning methods, which try to discriminate "*Attack*" profiles from "*Authentic*" ones based on the detection attributes calculated for each profile. Burke et al. (2006a,b) introduce a set of classification attributes derived based on the expected characteristics of attack profiles, including both generic attributes, several of which are extended from attributes originally proposed by Chirita et al. (2005), as well as model-specific ones. Mobasher et al. (2006b) introduce two more classification attributes, *Weighted Degree of Agreement* and *Filler Mean Target Difference*, which are particularly effective at detecting segment attacks. By using the mix of known classification attributes with some additional ones, Williams and Mobasher (2006) examine their combined effectiveness at defending against shilling attacks. The effectiveness of these studies in attack detection is evaluated generally with the well-known supervised classification algorithms, such as simple nearest-neighbour classification using $k$NN, decision-tree learning using C4.5, and Support Vector Machine (SVM) (Burke et al., 2006a,b; Mobasher et al., 2006b, 2007b; Williams et al., 2006a; Williams and Mobasher, 2006). Later, He et al. (2010) classify and detect shilling attacks with rough set theory. Zhang and Zhou (2012) apply the ensemble learning approach in attack detection, where the underlying idea of the approach is to improve predictive ability based on relearning the existing knowledge. They propose a meta-learning-based detection algorithm, which contains two training phases, namely base-level training and meta-level training, and uses SVM as the basic learning method in both phases. This method effectively detects profile injection attacks, but suffers from low precision particularly when the attack size is small. Zhang and Zhou (2014) also propose an online method, HHT-SVM, for attack detection based on Hilbert-Huang transform (HHT) and SVM. The crucial point of the algorithm is the HHT-based feature extraction method, which makes it operate online. Only limitation of the method is that the SVM classifier needs to be re-trained offline when new attack types are generated. To improve the performance of attack detection, Morid et al. (2014) applied a $k$NN supervised classification method to the influential users, instead of the whole user set. Zhang and Zhou (2015) propose an ensemble detection model (EDM) by introducing back-propagation neural network and ensemble learning technique to detect profile injection attacks through selecting and integrating parts of the base classifiers using voting strategy. In an other work, Zhang and Chen (2016) illustrate the effectiveness of ensemble method for detecting shilling attacks based on ordered item sequences (EMDSA-OIS), which uses simple majority voting strategy to combine the predictive results of multiple C4.5-based classifiers. Yang et al. (2016) apply a variant of Boosting algorithm, called the re-scale AdaBoost (RAdaBoost) as an attack detection method, which is very effective in some hard scenarios as imbalanced classification.

There are also several studies focus on detecting shilling attacks using unsupervised approaches. O'Mahony (2004) utilizes clustering approach in attack detection as a neighbourhood selection scheme. Su et al. (2005) introduce a similarity spreading algorithm to detect and prevent group shilling. Mehta et al. (2007a) present a simple variable selection algorithm using Principle Component Analysis (PCA), which can be used as a pre-step in any CF algorithm to filter out groups of spam users that are highly correlated. Later, this approach is utilized in Mehta (2007) and Mehta and Nejdl (2009), and a PCA-based and Probabilistic Latent Semantic Analysis (PLSA)-based clustering algorithms are presented for attack detection, among which PCA-based one provides better performance. Bryan et al. (2008) propose the UnRAP algorithm, which also uses clustering in separating fraudulent attack profiles from genuine users, and builds on the strength of $H_v$*-score* as its principal metric. Bhaumik et al. (2011) describe an attribute-based $k$-means clustering approach to identify attack profiles. Lee and Zhu (2012) adopt a multidimensional scaling approach to identify distinct behaviours, and propose to discriminate attackers with clustering-based methods. Zhang et al. (2013) propose two algorithms, CLUTR (clustering by using "trust" to filter out suspicious fake users) and WCLUTR (clustering with weighed similarities derived from "trust"), to combine clustering with trust among users, and show that both are much more robust than traditional user-based CF algorithm against average attacks. Chakraborty and Karforma (2013) consider the problem of attack detection as a problem of outlier detection in the user rating database, and use Partition around Medoid (PAM) clustering algorithm in detecting attack profiles. Bilge et al. (2014b) utilize the idea of bisecting $k$-means clustering in shilling attack detection, which is very successful at detecting bogus profiles generated from basic attack models. Zhang and Kulkarni (2014) utilize a spectral clustering algorithm in shilling attack detection.

Beyond clustering, statistical approaches are also applied in detecting anomalies arise from doubtful ratings. Zhang et al. (2006c) develop a probabilistic approach built on a Singular Value Decomposition (SVD)-based algorithm, where a low-dimensional linear model that describes the rating matrix is competed by maximizing the log-likelihood of ratings. Hurley et al. (2009) propose to use Neyman-Pearson statistical detection in order to identify attack profiles. Li and Luo (2011) build a probabilistic model for attack detection in the framework of probabilistic generative model. An unsupervised detection algorithm based on beta probability distribution, Beta-Protection ($\beta\mathcal{P}$), is proposed by Chung et al. (2013), which follows the characteristics that an ideal algorithm should have: strives to identify as many attackers as possible while keeping as many normal users intact as possible, easy to understand and immune to missing values. Zou and Fekri (2013) develop a probabilistic inference framework, which utilizes the Belief Propagation algorithm for inference, and exploits the target items for attack detection. Zhou et al. (2014b, 2015c) study the use of statistical metrics in detecting rating patterns of attackers, and propose the unsupervised statistical RD-TIA (Target Item Analysis) approach, which finds the set of suspicious attack profiles using the chosen metrics, and then refines the set by target item analysis. The authors also adapted this approach for group attack detection, which is named as DeR-TIA (Zhou et al., 2014a). Bhaumik et al. (2006) introduce an item-based approach, which aims to identify what items may be under attack based on the rating activities of the item. They present two Statistical Process Control techniques; X-bar control limit and Confidence Interval control limit, for detecting items under attack, and a time-series technique for detecting time intervals. Assuming that rating distributions of items over time can reveal the presence of attacks, Zhang et al. (2006a) propose a time series based attack detection method, which uses sample average and sample entropy to construct time series that are appropriate for attack detection. Tang and Tang (2011) present an attack detection method that takes into account the time attributes of user behaviours, namely span, frequency, and mount, which are called as *time SFM factors*.

Gao et al. (2014) propose a revised bottom-up discretized approach based on two common features of all attack models (item abnormality, and attack promptness), and time intervals. Moreover, they utilize the chi square distribution ($\chi^2$) to detect abnormal intervals. Their approach concerns only the ratings of the target item, instead of user profiles; hence it has no correlation with attack models and needs low computational cost. However, they use a fixed time window, which is an effective determinant on the detection rate. To overcome the problems of this method, the same authors propose a dynamic partitioning for time series method based on important points followed by applying $\chi^2$ to find abnormal time intervals (Gao et al., 2015). Zhou et al. (2015b) present the TS-TIA algorithm, which first determines the suspicious profiles by constructing time series approach proposed by Zhang et al. (2006a), and then applies TIA method proposed by Zhou et al. (2014b) to detect anomalies. This algorithm examines the suspected rating segments instead of the whole rating matrix, which makes it less time consuming and complex at detecting items under attacks in big datasets. Xia et al. (2015) present a dynamic time interval segmentation technique based item anomaly detection approach that can detect any attack regardless of the specific attack type through evidence of changes in rating distribution. Zhang and Kulkarni (2013) propose a graph-based detection strategy, where the problem is formulated as finding a maximum sub-matrix in the similarity matrix, which is discovered by transforming the problem into a graph and merging nodes by heuristic functions or finding the largest component.

To be able to benefit from both supervised and unsupervised learning methods, some researchers apply semi-supervised learning approaches in attack detection with an aim of detecting different attack types simultaneously. Wu et al. (2011) and Cao et al. (2013) propose a Semi-supervised learning based Shilling Attack Detection algorithm (*Semi-SAD*), which first trains a Naïve Bayes Classifier (NBC) on a small set of labeled users as the initial detector, and then incorporates unlabelled users with augmented expectation maximization (EM-$\lambda$) to improve the initial classifier.

Wu et al. (2012) present a Hybrid Shilling Attack Detector (HySAD), which introduces MC-Relief to select effective detection metrics, and Semi-supervised Naïve Bayes ($SNB_\lambda$) to precisely separate random-filler model attackers and average-filler model attackers from normal users. There are also some hybrid detection algorithms that combine two or more base detection methods (Burke et al., 2015). Zhou and Zhang (2012) propose a hybrid unsupervised approach based on two unsupervised detection algorithms, namely PCA-based and UnRAP, to detect profile injection attacks. In Huang et al. (2012), a hybrid decision approach based on modified versions of two features proposed by Chirita et al. (2005) and UnRAP algorithm is illustrated. To overcome the class unbalance problems in SVM-based detection methods, Zhou et al. (2015a, 2016) propose the SVM-TIA detection method consisting of two stages; classification based on Borderline-SMOTE method used to alleviate the class unbalance problem of SVM classifier, and target item analysis (Zhou et al., 2014b) to reduce the false positive rate.

***Analyzing the Robustness of Existing CF Algorithms:*** The concept of system robustness is introduced by O'Mahony et al. (2002) as an additional performance measure for recommender systems. O'Mahony et al. (2004d) examine the robustness of memory-based CF systems by means of generally used neighbourhood formation schemes and similarity measures. Burke et al. (2005d) show the vulnerability of widely used $k$-nearest neighbour algorithm. The effects of attack models in the context of user-based CF, and item-based CF are analyzed, respectively by Burke et al. (2005a) and Mobasher et al. (2005). Some studies focus on the robustness of model-based algorithms, and demonstrate the relative robustness and stability of model-based algorithms over the memory-based approaches (Mobasher et al., 2006a; Sandvig et al., 2008). However, Cheng and Hurley (2009) argue that the robustness observed in these studies is due to the fact that the proposed attacks are not targeting the model-based algorithms, and suggest to design attack strategies specific for model-based algorithms. Sandvig et al. (2007) look at the robustness of association rule mining based CF algorithm, and show that Apriori algorithm is more robust compared to $k$NN, $k$-means, and PLSA approaches. Robustness of trust-based CF algorithms, and linear CF algorithms are also studied using real data-based experiments (O'Donovan and Smyth, 2006; Fug-uo and Sheng-hua, 2007; Van Roy and Yan, 2010).

As well as the robustness of numeric ratings-based CF algorithms, a very few researcher study the robustness of binary ratings-based CF algorithms. Long and Hu (2010) compare the robustness of user-based $k$NN and binary $k$NN algorithms under multiple types of profile injection attacks on large dataset. Their empirical results show that binary CF is more robust than actual ratings based CF against attacks. Kaleli and Polat (2013) present the binary forms of basic shilling attack models, and propose a new metric, *ratio shift*, to assess the success of binary attacks. Real data-based experiments indicate the possibility of manipulating the prediction outcomes of NBC-based CF algorithm with the proposed binary versions of the attack models.

A few studies investigate the robustness of PPCD schemes with respect to shilling attacks (Güneş et al., 2013a,b; Bilge et al., 2014a). In order to examine the effects of inserting maliciousness into PPCF databases, Güneş et al. (2013a) present the modified versions of random and average attack models, and investigate the robustness of a memory-based privacy-preserving $k$NN CF scheme in respect of shilling attacks. In another study, Güneş et al. (2013b) extend this work, and describe design methodologies to revise six well-known attack models, so that they can be implemented in privacy-preserving environments. They analyze the robustness of two memory-based PPCF algorithms, $k$NN and correlation threshold-based methods, against the revised attack models, and discover that these systems are also vulnerable to attacks, similar to traditional CF schemes. Bilge et al. (2014a) investigate the robustness of four well-known model-based PPCF schemes with respect to shilling attacks. In a more recent work, a hybrid PPCF scheme is tested in terms of robustness against shilling attacks (Güneş and Polat, 2015).

***Developing Robust CF Algorithms:*** Researchers also study on developing robust algorithms, which are intrinsically resistant to attacks. O'Mahony et al. (2004b,c) propose an intelligent neighbourhood formation and similarity weight transformation schemes for CF systems, which are secure against malicious attacks. Matrix factorization approaches based on SVD are applied in several studies as a robust CF solution (Mehta et al., 2007b; Mehta and Nejdl, 2008; Cheng and Hurley, 2010). Resnick and Sami (2007) present the influence limiter algorithm that is provably manipulation-resistant. Ji et al. (2007) propose a robust architecture, which considers the trust relationships between users in web of trust. Linear and asymptotically linear CF algorithms are developed by Van Roy and Yan (2009, 2010), which are robust against attacks. In more recent works, least median squares estimator based (Zhang and Sun, 2014), kernel function and Welsch reweighted M-estimator based robust CF algorithms are introduced (Zhang et al., 2015). A robust recommendation method based on suspicious user measurement and multidimensional trust is developed by Yi and Zhang (2016). More detailed research on development of robust algorithms can be found in literature (Mehta and Hofmann, 2008; Güneş et al., 2014; Burke et al., 2015; Aggarwal, 2016a).

## 2.2. Shilling Attack Models

A *profile injection attack* or *shilling attack* against a recommender system aims to shift the recommendation results of the system with regard to a single target item by inserting a set of attack profiles (Burke et al., 2006a). In general, the intend of shilling attacks is either to "push" or "nuke" the desired target item. The purpose of *push attacks* is to promote or increase the popularity of the target item, in order to make it look like a good recommendation, when actually it is not (Burke et al., 2015). Contrarily, *nuke attacks* aim to demote or reduce the popularity of the target item, when in reality it is a good option for the customer (Burke et al., 2015). For instance, an attacker might compel the system to give poor referrals regarding the

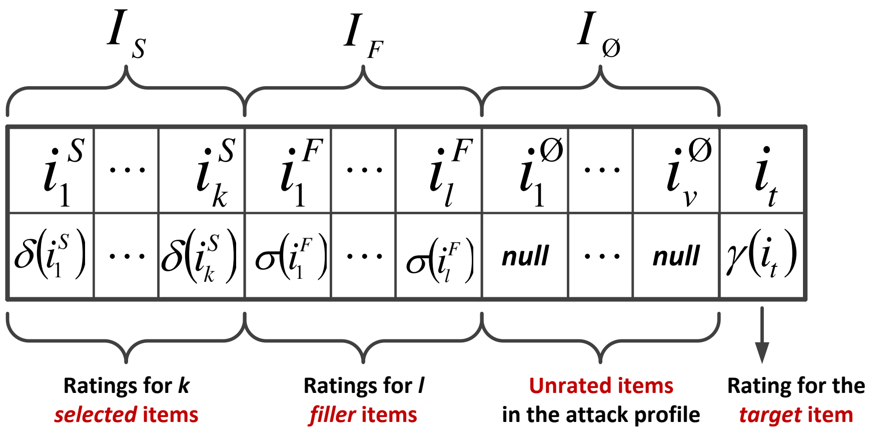

$I_S$
$I_F$
$I_\emptyset$
$i_1^S$ ⋯ $i_k^S$ $i_1^F$ ⋯ $i_l^F$ $i_1^\emptyset$ ⋯ $i_v^\emptyset$ $i_t$
$\delta(i_1^S)$ ⋯ $\delta(i_k^S)$ $\sigma(i_1^F)$ ⋯ $\sigma(i_l^F)$ null ⋯ null $\gamma(i_t)$
Ratings for *k* *selected* items
Ratings for *l* *filler* items
Unrated items in the attack profile
Rating for the *target* item

**Figure 2.1.** General Form of an Attack Profile *(Güneş et al., 2014)

competing products, while aiming to bring forward her own product by adding nuke and push attacks to the system, respectively.

To manipulate the results of any CF system, an attacker can inject attack profiles, whose general form is defined by Bhaumik et al. (2006), and consist of $m$-dimensional vector of ratings, where $m$ is the total number of items in the system. These ratings contracting the attack profile can be divided into four parts as depicted in Fig. 2.1. $i_t$ is the single target item, whose popularity is manipulated by giving a rating as determined by the rating function $\gamma$. Generally, this rating is either the maximum ($r_{max}$ - for *push* attack) or minimum ($r_{min}$ - for *nuke* attack) possible rating in the system depending on the intend of the attack model. $I_S$ 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 special group of items (Williams and Mobasher, 2006). This set receives ratings as determined by the rating function $\delta$, where as 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 more difficult to detect, whose ratings are provided by the rating function $\sigma$. The remaining items that have not been rated compose the null portion of the attack profile indicated as $I_\emptyset$. The thing that gives the characteristics of an attack model, and distinguishes attack models from each other is the strategy used for identifying the set of items in

$I_S$ and $I_F$, and the way the ratings assigned to each of these sets of items, besides the target item; hence the rating functions $\delta$, $\sigma$, $\gamma$.

From the attacker's point of view, some amount of knowledge about the recommender system attempted to attack is required in order to perform an attack, such as knowledge about the algorithm, users, items, and ratings (Lam and Riedl, 2004). While a *high-knowledge attack* requires a detailed knowledge of the ratings distribution in the system's database, a *low-knowledge attack* requires system-independent knowledge (Mobasher et al., 2007b). The most popular and well-known shilling attack models, which are subject to many research, can be briefly explained as follows (Bhaumik et al., 2006; Mobasher et al., 2007a; Bilge et al., 2014a; Burke et al., 2015; Aggarwal, 2016a):

***Random attack:*** In random attack model, the filler items are chosen randomly, and assigned ratings distributed around the system overall mean, which is the mean of all ratings across all items in the user-item matrix. In this attack, the set of selected items is empty, and the target item is set to the maximum possible rating, $r_{max}$, or the minimum possible rating, $r_{min}$, whether the intend of the attacker is push or nuke, respectively. In most of the systems, it is not very difficult to determine the system overall mean by an outsider, thus knowledge required to mount this attack is quite minimal. However, this attack is often not very effective.

***Average attack:*** The only difference between the average attack and the random attack is in the way ratings assigned to the filler items. In average attack model, filler items are assigned ratings corresponding either exactly or approximately to the mean rating for that specific item, across all the users, who have rated that item in the database. As in random attack, the set of selected items is empty, and the target item is assigned either to the maximum possible rating value, $r_{max}$ or the minimum possible rating value, $r_{min}$, depending on the intend. Similar to random attack, filler items are selected randomly; but different from random attack, average attack uses each item's mean, rather than the system overall mean. Since mean of each filler item needs to be known, average attack requires a greater amount of knowledge than the random attack, thus it is harder to implement.

***Bandwagon attack:*** In order to increase the chance of the injected fake profiles being similar to the existing user profiles, a small number of popular items are used as the set of selected items in bandwagon attack model. The term of *popular* items actually refers to the items that are likely to be rated by a large number of users in a positive manner, such as bestseller books, blockbuster movies, or widely used textbooks. If an attacker always rates these popular items in the fake user profile, the predicted ratings of the target user are more likely to be biased by the attack. Hence, in bandwagon attack, selected items and the target item are assigned the maximum possible rating value, $r_{max}$. In addition to selected items, a set of filler items is also used, which are selected randomly, and given rating values distributed around the system overall mean as in random attack. Even though to mount a bandwagon attack some knowledge about the popularity of the items is required to be known, this knowledge is not system dependent; alias it is a public knowledge, and in general, it is not vey difficult to determine the most popular items of any item space. Therefore, it is easy to implement this attack model. Besides, in despite of less knowledge requirements, bandwagon attack can perform as well as average attack.

***Segment attack:*** Segment attack model is designed as an attack model against item-based CF algorithms, which generates neighbours of similar items, instead of neighbours of similar users. The basic idea of segment attack is to push an item to a targeted group of users with specific interest. Particularly in this attack, the aim of the attacker is to promote the target item, not to all users, but to likely buyers, who have liked items that are in similar segment with the target item in the past. In order to mount segment attack, a set of segment items that are presumably to be well-liked by the projected target group of users, needs to be determined. These segment items, and the target item are given the maximum possible rating, $r_{max}$.

Additionally, a set of filler items are chosen randomly, and assigned the minimum possible rating, $r_{min}$, in order to obtain maximum effect. Corresponding segment of an item is usually a common knowledge, thus as in bandwagon attack, it is not hard to predict the most popular items of a user segment.

***Reverse bandwagon attack:*** Reverse bandwagon attack model is a variation of bandwagon attack model, which is especially designed to nuke items. In this attack model, selected items are chosen from *unpopular* or *widely disliked* items, which are the items poorly rated by many users. These items are assigned row ratings together with the target item, in order to increase the probability of generating low predicted ratings for the nuked item. Similar to bandwagon attack, a set of filler items are selected randomly, and given ratings within a rating scale centered on system overall mean. As in the case of bandwagon attack, it is not so difficult to determine these unpopular items outside the system, hereby it does not require any system dependent knowledge.

***Love/hate attack:*** Love/hate attack model is a very simple attack model, which is particularly designed to be a nuke attack with no knowledge requirements. In this attack, there are no selected items. The target item is set to the minimum possible rating value, $r_{min}$, whereas the randomly chosen filler items are set to the maximum possible rating value, $r_{max}$. Despite the ease, it is an extremely effective nuke attack model against item-based CF algorithms.

Attack profiles of these attack models relating to the general attack profile given in Fig. 2.1 are summarized as shown in Table 2.1 (Yılmazel and Kaleli, 2016).

**Table 2.1.** Attack Profile Summary *(Bilge et al., 2014a)

| Attack Type | $I_S$ | | $I_F$ | | $I_\emptyset$ | $i_t$ |
|---|---|---|---|---|---|---|
| | 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 | $r_{max}$ | $I - I_F$ | $r_{min}$ |

## 2.3. Classification-based Shilling Attack Detection Approach

Some researchers treat attack detection as a traditional pattern classification problem in which they work toward to classify profiles as matching known attack models by applying supervised learning methods (Burke et al., 2006a; Williams et al., 2006b,a; Mobasher et al., 2007b; Williams et al., 2007). This classification learning based approach aims to learn to label each profile as either being part of an attack or as coming from an authentic user by using attributes derived from each individual profile (Burke et al., 2006a). For this approach, training data is created as a combination of a number of genuine profiles from an attack-free dataset and attack profiles are generated using the attack models described in Section 2.2 (Williams and Mobasher, 2006). Each profile is labeled as either being part of an attack or as coming from a genuine user (Williams et al., 2007). A binary classifier is then created based on this set of training data using a number of classification attributes (Mobasher et al., 2007b).

In general, these 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 authentic user profile. Attack model-specific attributes are generated to detect the characteristics of a profile, especially 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 reasons for choosing this method to work on through this dissertation are: one of the earliest proposed methods, taken as the baseline algorithm in many studies (Mehta and Nejdl, 2009), the defined attributes are used as they are or with some modifications in many other detection methods (Morid et al., 2014; Zhou et al., 2014a), compared with many other detection methods in several papers (Mobasher et al., 2007b), and concluded to be an effective method. The details of these classification attributes are given in the following subsections based on several studies in literature (Burke et al., 2006b,a; Mobasher et al., 2007b; Williams et al., 2006a; Williams and Mobasher, 2006; Williams et al., 2007).

### 2.3.1. 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 given to the target item, and the distribution of ratings among the filler items are the informers that cause this difference (Mobasher et al., 2007b). As many researchers have postulate (Lam and Riedl, 2004; Chirita et al., 2005; O'Mahony et al., 2004a; Mobasher et al., 2005), 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's why an attribute that captures these irregularities can be descriptive in detecting attack profiles. There are a number of generic attributes proposed in literature for profile classification (Chirita et al., 2005; Burke et al., 2006b; Williams et al., 2006b,a; Mobasher et al., 2007b; Williams et al., 2007).

#### 2.3.1.1 *Rating deviation from mean agreement*

Chirita et al. (2005) introduce the Rating Deviation from Mean Agreement (RDMA) attribute, which is designed to identify attackers through examining the profile's average deviation per item, weighted by the inverse of the number of ratings in the system for that item. The RDMA attribute can be computed as in Eq. 2.1:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \overline{r_i}|}{R_{U,i}}}{N_u} \tag{2.1}$$

where $N_u$ is the number of items user $u$ rated, $r_{u,i}$ is the rating given by user $u$ to item $i$, $U$ is the universe of all users, $R_{U,i}$ is the total number of ratings in the system provided for item $i$ by all users, and $\overline{r_i}$ is the average rating of item $i$ across all users.

#### 2.3.1.2 *Weighted degree of agreement*

Weighted Degree of Agreement (WDA) attribute introduced by Burke et al. (2006a) is a variant of RDMA attribute, which uses only the denominator of the RDMA equation. This attribute captures the sum of the differences of the profile's ratings from the item's average rating divided by the item's rating frequency. It is computed as in Eq. 2.2:

$$WDA_u = \sum_{i=0}^{N_u} \frac{|r_{u,i} - \overline{r_i}|}{R_{U,i}} \tag{2.2}$$

#### 2.3.1.3 *Weighted deviation from mean agreement*

Weighted Deviation from Mean Agreement (WDMA) attribute introduced by Burke et al. (2006a) is strongly based on RDMA attribute, but provides higher information gain. WDMA attribute is designed to identify anomalies, and places a high weight on rating deviations for sparse items. While calculating the WDMA attribute, in order to reduce the weight interrelated with items rated by many users, the number of ratings for an item is squared in the denominator inside the sum of RDMA equation. This attribute can be computed as in Eq. 2.3:

$$WDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \overline{r_i}|}{{R_{U,i}}^2}}{N_u} \tag{2.3}$$

#### 2.3.1.4 *Degree of similarity with top neighbours*

Degree of Similarity with Top Neighbours (DegSim) attribute introduced by Chirita et al. (2005) is based on the average similarity of a profile's top $k$ nearest neighbours, which is calculated as in Eq. 2.4:

$$DegSim_u = \frac{\sum_{v \in neighbours(u)} W_{u,v}}{k} \tag{2.4}$$

where $v$ is the set of top $k$ similar users, which are the nearest neighbours of user $u$, $k$ is the size of this set, in other words, the number of top neighbours, and $W_{u,v}$ is the similarity between user $u$ and $v$ calculated via Pearson Correlation.

#### 2.3.1.5 ***Degree of similarity with co-rated factor***

Degree of Similarity with Co-Rated Factor (DegSim') attribute introduced by Burke et al. (2006a) is slightly different from *DegSim* attribute, and takes into account the number of co-rated items between users in similarity calculation. Let's assume that $I_{u,v}$ be the set of items $i$, such that both user $u$ and user $v$ are given ratings. More specifically, for each item in $i$ user $u$'s rating $(r_{u,i})$ and user $v$'s rating $(r_{v,i})$ are defined, hence not null. $|I_{u,v}|$ is the size of this set, and $d$ is a pre-defined constant. Similar to *DegSim*, profile's top $k$ nearest neighbours are detected, however *DegSim'* discounts the similarity if the neighbour shares fewer than $d$ ratings in common. In this case, the similarity between users $u$ and $v$ is adjusted and *DegSim'* attribute is calculated as in Eq. 2.5:

$$\begin{gathered} W'_{u,v} = \begin{cases} W_{u,v} \times \frac{|I_{u,v}|}{d} & , \quad \text{if } |I_{u,v}| < d \\ W_{u,v} & , \quad \text{otherwise} \end{cases} \\ DegSim_u{}' = \frac{\sum_{v \in neighbours(u)} W'_{u,v}}{k} \end{gathered} \tag{2.5}$$

#### 2.3.1.6 *Length variance*

Burke et al. (2006a) introduce the Length Variance (LengthVar) attribute, which measures how much the length of a given profile varies from the average length of all profiles in the system, where length indicates the number of total ratings. *LengthVar* attribute is computed as in Eq. 2.6:

$$LengthVar_u = \frac{\left|\#ratings_u - \overline{\#ratings}\right|}{\sum_{i=0}^{N}\left(\#ratings_i - \overline{\#ratings}\right)^2} = \frac{\left|N_u - \overline{N}\right|}{\sum_{i=0}^{N}\left(N_i - \overline{N}\right)^2} \tag{2.6}$$

where $\#ratings_u$ is the total number of ratings in the system provided by user $u$, which is same as $N_u$ used in the previous equations. $\overline{\#ratings}$ (or, $\overline{N}$) is the average number of ratings across all users, and $N$ is the total number of users in the system.

### 2.3.2. Model-specific attributes

Researchers have shown that when the profiles are small - containing fewer filler items - generic attributes are insufficient to distinguish a true attack profile from eccentric, but authentic profiles, which results with large false positive rate in classification (Burke et al., 2006a; Mobasher et al., 2006b). In order to reduce the success of shilling attacks, in addition to generic attributes, Williams et al. (2007) propose the use of model-specific attributes, which are designed to detect attack profiles by comparing the similarity of a profile with known attack models.

As mentioned before, attacks can be characterized based on the characteristics of the attack profile partitions: $i_t$ (targeted item), $I_S$ (set of selected items), and $I_F$ (set of filler items). By looking at Table 1, one can easily state the key distinguishing aspects among the attack models. For instance, apart from the others, there is an $I_S$ part in bandwagon, segment, and reverse bandwagon attacks. Specifically, these attack models have a group of items ($I_S \cup i_t$) that are common across all profiles, whereas in the rest of the attack models only the target item $i_t$ is the guaranteed common item. Bandwagon and reverse bandwagon attacks are variants of each other, such that bandwagon is used as a push attack, where as reverse bandwagon is used as a nuke attack. Random and average attacks differentiate in the ratings given to filler items. Moreover, the only difference between random and bandwagon attacks is the use of $I_S$ partition in bandwagon. This is also true for segment and love/hate attacks, but with the ratings of the $I_F$ and $i_t$ items reversed (Williams and Mobasher, 2006).

Model-specific attributes aim to capture the distinctive signature of specific attack models. These attributes are based on partitioning each user profile somehow to maximize the profile's similarity with the one created by a particular attack model. This partitioning can be modelled by splitting each user profile into three sets. The set $P_{u,T}$ contains all the items in user $u$'s profile that are suspected to be targets. According to the intent of the attack, suspected 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 are 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 of these partitions can be used for generating the model-specific detection attributes.

Williams et al. (2007) introduce several measures for detecting the distinctive signatures of attack models.

#### 2.3.2.1 *Average attack model-specific attributes*

Average attack model divides the user profile into three partitions, such that the target item given an extreme rating, the filler items given other ratings, and the unrated items. Mainly the model just needs to select an item as the target and all other rated items turn into fillers. According to the definition of average attack, each filler item will get ratings around the individual mean of that filler item. Thus, except for the single item chosen as the target, a shilling profile generated by an average attacker expected to display a high degree of similarity between its ratings and the average ratings for each item. In other words, there will be a high correlation between the filler items and the item averages.

For finding the optimal partitioning the formalization of this idea is to iterate through all the highly-rated items (depending on the intend of the attack either the highly rated items, or the opposite), select each in turn as the possible target, in this case the rest of the rated items become the filler or non-target items for that partitioning, and compute the mean variance between the non-target items and the average ratings for that items (Williams and Mobasher, 2006). The optimal partitioning is the one where the mean variance is minimized, and the magnitude of the variance is a clue of confidence.

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 $r_{max}$ (or $r_{min}$ according to the intend of the attack). *Mean Variance (MeanVar)* metric needs to be computed for each possible target item ($p_{target}$) of the $P_{u,target}$ set as shown in Eq. 2.7. More formally, first the set of items that are potential targets are defined, which is the set $P_{u,target} = \left\{p_{target} \in P_u,\ \textit{such that}\ r_{u,p_{target}} = r_{max}\right\}$ for push attacks. Then, iteratively, by taking one of the $p_{target}$ item from the $P_{u,target}$ set as suspected target, and assigning rest of the rated items as filler items, $MeanVar(u, p_{target})$ value of the $p_{target}$ item is calculated.

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

$$MeanVar(u, p_{target}) = \frac{\sum\limits_{i \in (P_u - (p_{target} \cup P_{u,\emptyset}))} (r_{u,i} - \overline{r_i})^2}{|P_u - P_{u,\emptyset}| - 1} = \frac{\sum\limits_{i \in P_{u,F}} (r_{u,i} - \overline{r_i})^2}{|P_{u,F}|} \tag{2.7}$$

where $r_{u,i}$ is the rating given by user $u$ to item $i$, and $\overline{r_i}$ is the mean rating of item $i$ across all users. $|P_u - P_{u,\emptyset}| - 1$ describes the number of rated items in user profile $P_u$, which is actually the $N_u$ value, minus 1 to exclude the suspected target item.

In other words, the denominator of the equation is in fact same as the size of the filler item set, $|P_{u,F}|$. Among the calculated $MeanVar(u, p_{target})$ values, whichever yields the lowest value is considered as the optimal partitioning, and $p_{target}$ item that generates this minimum value is selected as the target item $t$. Thus, based on this optimal partitioning, the item $t$ becomes the set $P_{u,T}$, and all other rated items in $P_u$ becomes $P_{u,F}$, which is actually equal to $P_u - (p_{target} \cup P_{u,\emptyset})$. The partitioning sets $P_{u,T}$ and $P_{u,F}$ are used to create the following attributes:

- ***Filler Mean Variance (FMV):*** This is the minimum $MeanVar(u, p_{target})$ value that gives the optimal partitioning, which is discovered above.
- ***Filler Mean Difference (FMD):*** This feature gives the average of the absolute value of the difference between the user's ratings, and the mean rating of the filler items. *FMD* attribute is calculated as shown in Eq. 2.8.

$$FMD_u = \frac{\sum\limits_{i \in (P_u - (p_{target} \cup \ P_{u,\emptyset}))} |r_{u,i} - \overline{r_i}|}{|P_u - P_{u,\emptyset}| - 1} = \frac{\sum_{i \in P_{u,F}} |r_{u,i} - \overline{r_i}|}{|P_{u,F}|} \tag{2.8}$$

- ***Profile Variance (ProfileVar):*** Attack profiles generated by an average attacker are likely to have similar ratings for the filler items, but an extreme rating for the target item. Hence, these profiles need to have very similar within-profile variances. *ProfileVar* attribute shown in Eq. 2.9 captures the variance associated with the profile itself.

$$ProfileVar_u = \frac{\sum_{i \in P_{u,T} \cup P_{u,F}} (r_{u,i} - PMean)^2}{|P_{u,T} \cup P_{u,F}|}, \quad where\ PMean = \frac{\sum_{i \in P_{u,T} \cup P_{u,F}} r_{u,i}}{|P_{u,T} \cup P_{u,F}|} \tag{2.9}$$

*MeanVar* metric needs to be computed twice; ones for push attack, where the set $P_{u,target}$ contains items having rating $r_{max}$, and ones for nuke attack, where the set $P_{u,target}$ contains items having rating $r_{min}$. Hence, these three attributes, *FMV*, *FMD*, and *ProfileVar*, need to be calculated correspondingly according to the optimal partitioning obtained for push and nuke versions of the average attack.

#### 2.3.2.2 ***Random attack model-specific attributes***

Similarly, random attack is also a partitioning attack, which tries out to divide a user profile into the same three partitions discussed in average attack, in which the target partition contains a single rating. However, unlike average attack, the filler items in random attack are assigned values generated randomly within the rating scale with a distribution centered around the mean for all user ratings across all items (Burke et al., 2005a). As in average attack, among all possible choices the optimal partitioning needs to be selected. Williams and Mobasher (2006) propose to use the correlation between a profile and the rating average for each item as a metric to discriminate random attackers. According to the definition of random attack, since the ratings of filler items are generated randomly, low correlation is expected between filler items and the real item means, which is opposite of what is expected for average attack.

First of all, depending on the aim of the attack, the set of potential target items, $P_{u,target}$, which are either the highly rated items (for push attack - $r_{max}$), or the lowly rated items (for nuke attack - $r_{min}$) are determined. Iteratively, each item in the set is selected as the suspected target item, ($p_{target}$), and rest of the rated items ($P_u - (p_{target} \cup P_{u,\emptyset})$) are considered as the filler items, or the $P_{u,F}$ partition of the profile. Then, the correlation between the ratings in the profile given to filler items in $P_{u,F}$ set, and the overall rating for each item is calculated. The $p_{target}$ item, which gives the minimum correlation is selected as the most likely target $t$. Hence $t$ forms the partitioning set $P_{u,T}$, and all other rated items in $P_u$ forms the set $P_{u,F}$, which is expected to be the optimal partitioning. Obtained minimum correlation is called the ***Filler Average Correlation (FAC)***, and used as a classification attribute in detecting random attackers (Burke et al., 2006b).

The other classification attribute used in random attack detection is the ***FMD*** attribute (Burke et al., 2006b). By using the optimal partitioning sets $P_{u,T}$, and $P_{u,F}$ discovered above, *FMD* attribute is calculated as in Eq. 2.8.

Same as in average attack, *FAC* and *FMD* attributes need to be computed twice based on the optimal partitioning obtained for push and nuke versions of the random attack.

#### 2.3.2.3 *Group attack model-specific attributes*

Group attack model-specific 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 ($I_F$), which is the case in bandwagon and segment attack models. Therefore, in group attacks partitioning of a user profile is done differently from average and random attacks. In group attack partititoning, all items in $P_u$ that are given the maximum rating (or the minimum rating 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. Hence, model-specific attributes for detecting both bandwagon and segment attack models are calculated based on this partitioning, which is given in Eq. 2.10 (Williams et al., 2007).

$$P_{u,T} = \{i \in P_u, \ \mathit{such\ that}\ r_{u,i} = r_{max}\} \tag{2.10}$$

$$P_{u,F} = P_u - (P_{u,T} \cup P_{u,\emptyset})$$

For bandwagon attacks, investigation of the filler ratings is identical to the random attack model. As in random attack, ***FAC*** and ***FMD*** attributes are need to be generated, but in this case group attack partitioning given in Eq. 2.10 is 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. *FMTD* attribute is computed as in Eq. 2.11.

$$FMTD_u = \left| \left( \frac{\sum_{i \in P_{u,T}} r_{u,i}}{|P_{u,T}|} \right) - \left( \frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|} \right) \right| \tag{2.11}$$

where $i$ represents the items in target partition, or the set $P_{u,T}$. $r_{u,i}$ is the rating given by user $u$ to item $i$, and $|P_{u,T}|$ is the length of the $P_{u,T}$ set. Similarly, $k$ represents the items in filler partition, or the set $P_{u,F}$. $r_{u,k}$ is the rating given by user $u$ to item $k$, and $|P_{u,F}|$ is the length of the $P_{u,F}$ set. The overall average FMTD value ($\overline{FMTD}$), calculated among all user profiles is then subtracted from $FMTD_u$ as a normalizing factor.

***Group Filler Mean Variance (GFMV)*** is an other attribute used for detecting segment attacks. *GFMV* is the variance of the filler items identified by the group attack partitioning, and is calculated as shown in Eq. 2.12.

$$GFMV_u = \frac{\sum_{i \in P_{u,F}} (r_{u,i} - \overline{r_i})^2}{|P_{u,F}|} \tag{2.12}$$

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 $|P_{u,F}|$ is the length of the $P_{u,F}$ set.

### 2.3.3. Intra-profile attributes

All of the classification attributes mentioned so far have concentrated on inter-profile statistics, which take into consideration characteristics within a single profile (Williams et al., 2007). In fact, a single profile cannot actually effect the recommender system, and for this reason attackers need to inject multiple shilling profiles in order to cause a significant bias (Williams and Mobasher, 2006). Williams et al. (2006a) introduce 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 (Mobasher et al., 2007b).

$$TMF_u = \max_{j \in P_T} F_j, \; where$$

$$F_i = \frac{\sum_{u \in U} \Theta_{u,i}}{\sum_{u \in U} |P_{u,T}|}, \; and \tag{2.13}$$

$$\Theta_{u,i} = \begin{cases} 1, & \text{if } i \in P_{u,T} \\ 0, & \text{otherwise} \end{cases}$$

To calculate *TMF* attribute of a user profile, first $F_i$, which is the degree of focus on a given item $i$, 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* attribute is computed as in Eq. 2.13.

# 3. PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING

In this chapter, data partitioning scenarios proposed in privacy-preserving distributed collaborative filtering works are explained. Then, studies in literature on privacy-preserving distributed collaborative filtering are described. The definition of privacy that has been followed all through the suggested methods in this thesis is given. Finally, privacy protection methods, which are utilized to achieve privacy, are explained.

## 3.1. Data Partitioning in Distributed Data

Success of a CF system strongly depends on having adequate number of user data. Otherwise, accuracy of the produced referrals might not satisfy customers' expectations, and as a result, reputation of such systems can decrease. Conversely, it is not always possible to collect sufficient amount of customer preferences. Especially newly established e-companies might have trouble to offer recommendation service to their customers due to lack of qualified data (Kaleli and Polat, 2012b). In order to overcome this challenge, online vendors, who own limited number of ratings, may decide to collaborate with other online vendors to produce better referrals. By sharing their customers information with other vendors, they may produce more truthful and dependable referrals, which cause them to be more preferable by the customers. Hence, joint data is beneficial for online vendors to increase their sales.

Due to customers' unique shopping routines, they may prefer to buy different products from different sites. On account of this, preferences of users on some items, and values given to items by some users might be split among multiple sites. Data collected for referral purposes might be distributed between two or more parties, and the distribution of data among these parties can be horizontal, vertical or arbitrary.

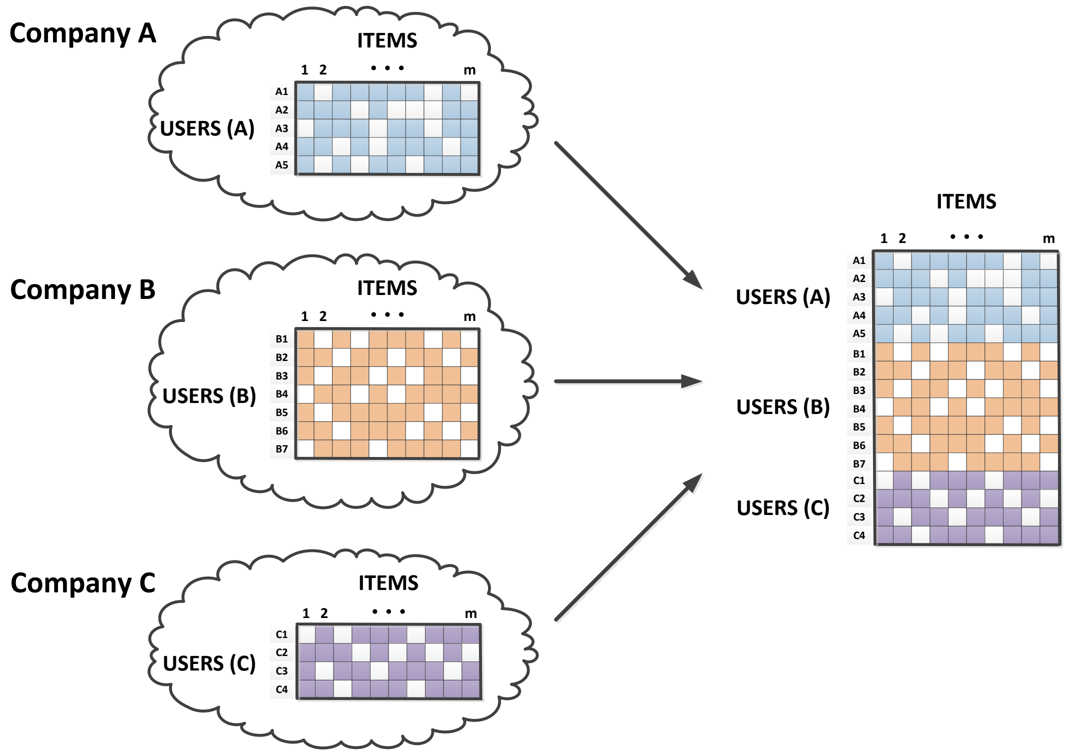


**Figure 3.1.** Horizontally Distributed Data

***Horizontally Distributed Data (HDD):*** Since there are a lot of e-commerce sites, and a specific product can be sold in more than one site, customers have the opportunity to buy products from different online vendors they preferred. HDD is such a data distribution that, *two or more different companies hold disjoint sets of users' ratings for the same items.* An example of HDD between three companies, which are marketing exactly the same set of products, namely *Item 1, 2, ..., m*, but owning completely different sets of customers, is given in Fig. 3.1. For instance, for *Item 2* it can be observed that, while user *A2* prefers to buy *Item 2*, from *Company A*, user *B2* choses to buy the same product from another site, namely *Company B.* Yet, another user, *C2*, likes better to buy this same item, *Item 2*, from *Company C.* Thus, in horizontal partitioning, different companies gather the preferences of different users for the same products.

***Vertically Distributed Data (VDD):*** VDD is such a data distribution that, *two or more different companies have disjoint sets of items' ratings collected*

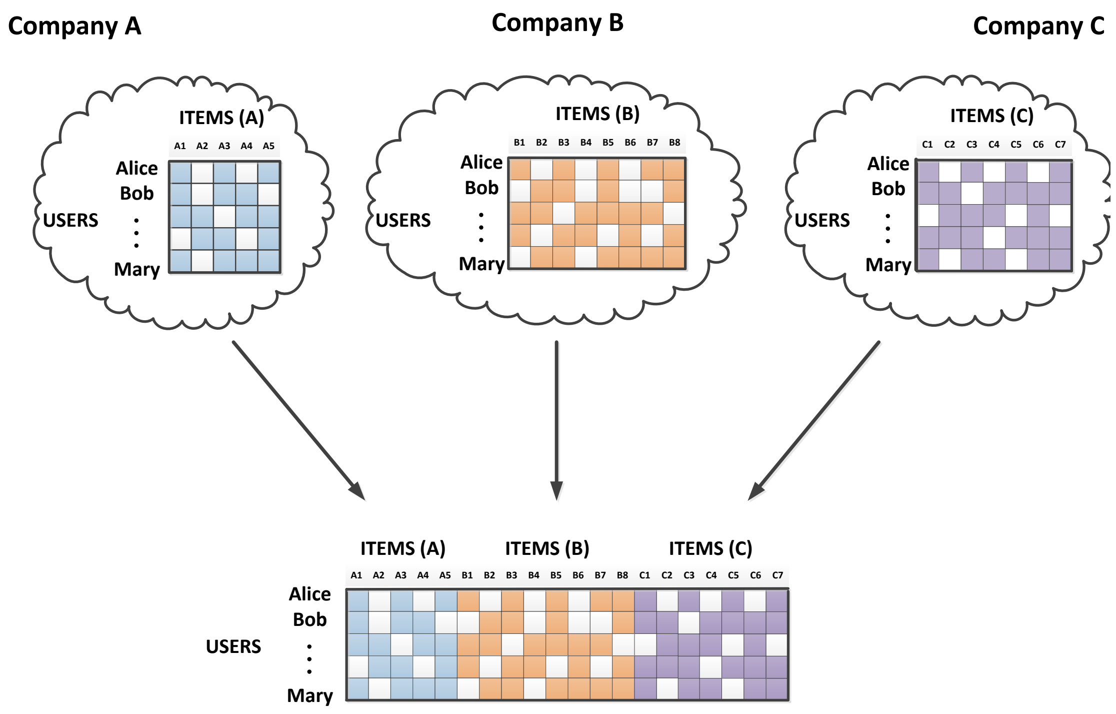


**Figure 3.2.** Vertically Distributed Data

*from the same users*. An example of VDD between three companies, which are owing exactly the same set of customers, namely *Alice, Bob, ..., Mary*, but marketing in different business sectors, so owning completely different sets of products, is given in Fig. 3.2. Let's assume that *Alice* is a user, who mostly prefers shopping from these three companies. However, Alice usually buys her clothes from *Company A*, her shoes from *Company B*, and for electronic goods Alice prefers shopping from *Company C*. As *Alice*, other users may also have similar routines, hence preferences of some specific users on different products might be distributed among various sites. Thus, in vertical partitioning, different companies gather preferences of same users about different set of products.

***Arbitrarily Distributed Data (ADD):*** ADD is a special case, which can be seen as a combination of multiple horizontal and vertical partitionings. As Jagannathan and Wright (2005), who introduced the idea of ADD, have stated, unlike HDD and VDD, in ADD, there is no simple pattern of how data is distributed between two parties. *Each rating can belong to either same or disjoint set of users*

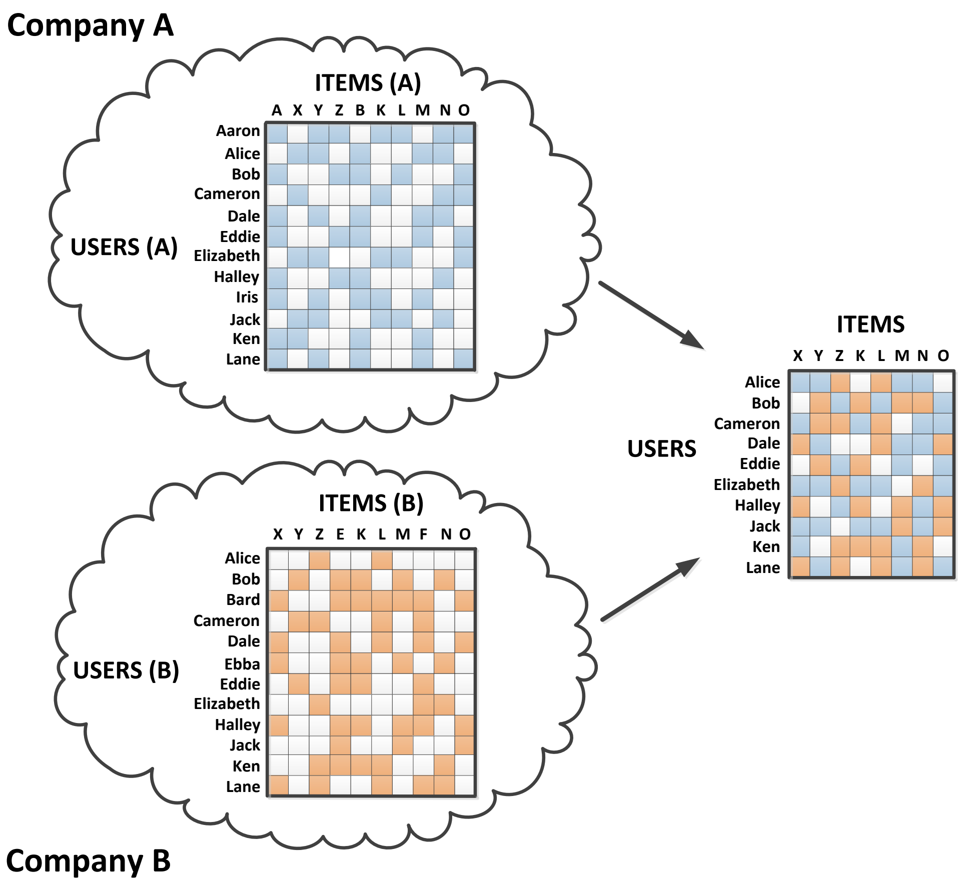


**Figure 3.3.** Arbitrarily Distributed Data

*for either same or disjoint set of items* (Yakut and Polat, 2012a). That is to say, different disjoint portions are held by different parties in ADD. Two online vendors, *Company A* and *Company B*, might end up with ADD as exemplify in Fig. 3.3. As can be seen, the referrals given by the same users to a set of items that are common in both companies forms ADD. The only assumption in ADD is that there are no overlapping ratings, which means, a user can rate an item either in *Company A* or *Company B*, but not in both. It is not rational for a user to give preferences for all items in the system, for that reason there are also unrated cells, as in VDD and HDD. ADD is a more general case than HDD and VDD, and this makes it better suited to real-word settings. Moreover, protocols produce for ADD can be put into practical use for horizontal and vertical partitioning cases (Jagannathan and Wright, 2005).

Combining data horizontally, vertically or arbitrarily has several advantageous for both customers and online-vendors. First of all, with joint data better recommendation services can be offered, since joint data can solve the insufficient data problem. Hence, it is more likely to provide truthful predictions from combined data than the results calculated from split data sets alone. For online vendors to be able to make more money, it is very important to produce accurate referrals to their customers, thus, combined data is beneficial for them.

## 3.2. Privacy-Preserving Distributed Collaborative Filtering Studies

Although collaboration is advantages for online vendors, due to privacy, legal, and financial reasons they may not want to collaborate. Collected data are considered as companies' confidential data, and they add value to the online shopping site on the market place, thus, is very valuable, and should not be lost to a competitor site. Moreover, according to regulations about online shopping, customer data should only be under the control of the site that holds it, and it should not be shared with third parties without the users' or agencies' knowledge, thus transferring them may cause legal problems. Because of these, even if online sites have insufficient customer data, without privacy guarantee they refrain to collaborate with similar vendors.

If privacy measures are provided, online vendors can feel more comfortable and collaborate. Providing privacy measures is essential to perform referrals on joint data. Privacy is also important for customers point of view. Without privacy protection, they do not want to provide their true preferences, or refuse to provide data at all, but the quality and the amount of data collected are significant for the overall success of the CF system. Researchers propose PPDCF methods enabling data holders' collaboration without jeopardizing their privacy (Kaleli and Polat, 2012a; Yakut and Polat, 2012a). As previously mentioned, data distribution scenarios might be horizontal, vertical and arbitrarily in PPDCF systems, and data can be distributed between two or more parties.

Studies based on distribution of data between two parties include the following. Polat and Du (2005c) present a scheme that makes it possible for two parties to produce binary ratings-based top-$N$ recommendations with decent accuracy on HDD without greatly jeopardizing data owners' privacy. Same authors also focus on VDD, and provide a solution for two parties to conduct filtering services using combined numerical data without violating confidentiality (Polat and Du, 2005b). Kaleli and Polat (2007a) discuss how to provide CF services using Naïve Bayesian Classifier-based CF algorithm on horizontally or vertically distributed binary data between two parties with privacy. The provision of recommendations based on distributed data while protecting data owners' privacy has been investigated for the first-time by Polat and Du (2008). Their solution enables two data holders to produce binary rating based top-$N$ recommendations on HDD or VDD, while making it possible to find an equilibrium among accuracy, privacy, and efficiency. Yakut and Polat (2010) examine how to provide singular value decomposition-based referrals on HDD or VDD while guaranteeing privacy, and introduce a model-based PPDCF scheme. Researchers also offer solutions for arbitrarily distributed case, which is more general and common than horizontal or vertical distribution of data. The first work for providing predictions on ADD while preserving data owners' privacy is proposed by Yakut and Polat (2012a), in which they apply item-based CF techniques on ADD between two parties. In another work, Yakut and Polat (2012b) offer a privacy preserving Naïve Bayesian Classifier (NBC)-based CF solution to produce binary referrals on ADD. Yakut and Polat (2012c) introduce a special and much simpler case of ADD, which they named as cross distributed data. They examine the applicability of hybrid CF-based approach to produce predictions for single items having numerical rating values on CDD focused on data owners' privacy, and prediction quality.

Researchers also introduce various solutions that allow collaboration of more than two parties in order to produce more accurate referrals from distributed data. Hsieh et al. (2008) introduce an ElGamal homomorphic encryption-based algorithm to merge recommender systems' databases without disclosing customer privacy. Later, Hsieh (2011) propose a cryptographic method for securely joining recommender systems, which adopts scalar product protocol. To solve confidentiality problem of data holders Zhan et al. (2008) employ scalar product protocol to produce recommendations from distributed data. In another work, Zhan et al. (2010) address how to avoid privacy disclosure in cooperation of online vendors on HDD. They compare the major cryptology approaches, and propose an efficient scalar product protocol based privacy-preserving method. Basu et al. (2011a,b) present a privacy-preserving item-based CF scheme through the use of an additively homomorphic public-key crypto-system on the weighted Slope One predictor. They show its applicability on both HDD and VDD between multiple parties. Basu et al. (2012a) show the feasibility of privacy-preserving prediction services on a cloud platform. They implement and analyze the proposed PPDCF scheme on real cloud platforms (Basu et al., 2012c, 2011c, 2013). Kaleli and Polat (2012a,b) introduce solutions for providing self-organizing map clustering-based predictions for HDD or VDD among multiple parties without greatly exposing data holders' privacy. A trust-based multiparty solution for VDD is also introduced by Kaleli and Polat (2011). Same authors also study how to make binary rating-based referrals using Naïve Bayesian Classifier, when data is distributed among multiple parties without jeopardizing privacy (Kaleli and Polat, 2015).

## 3.3. Privacy Definition

Privacy is a concept, which is hard to be defined exactly, and complicated to put into certain boundaries. In the context of CF, it has various meanings according to different viewpoints. In this thesis, privacy is defined as follows: *The parties should not be able to learn the true rating values and the rated and/or unrated items held by each other during collaborative work.* Hence, actual rating values and the rated items are considered as confidential. True rating values are actually the opinions of the users' on commercial products, and they can be used to profile the customers. Therefore, for the collaborating companies true ratings are intimate. Furthermore, the products rated by the customers' are also considered as the private data of the company, because price discrimination and special advertisements can be made for unrated ones. E-companies do not want to disclose their confidential data to others while producing collaborative tasks. Different from the true ratings and the rated items held by a company, user and item IDs are regarded as public data, thus sharing them does not contravene privacy. Moreover, collaborative companies are assumed to be *semi-honest*, which means that they obey the protocols properly, but they may try out to gather as much private data as possible by resolving the inputs and the outputs. Besides, it is also assumed that non-overlapping ADD occurs between two companies, which means any user can vote any item only once and only on one of the sides, but not on both. Consequently, proposed protocols should enable cooperative work on ADD within boundaries of the specified confidentiality constraints.

## 3.4. Privacy Protection Methods

In order to achieve privacy in CF schemes, several privacy protection methods are introduced (Bilge et al., 2013). Among all, *Randomization* and *Cryptographic Techniques* are two major methods, which are mostly applied. Private protocols proposed through this thesis, also utilize ***Random Filling (RF)*** method, which is one type of randomization technique, and ***Homomorphic Encryption (HE)***.

Randomization is actually a privacy protection technique used for hiding or masking confidential data. Randomization methods attempt to preserve privacy either by perturbing original ratings, or by masking unrated item cells. The well-known types of perturbation-based randomization methods applied, respectively, in numeric, and binary ratings-based PPCF schemes, are Randomized Perturbation Techniques (RPT) (Zhang et al., 2006b; Bilge and Polat, 2011, 2012; Gong, 2011; Basu et al., 2012b), and Randomized Response Techniques (RRT) (Polat and Du, 2006; Kaleli and Polat, 2007b), which hide true ratings by adding random numbers to original data. Besides perturbing real data, in some cases masking the unrated item cells is also crucial for privacy. Random filling is such a randomization method, which is commonly employed in PPCF schemes with an aim of masking the unrated item cells by inserting some noise (Bilge et al., 2013). In RF method, collaborative companies selectively or uniformly randomly choose some of their empty cells, and fill them with fake or default ratings. There are several ways for fake rating determination. One of them is filling the unrated item cells with random numbers (Polat and Du, 2005d,a, 2007; Dokoohaki et al., 2010). Personalized ratings, such as user mean or item mean, or non-personalized ratings, like system mean can also be used to fill the empty cells (Kaleli and Polat, 2012b, 2010; Yakut and Polat, 2010, 2012c,a).

Secure two-party computation enables two parties to jointly compute any function on their inputs without divulging to either party anything no more than the correct output (Yao, 1982; Lindell and Pinkas, 2009). In order to perform secure computations on ADD, cryptographic techniques, especially homomorphic encryption, are utilized when ever required in the thesis. Paillier (1999)'s crypto-system is preferred for protecting data holders' confidential data. The underlying reason is Paillier crypto-system provides faster encryption and decryption algorithms compared to its alternatives, and forestalls many of the drawbacks of the earlier homomorphic crypto-systems (Pedersen et al., 2007).

Paillier crypto-system is additively homomorphic over plain-texts, and also allows for multiplication of plaintext by a constant. Suppose that $\xi$ is an encryption function, $\mathfrak{D}$ is a decryption function, and $K$ is a public key. Plain-texts, $x$ and $y$, are the private data values that will be encrypted, hence, $\xi_K(x)$ and $\xi_K(y)$ are the cipher-texts of them, respectively. According to Paillier crypto-system, the product of two cipher-texts will decrypt to the sum of their corresponding plain-texts as in Eq. 3.1.

$$\mathfrak{D}\left(\xi_K(x) \times \xi_K(y)\ mod\, n^2\right) \equiv x + y\, mod\, n \tag{3.1}$$

Moreover, each cipher-text can be homomorphically multiply with a plaintext as in Eq. 3.2.

$$\mathfrak{D}\left(\xi_K(x)^y\ mod\, n^2\right) \equiv xy\, mod\, n \tag{3.2}$$

In addition to these, Paillier crypto-system has *self-blinding property*, which is the ability to change one cipher-text into another without affecting the plaintext. This can be accomplished by multiplying the cipher-text with $R^N$, where $R$ is a random integer value and $N$ is modulus of the operated public crypto-system (Memiş and Yakut, 2014), as in Eq. 3.3.

$$\mathfrak{D}\left(\xi_K(x) \times R^N mod\, n^2\right) \equiv x \tag{3.3}$$

Overtime, researchers proposed various extensions to Paillier's crypto-system, and described several cryptographic sub protocols for different operations, such as equality/inequality comparison, division by a constant, absolute value, subtraction and negation (Damgård and Jurik, 2001; Parkes et al., 2008). In a more recent work, Dahl et al. (2012) proposed two new protocols for solving the problem of secure integer division, which is performed on a single party where the numerator and the denominator are encrypted values. This secure integer division protocol is also employed, when required in the private protocols proposed through the thesis.

# 4. PRIVACY-PRESERVING DISTRIBUTED COLLABORATIVE FILTERING SOLUTIONS ON ARBITRARILY DISTRIBUTED DATA

In this chapter, three state-of-the-art PPDCF schemes proposed on ADD, which can be categorized as the one focused on private collaboration of parties, when data is numeric, or when data is binary on ADD, besides when data is numeric, but data distribution is a special case of ADD, are briefly explained.

## 4.1. Arbitrarily Distributed Data-Based Recommendations with Privacy

Yakut and Polat (2012a) propose a privacy-preserving scheme to provide predictions for a single item on ADD. They utilize the item-based algorithm proposed by Sarwar et al. (2001). An overview of their proposed method including off-line and online phases is given in Figure 4.1.

At the beginning of the off-line phase, firstly, the parties mask their data by filling some of the unrated item cells in their database with fake ratings in order to protect their own privacy. As a second step, the parties employ private mean estimation and length estimation protocols. Item similarities are estimated with private adjusted cosine estimation protocol and a neighbourhood model is constructed by determining a threshold value for similarities.

Online phase is triggered when $a$ sends her ratings vector and $q$ to one of the parties which acts as master party during collaboration. After masking and normalizing $a$'s preferences, master party collaborates securely with the other party to provide prediction by utilizing constructed model.

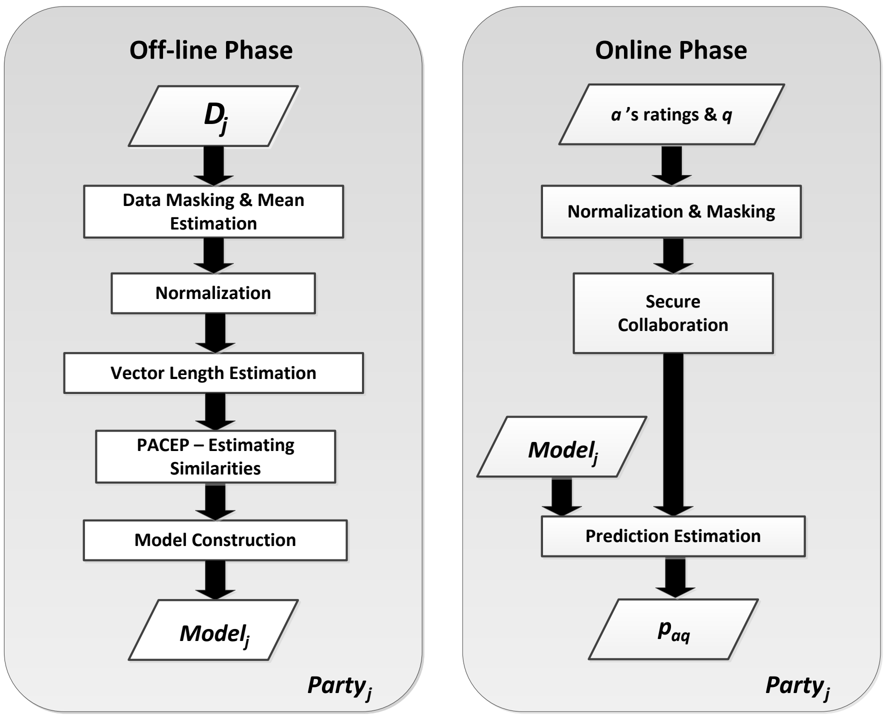


**Figure 4.1.** An Overview of the Proposed Scheme *(Yakut and Polat, 2012a)

Yakut and Polat (2012a) analyze their scheme in terms of privacy, accuracy, and performance. They show that their scheme can be used to provide accurate item-based recommendations efficiently on ADD while providing data owners' confidentiality.

## 4.2. Privacy-Preserving Hybrid Collaborative Filtering on Cross Distributed Data

In another study, Yakut and Polat (2012c) introduce Cross Distributed Data (CDD) concept, which is a special and simpler case of ADD. The authors assume that there can be some special conditions for ADD, where ratings of a user are partitioned as in vertical partitioning scenario, which they named as CDD. CDD is schematized in Figure 4.2 in which, company A has the ratings of the users from 1 to $n_1$ for the items from 1 to $m_1$, which is the upper left part of the user-item matrix, named as $D_1$, and also the ratings of the users from $n_1 + 1$ to n for the items from $m_1 + 1$ to m, which is the lower right part of the matrix, named as $D_4$. Company B has the

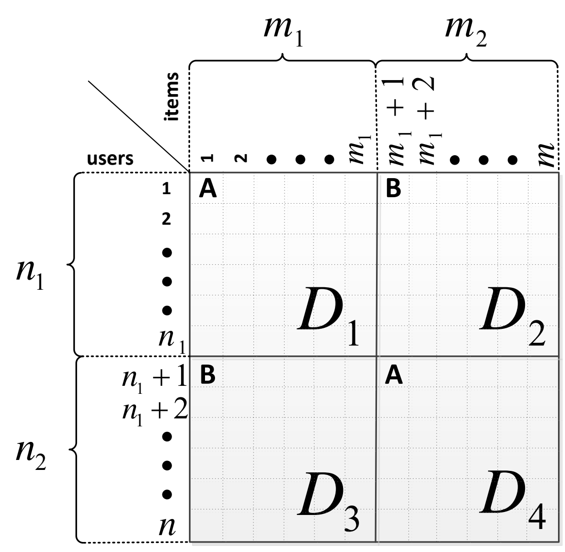


**Figure 4.2.** Cross Distributed Data *(Yakut and Polat, 2012c)

remaining ratings, which are the upper right and lower left parts of the user-item matrix, namely $D_2$ and $D_3$ in the figure, respectively.

In this study, Yakut and Polat (2012c) examine applicability of hybrid version of nearest neighbour CF approach and focus on protecting data owners' confidentiality while producing accurate recommendations. The proposed solution consists of two stages, which are off-line model construction and online prediction estimation. In off-line model construction process, necessary computation functions, such as mean, deviation from mean normalization, and the cosine similarities between items, are computed privately on CDD. Then, based on the computed similarity values, best similar items to each item are required to be chosen to construct a model. Likewise, in the online prediction estimation process, user similarities are computed using cosine measure in a secure manner. Then, the neighbours of the active user is determined. Finally, a referral is estimated by using a prediction algorithm.

Their experiments confirm that the proposed scheme produces accurate predictions efficiently while maintaining data owner's privacy. Besides, online extra costs due to privacy concerns are negligible for the proposed schemes, which makes it preferable for online vendors.

## 4.3. Estimating NBC-Based Recommendations on Arbitrarily Partitioned Data with Privacy

Instead of numeric ratings, some recommender systems collect binary votes, since for some applications it is more important to know whether a user liked or disliked an item, rather than knowing how much the product it liked by that user. Apart from the above studies, which allow collaboration of parties on ADD having numeric ratings, Yakut and Polat (2012b) also focus on private collaboration of parties having binary ratings.

Yakut and Polat (2012b) introduce protocols that enables NBC-based binary predictions on ADD. In order to improve online efficiency, the construction of NBC-based prediction model is carried out off-line. For model generation, two arguments, likelihood and priori probabilities, are required to be computed without jeopardizing data owners' confidentiality. Therefore, the authors propose Numerator of Likelihood Computation Protocol and Denominator of Likelihood Computation Protocol for estimating the likelihood probabilities in a secure manner, and also Privately Priori Estimation Protocol for determining priori values, while satisfying confidentiality requirements of the parties. In the online phase, parties collaborate securely by using the proposed Online Recommendation Estimation Protocol for providing predictions to their customers.

Experiments of the study prove that proposed scheme provides accurate predictions efficiently on partitioned data without violating confidentiality. Hence, this study makes the collaboration of online-vendors with binary ratings possible for suppling better CF services with privacy.

# 5. ROBUSTNESS ANALYSIS OF ARBITRARILY DISTRIBUTED DATA-BASED RECOMMENDATION METHODS

In this chapter, a new research problem is pointed out, which is the robustness of PPDCF schemes against shilling attacks. ADD is chosen as the focus of the study, hence the robustness of the methods proposed for PPCF on ADD (PPCFADD) are considered. First, an attack strategy that can be applied by an attacker in generation of distributed adaptations of formerly proposed attack models both for numeric and binary data, in order to insert malicious attack profiles on PPCFADD methods is proposed. Then, the robustness of the PPCFADD methods against the proposed distributed versions of the six well-known shilling attack models are analyzed. Next, the empirical studies are described, and the outcomes that prove the vulnerability of these schemes against shilling attacks, are demonstrated. Moreover, the reasons of why existing shilling attack detection methods cannot detect malicious user profiles in distributed data, and why new detection methods or the distributed versions of the existing attack detection methods need to be implemented, are discussed.

## 5.1. Introduction

Due to different shopping routines of people, rating preferences of many customers might be distributed among e-companies. Since two different e-companies might sell products from same range to identical set of customers, the type of data distribution between these e-companies is called arbitrary. In ADD, it is a challenge to produce accurate referrals for those customers, because their ratings are split. Therefore, researchers propose methods for enabling collaboration of data holders. In this scenario, privacy becomes a deterrent barrier for collaboration, accordingly, the introduced solutions include private protocols for protecting data holders' confidentiality. Although, private protocols encourage data holders to collaborate, they introduce a new drawback for partnership. Since, whole data is distributed and collaborating parties do not have full control of data, any malicious user, who knows that two parties are in collaboration, can easily insert shilling profiles to system by partitioning the fake profile between data holders. Parties may have trouble to find such profile injection attacks by employing existing detection methods, due to arbitrary distribution of data. Since profile injection attacks can easily performed on arbitrarily distributed data-based CF systems, quality, and reliability of such systems can decrease, and this may cause unhappy customers. It is obvious that, if e-companies are sure about being robust against malicious user attacks, they might hesitate to collaborate with other parties, even if, they need it. Therefore, in this chapter, aforementioned problems with arbitrarily distributed data-based CF systems are investigated, and a new research direction, *robustness of PPDCF schemes against shilling attacks*, is figured out.

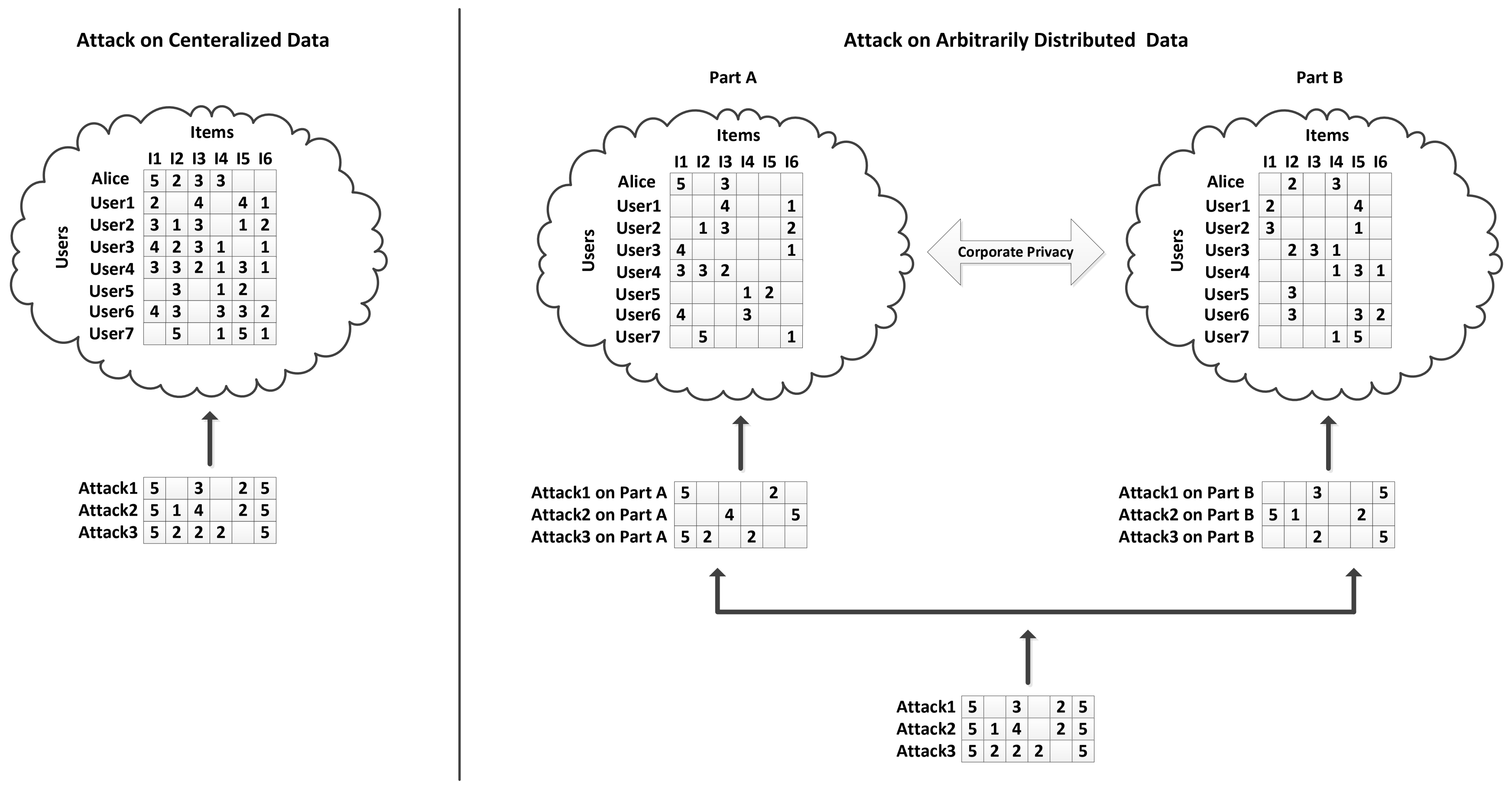
## 5.2. Designing Shilling Attacks Against Arbitrarily Distributed Data

On one hand, as explained in Chapter 2, there are many studies on profile injection attacks against CF schemes, including attack profile generation, attack strategies, and attack detection methods. On the other hand, as described in Chapter 4, there are three state-of-the-art PPDCF schemes proposed on ADD, which are presented to offer reliable predictions, while guaranteeing privacy. However, shilling attack problem is not considered in the proposed privacy-preserving schemes. Even though, data holders can overcome insufficient data problem by employing private protocols, PPDCF schemes proposed on ADD should also be evaluated with respect to robustness against shilling attacks, since, despite privacy, they might be subject to attacks. In order to analyze robustness of PPCFADD schemes, modified versions of attack generation methods are required. Therefore, the focus of this section is the design strategies of shilling attacks against PPCFADD schemes, and then, the robustness of these schemes is investigated.

In previous studies on shilling attacks, whole user preferences are owned by one data holder. Accordingly, to manipulate the results of the system, malicious users are required to insert the fake profiles into a single center. Moreover, all the proposed detection methods are developed for the case, where data is centralized and data holder has the control of all data. However, when data is arbitrarily distributed between two companies, due to privacy concerns, the collaborating companies do not have full control of distributed data. Since user preferences in ADD are not in control of a single company, attackers must decide how to insert attack profiles into PPCFADD schemes.

Possibility of inserting fake profiles into the database of one of the collaborating companies may be thought. However, in this case, since companies are producing referrals with collaboration, attacks cannot achieve the desired intend. Furthermore, data holder, which fake profiles are injected, can eliminate these profiles by employing any shilling attack detection algorithm proposed for centralized data. Thus, in order to be successful, and not to be detected, instead of inserting the malicious profiles in one part, malicious users, who know the collaboration of two parties, can insert shilling profiles to the system by partitioning the profiles between data holders. The left side of Figure 5.1 shows how to insert an attack on centralized data, and the right side shows how to insert an attack having numeric ratings into PPCFADD schemes.

In the left hand side of Figure 5.1, an example of a push attack favouring the target item Item6 is shown (Mobasher et al., 2007b). In this example, system uses a simplified user-based CF approach, where the predicted ratings for Alice on Item6 will be obtained by finding the closest neighbour to Alice. Before the attack profiles are injected to the system, the most similar user to Alice would be User6, thus the prediction for Item6 would be 2, which means Alice will probably disliked this item. However, after the attack, the most similar user to Alice will be Attack1, and the predicted rating for Item6 will be 5, which is opposite of what would be predicted



Attack on Centeralized Data
Attack on Arbitrarily Distributed Data
Part A
Part B
Items
Users
Corporate Privacy

Attack on Centeralized Data

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice | 5 | 2 | 3 | 3 |  |  |
| User1 | 2 |  | 4 |  | 4 | 1 |
| User2 | 3 | 1 | 3 |  | 1 | 2 |
| User3 | 4 | 2 | 3 | 1 |  | 1 |
| User4 | 3 | 3 | 2 | 1 | 3 | 1 |
| User5 |  | 3 |  | 1 | 2 |  |
| User6 | 4 | 3 |  | 3 | 3 | 2 |
| User7 |  | 5 |  | 1 | 5 | 1 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 | 5 |  | 3 |  | 2 | 5 |
| Attack2 | 5 | 1 | 4 |  | 2 | 5 |
| Attack3 | 5 | 2 | 2 | 2 |  | 5 |

Attack on Arbitrarily Distributed Data

Part A

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice | 5 |  | 3 |  |  |  |
| User1 |  |  | 4 |  |  | 1 |
| User2 |  | 1 | 3 |  |  | 2 |
| User3 | 4 |  |  |  |  | 1 |
| User4 | 3 | 3 | 2 |  |  |  |
| User5 |  |  |  | 1 | 2 |  |
| User6 | 4 |  |  | 3 |  |  |
| User7 |  | 5 |  |  |  | 1 |

Part B

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice |  | 2 |  | 3 |  |  |
| User1 | 2 |  |  |  | 4 |  |
| User2 | 3 |  |  |  | 1 |  |
| User3 |  | 2 | 3 | 1 |  |  |
| User4 |  |  |  | 1 | 3 | 1 |
| User5 |  | 3 |  |  |  |  |
| User6 |  | 3 |  |  | 3 | 2 |
| User7 |  |  |  | 1 | 5 |  |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 on Part A | 5 |  |  |  | 2 |  |
| Attack2 on Part A |  |  | 4 |  |  | 5 |
| Attack3 on Part A | 5 | 2 |  | 2 |  |  |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 on Part B |  |  | 3 |  |  | 5 |
| Attack2 on Part B | 5 | 1 |  |  | 2 |  |
| Attack3 on Part B |  |  | 2 |  |  | 5 |

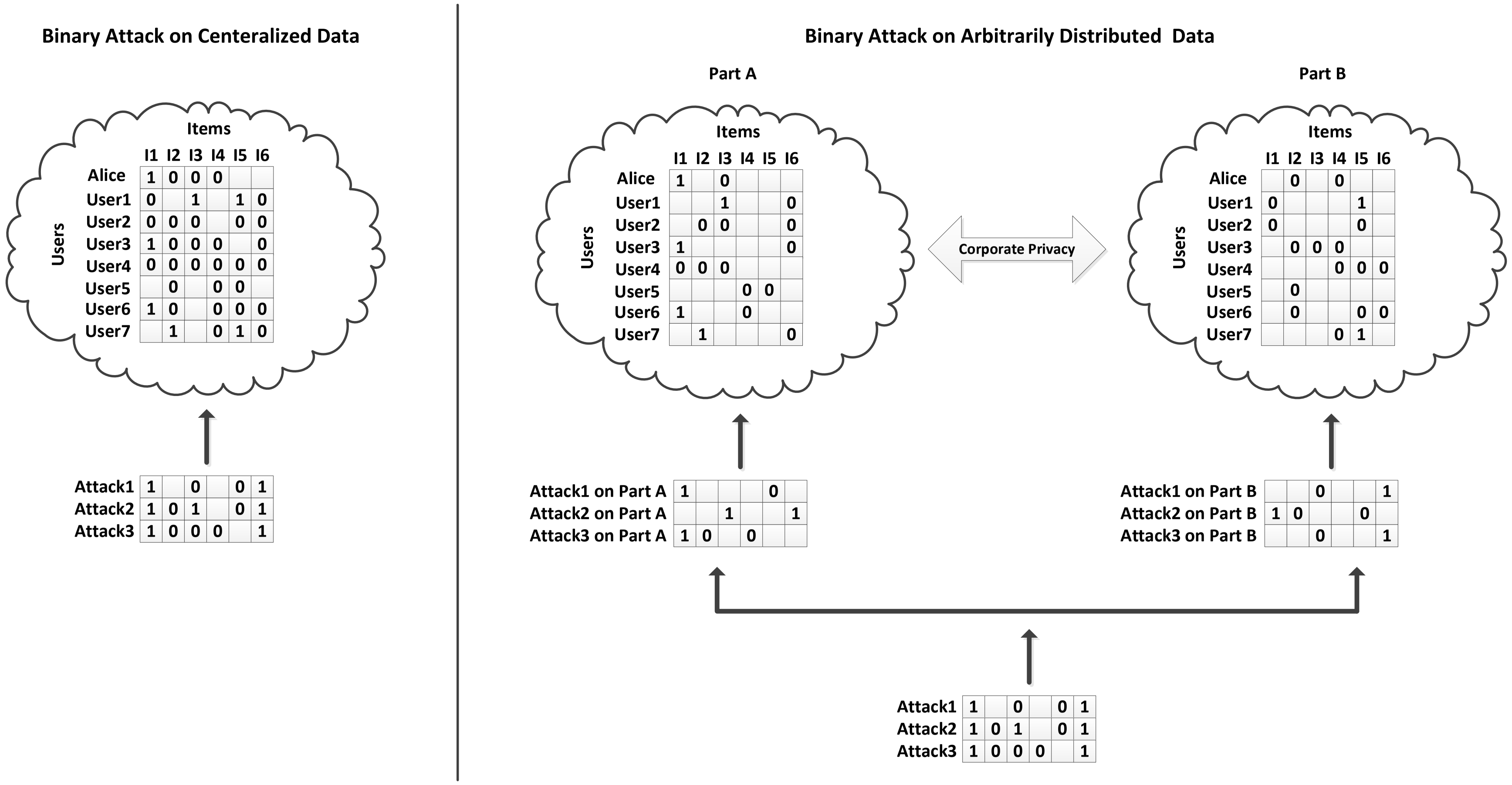
|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 | 5 |  | 3 |  | 2 | 5 |
| Attack2 | 5 | 1 | 4 |  | 2 | 5 |
| Attack3 | 5 | 2 | 2 | 2 |  | 5 |

**Figure 5.1.** Attack on Arbitrarily Distributed Numeric Data

before inserting the attack profiles. Hence, attacker reaches her goal in this example (Mobasher et al., 2007b).

The same example, but this time in distributed manner, is shown in the right hand side of Figure 5.1. In the figure, there are two e-commerce, Part A and Part B, who desire to utilize one of the PPCFADD solutions provided by Yakut and Polat (2012a,c) for numeric datasets. Any malicious user, who knows collaboration, can insert an attack by partitioning the ratings of the attack profile between these two parties. For instance, Attack1 inserts the rating vector $\{5, -, 3, -, 2, 5\}$ into the centralized data in the left hand side of Figure 5.1. In the case of ADD, attacker can distribute the attack profile between two sides, thus Attack1 is inserted as the rating vector $\{5, -, -, -, 2, -\}$ to Part A, and the rating vector $\{-, -, 3, -, -, 5\}$ to Part B, as seen in right hand side. In fact, in both cases, the attacker inserts the same attack profile to CF system, but in the in the left hand side of Figure 5.1, attack profile is inserted directly into the centralized data, and in the right hand side of Figure 5.1, it is inserted into ADD by partitioning the ratings of the attack profile between Part A and Part B. Since, Part A and Part B will provide recommendations to their customers on their integrated data, attacker will again succeed, and will be able to push the ratings of Item6. Although parties increase their recommendation quality by employing PPCFADD solution, an attacker who knows their cooperation can easily manipulate predictions. Moreover, since introduced shilling attack detection methods works on centralized data, the parties in collaboration cannot detect fake profiles. Consequently, data holders in collaboration becomes more vulnerable to shilling attacks.

Similarly, Figure 5.2 shows inserting a binary attack profile into the proposed NBC-based PPCFADD scheme (Yakut and Polat, 2012b). The numeric ratings in the example shown in Figure 5.1 are converted into binary ratings. If the given rating is bigger than 3, it is replaced with 1, otherwise it gets 0 rating. When data is centralized, attacker inserts binary rating vector $\{1, -, 0, -, 0, 1\}$ into the system


Binary Attack on Centeralized Data
Binary Attack on Arbitrarily Distributed Data
Part A
Part B
Items
Users
Corporate Privacy

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice | 1 | 0 | 0 | 0 |  |  |
| User1 | 0 |  | 1 |  | 1 | 0 |
| User2 | 0 | 0 | 0 |  | 0 | 0 |
| User3 | 1 | 0 | 0 | 0 |  | 0 |
| User4 | 0 | 0 | 0 | 0 | 0 | 0 |
| User5 |  | 0 |  | 0 | 0 |  |
| User6 | 1 | 0 |  | 0 | 0 | 0 |
| User7 |  | 1 |  | 0 | 1 | 0 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 | 1 |  | 0 |  | 0 | 1 |
| Attack2 | 1 | 0 | 1 |  | 0 | 1 |
| Attack3 | 1 | 0 | 0 | 0 |  | 1 |

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice | 1 |  | 0 |  |  |  |
| User1 |  |  | 1 |  |  | 0 |
| User2 |  | 0 | 0 |  |  | 0 |
| User3 | 1 |  |  |  |  | 0 |
| User4 | 0 | 0 | 0 |  |  |  |
| User5 |  |  |  | 0 | 0 |  |
| User6 | 1 |  |  | 0 |  |  |
| User7 |  | 1 |  |  |  | 0 |

|  | I1 | I2 | I3 | I4 | I5 | I6 |
|---|---|---|---|---|---|---|
| Alice |  | 0 |  | 0 |  |  |
| User1 | 0 |  |  |  | 1 |  |
| User2 | 0 |  |  |  | 0 |  |
| User3 |  | 0 | 0 | 0 |  |  |
| User4 |  |  |  | 0 | 0 | 0 |
| User5 |  | 0 |  |  |  |  |
| User6 |  | 0 |  |  | 0 | 0 |
| User7 |  |  |  | 0 | 1 |  |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 on Part A | 1 |  |  |  | 0 |  |
| Attack2 on Part A |  |  | 1 |  |  | 1 |
| Attack3 on Part A | 1 | 0 |  | 0 |  |  |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 on Part B |  |  | 0 |  |  | 1 |
| Attack2 on Part B | 1 | 0 |  |  | 0 |  |
| Attack3 on Part B |  |  | 0 |  |  | 1 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Attack1 | 1 |  | 0 |  | 0 | 1 |
| Attack2 | 1 | 0 | 1 |  | 0 | 1 |
| Attack3 | 1 | 0 | 0 | 0 |  | 1 |

**Figure 5.2.** Attack on Arbitrarily Distributed Binary Data

to push the predicted rating of Item6. On the other hand, when data is arbitrarily distributed, attacker can split up the ratings in the attack profile between Part A, and Part B. Thus, the attacker knowing the collaboration of these two parts can insert the binary rating vector $\{1, -, -, -, 0, -\}$ to Part A, and the binary rating vector $\{-, -, 0, -, -, 1\}$ to Part B. In spite of the private collaboration, attacker will succeed and will be able to turn this cooperation to her advantage.

By utilizing the proposed method, six well-known shilling attack models can be inserted into PPCFADD schemes. Hence, state-of-the-art privacy-preserving schemes on ADD are scrutinized in term of robustness against distributed versions of these shilling attack models. Four of these attacks are push attacks, which aim to make a target item more likely to be recommended. Distributed versions of random, average, segment, and bandwagon attack models are used as push attacks. Conversely, distributed versions of love/hate and reverse bandwagon attack models are used as nuke attacks, which aim to make a target item less likely to be recommended.

## 5.3. Experimental Evaluation

Real data-based experiments are performed to evaluate the effectiveness of six well-known shilling attack models on the three PPCFADD schemes. In the experiments, effects of two controlling parameters, filler size ($FS$) and attack size ($AS$), are analyzed. $FS$ is the percentage of empty cells chosen to be filled with fake ratings in the attacker profiles (Mehta and Nejdl, 2008; Güneş et al., 2014). For example, $FS$ value as 10% corresponds to 150 filler ratings in an attack profile for a system having 1500 rateable items. $AS$ is the number of injected attack profiles, and can be measured as a percentage of the pre-attack user count (Mobasher et al., 2007b). For a system having initially 1000 users, $AS$ value 1% means that the number of attack profiles inserted to the system is 10. Different sets of trials are conducted to show how shilling attacks effect PPCFADD solutions with varying values of the controlling parameters. Hence, 1%, 3%, 5%, 10%, 15%, and 25% values are set for

$FS$, and $AS$ on each attack model. For each of the PPCFADD schemes, privacy-preserving parameters are set according to the best results obtained by the authors (Yakut and Polat, 2012c,a,b).

### 5.3.1. Data set and evaluation criteria

In the experiments, a well-known publicly available MovieLens 100K[1] dataset is employed. The data set is collected by the GroupLens research project at the University of Minnesota, and it contains 100,000 ratings from 943 users on 1,682 movies, such that at least 20 movies are rated by each user. Within the dataset, all ratings are integer values between 1 and 5, where 1 indicates the lowest rating (disliked), and 5 indicates the highest rating (most liked).

In order to measure the effectiveness of the applied shilling attack models, *prediction shift* metric, which measures the change in the predicted rating of an item before and after the attack is applied (O'Mahony et al., 2004a; Burke et al., 2005a). *Prediction shift* is the most commonly used metric in measuring the success of an attack, but it works on only numerical ratings (Kaleli and Polat, 2013). Thus, to assess the success of shilling attacks on binary data, *ratio shift* metric proposed by Kaleli and Polat (2013) is employed. *Ratio shift* measures the ratio of 1's in prediction results before and after the attack, and can be formulated as follows:

$$RatioShift = RatioOf1s_{afterAttack} - RatioOf1s_{beforeAttack} \tag{5.1}$$

where $RatioOf1s_{beforeAttack}$, and $RatioOf1s_{afterAttack}$, represent the percentage of 1's predicted for a target item before and after the system has been attacked, respectively. For a successful attack, if $i_t$ is aimed to be pushed, then the value of ratio shift should be positive, where as, if the attacker's goal is to nuke an item, then ratio shift should be negative for that item (Kaleli and Polat, 2013).

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

### 5.3.2. Experimental methodology

During the experiments, *all-but-one* experimentation methodology is followed. According to this methodology, one of users is chosen as $a$ for each iteration, and all the remaining users treated as the training set. For push and nuke attack model generation, two different sets of 50 target movies is created (Bilge et al., 2014a). These movies is chosen randomly within different ranges of ratings to represent the characteristics of the original dataset. Moreover, since trying to push the prediction of a popular item or to nuke the prediction of an unpopular item is unreasonable, Bilge et al. (2014a) formed these push and nuke attack sets so as to contain movies having average rating within the range of 1 to 3, and 3 to 5, respectively. Statistics of the selected target movies for push and nuke attacks are presented in Table 5.1.

**Table 5.1.** Statistics of Target Movies *(Bilge et al., 2014a)

| Total rating count | Number of pushed movies | | Number of nuked movies | |
|---|---|---|---|---|
| | With average rating 1-2 | With average rating 2-3 | With average rating 3-4 | With average rating 4-5 |
| 1 - 50 | 30 | 15 | 12 | 18 |
| 51 - 150 | - | 3 | 5 | 6 |
| 151 - 250 | - | 1 | 2 | 3 |
| >250 | - | 1 | 1 | 3 |

To asses effects of binary shilling attacks on ADD, numerical ratings firstly is transformed into binary format. If the rating of a movie is bigger than 3, in other words 4 or 5, they are labeled as one (*like*), otherwise they are assign to zero rating value (*disliked*) (Kaleli and Polat, 2013). To constitute push and nuke item sets to be attacked in binary data, the methodology conducted by Kaleli and Polat (2013) is followed. Initially, ratio of 1's and 0's for each movie are analyzed. Then, two distinct sets of 50 target movies are randomly selected for binary push and nuke attacks. Additionally, target movies in binary push attack sets are selected from the ones having zero ratings more than ones; on the contrary, target movies in binary nuke attack sets are chosen from the movies having mostly ones.

Throughout the experiments, all target movies are attacked individually for all test users. Predictions are estimated before and after inserting the attack profiles.

Thereafter, the prediction shift values are calculated to examine the relative change in predicted values for each different attack models. In binary attack experiments, for all items in both target item sets, predictions are produced for all users who do not have a rating for those items. 1's ratio values for each of the target items are computed. All target items are attacked individually for all users in the system. Ratio shift values for each item are estimated, and overall averages are presented.

### 5.3.3. Empirical results

Two groups of experiments are performed to investigate the effects of shilling attacks on PPCFADD schemes. One group of experiments are conducted to examine the effects of shilling attacks on numerical data-based solutions (Yakut and Polat, 2012a,c), and the other group of experiments are conducted to investigate the effects of binary shilling attacks (Yakut and Polat, 2012b). In the next subsections, the empirical results obtained for both push and nuke attacks with respect to varying controlling parameters are presented.

#### 5.3.3.1 *Effects of shilling attacks on arbitrarily distributed numeric data*

In order to demonstrate effects of push and nuke attack models on numeric PPCFADD schemes, firstly, experiments are performed with varying $FS$ values. This is the number of ratings for the filler items added to fill out the attack profile, and thus is directly related to the power of the attack (Mobasher et al., 2007b). During the $FS$ experiments, $AS$ is set to 15%, and $FS$ is varied from 1% to 25%. Prediction shift values across all push and nuke attack models, which obtained for the numeric PPCFADD schemes (Yakut and Polat, 2012a,c) are demonstrated in Figure 5.3, and Figure 5.4, respectively.

As shown in Figure 5.3, and Figure 5.4, the most effective attacks against PPCFADD schemes are segment and bandwagon attacks. The bandwagon attack achieved a maximum prediction shift of 1.23 and 1.49 on the results of proposed

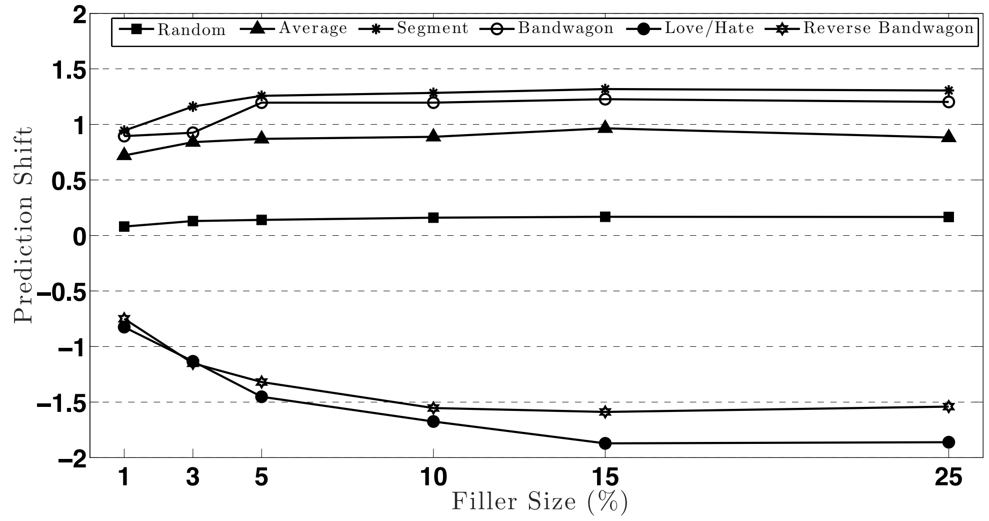


**Figure 5.3.** Prediction Shift with Varying Filler Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012a)

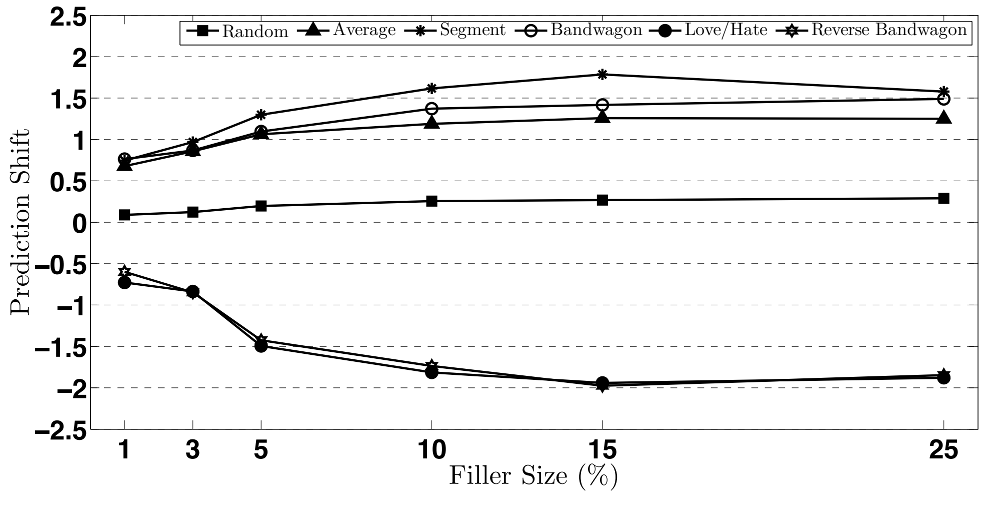


**Figure 5.4.** Prediction Shift with Varying Filler Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012c)

privacy preserving schemes on ADD and CDD, respectively. The segment attack is somewhat more successful, and achieved a maximum prediction shift of 1.32 and 1.79 on the results of privacy-preserving scheme for ADD and CDD, respectively, when $FS$ and $AS$ are set to 15%. This prediction shifts are considered to be significant on a five-star scale. Though not as successful as segment and bandwagon attacks, average attack achieved a prediction shift of 0.97 on ADD, and 1.26 on CDD. On the other hand, random attack could not be effective, and obtained approximately 0.2 prediction shift on both of the proposed privacy-preserving schemes. When the effects of nuke attacks on PPCFADD schemes are compared, it is seen that the love/hate attack is quite effective, and can reduce the predicted rating of a highly rated movie 1.87 points on ADD, and 1.94 points on CDD with 15% filler size. The reverse bandwagon attack also performed effectively, and reached a maximum nuke prediction shift of 1.59, and 1.97 on the results of proposed privacy-preserving schemes on ADD and CDD, respectively. If the obtained results are compared by considering the two schemes, it might be stated that both push and nuke attacks are more effective on the proposed scheme on CDD than the solution for ADD. Since ADD solution employs item-based algorithm, it is reasonable that it is more robust than CDD-based scheme. On both of the schemes, effects of the attacks increases up to the optimum value of $FS$ is reached. After the optimum value, since change of being in $a$'s neighbourhood for an attack profile decreases, increasing in $FS$ decreases the effectiveness of the attacks for all push and nuke attack models. Also, once the optimum value is exceeded, the balance between coverage and generality is broken, and profile becomes dissimilar to any given user (Mobasher et al., 2007b).

Secondly, another set of experiments are conduced with varying $AS$ to examine the effects of the number of injected attack profiles on prediction shift. During $AS$ experiments, $AS$ is varied from 1% to 25%, and $FS$ is fixed at 15%. The results of push and nuke attacks for various $AS$ values on ADD and CDD are presented severally in Figure 5.5, and Figure 5.6. Similar to the previous experiments, the

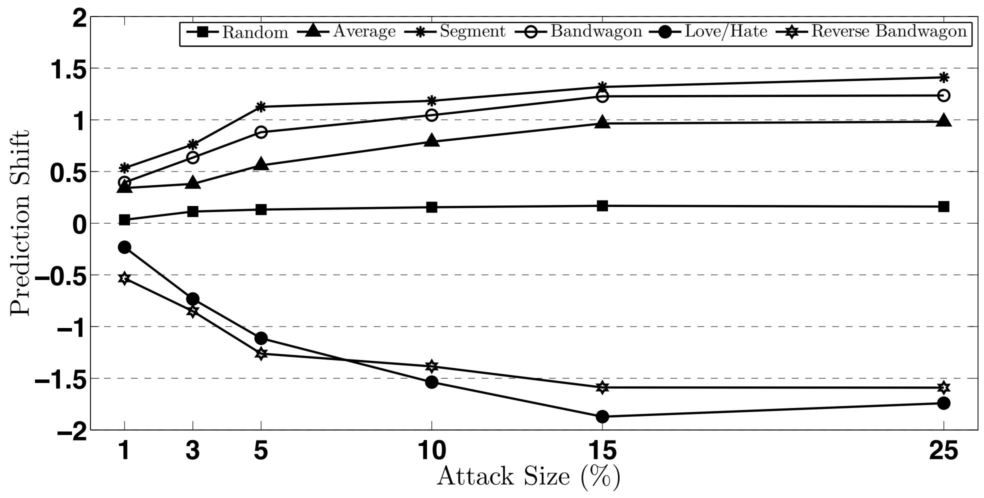


**Figure 5.5.** Prediction Shift with Varying Attack Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012a)

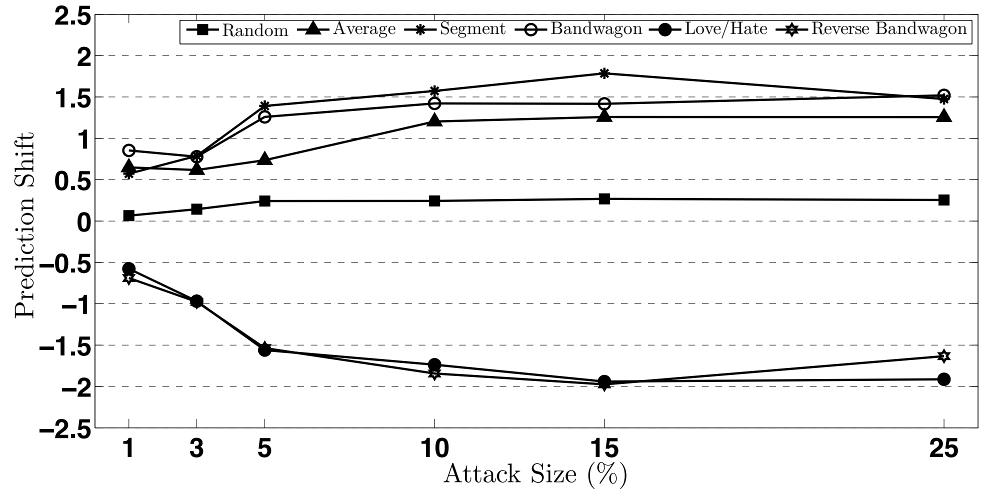


**Figure 5.6.** Prediction Shift with Varying Attack Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012c)

segment and the bandwagon attacks are the most successful push attacks against both schemes. The segment attack achieved a maximum prediction shift of 1.41, and 1.78, and the bandwagon attack achieved a maximum prediction shift of 1.24, and 1.52 on ADD and CDD, respectively. As in the $FS$ experiments, random attack could not achieve significant prediction shift values for varying attack sizes in both schemes. However, unlike random attack, average attack achieved a prediction shift of 0.98 on ADD, and 1.26 on CDD. When considering the effects of nuke attacks with varying $AS$ values, it can be concluded that both reverse bandwagon and love/hate attacks are very successful in manipulating the predictions, especially for $FS$ being 15%. The love/hate attack scaled down the predictions by 1.87 points on ADD, and 1.94 points on CDD. Likewise, the reverse bandwagon attack reduce the predictions 1.59 and 1.97 points on ADD and CDD, respectively. These negative shifts might be considered highly significant in a five-star rating scale, since these values are enough to drop an average rated movie to the end of the scale. Looking at the impact of push and nuke attacks on both schemes, attacks appear to be more effective on CDD than ADD, as in the previous experiment. Furthermore, as $AS$ increases the prediction shift also rises for all push attack models, until the attack size reached the optimal size. After this point, increasing $AS$ drops off the prediction shift on both schemes. This is also same for nuke attacks. Hence for nuke attack models, lower prediction shift values are achieved with greater values of attack size.

These experimental findings indicated that malicious users can manipulate the results of PPCFADD schemes for numeric datasets. Except random attack model, all push attacks significantly effect recommendation results. The most successful push attack models on both schemes are segment, bandwagon, and average attacks, respectively. On the other hand, both love/hate, and reverse bandwagon attack models achieved considerable negative prediction shift values, thus they are very effective on the results of proposed privacy-preserving schemes on ADD and CDD.

#### 5.3.3.2 *Effects of shilling attacks on arbitrarily distributed binary data*

To investigate the effects of binary push and nuke attack models on the privacy preserving scheme proposed to estimates NBC-based predictions on ADD (Yakut and Polat, 2012b), last set of experiments are performed. In order to examine ratio shift values across all binary push and nuke attack models with respect to $FS$, $AS$ is fixed at 15% while $FS$ is varied from 1% to 25%. The acquired ratio shift values are presented in Figure 5.7.

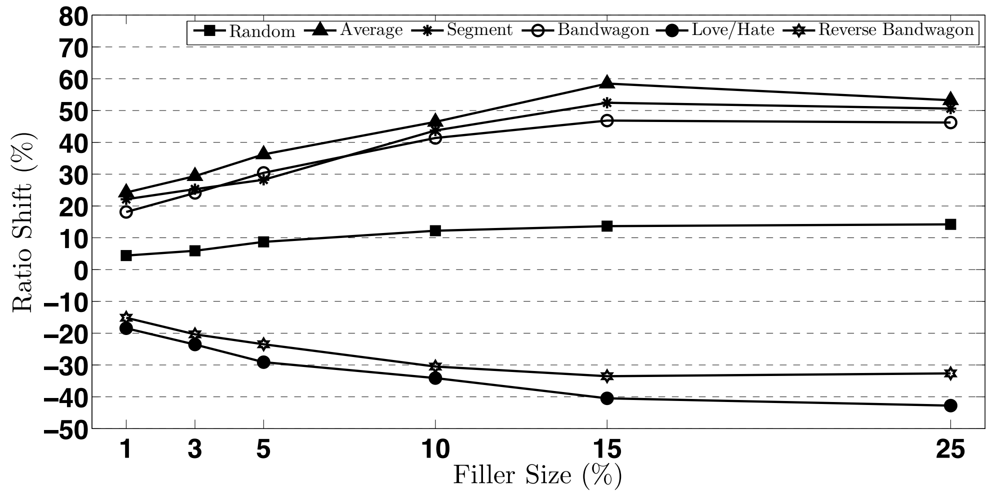


**Figure 5.7.** Ratio Shift Values for Varying Filler Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012b)

As seen in Figure 5.7, average and segment binary push attacks are significantly effective to manipulate recommendation system. Binary forms of such push attacks accomplished 58.50% and 52.45% ratio shift values when both $FS$ and $AS$ values are set to 15%. These values indicates that when an attacker aims to push a target movie having 1's ratio of 30%, she can increase 1's ratio to 88.50% and 82.45% with employing binary average attack and binary segment attack, respectively. Hence, with both attacks she might successfully push the predicted values of the target movie. The binary bandwagon attack are slightly less successful, however still very effective by achieving a maximum ratio shift of 46.85% on the results. On the other hand, the impact of binary random attacks are comparably smaller than the other push attack models. Binary random attacks achieved a maximum ratio shift of 14.22%, hence, in case of 5% or less attack sizes, random attack is even not successful at all. If effects of binary nuke attacks are compared, it is seen that binary love/hate attack is the most successful attack type for nuking a movie. Love/hate attack reached a maximal negative ratio shift of 42.76%, which connotes that 1's ratio of a target movie will be decreased to 47.24% from 90% when nuked. Besides, binary reverse bandwagon attacks can also be employed for nuking. However, their success ratio is lightly smaller than love/hate attacks. If binary reverse bandwagon attack is chosen to nuke a movie, negative ratio shift of 33.52% can be achieved. Similar to the effects of shilling attacks on arbitrarily distributed numeric datasets, as $FS$ increases success of both binary push and nuke attacks also increases. For smaller $FS$ values, improvements in ratio shift are noticeable. However, after reaching optimum value, increase in $FS$ decreases the effects of the attacks.

Secondly, effects of binary push and nuke attacks are tested with varying attack sizes on the NBC-based private scheme proposed for ADD. By keeping $FS$ as constant at 15%, and varying $AS$ from 1% to 25%, ratio shift values are obtained against whole binary push and nuke attack models. Figure 5.8 shows the results of these experiments. Similar to the $FS$ experiments on binary data, except random attack, push attacks can significantly manipulate the popularity of a movie. While average attack achieved a maximum ratio shift of 58.50%, segment attack reached 55.90%, and bandwagon attack obtained 47.83% maximal ratio shift values. Referring to these obtained ratio shift values, it can be concluded that an attacker might successfully push the predicted rating of a movie with both attacks. Binary random attack achieved a maximum ratio shift of 13.67%, which is not enough to change the predicted value of a movie. Furthermore, for small $AS$ values, binary random attack exactly fails. When effects of binary nuke attacks on NBC is analyzed, it is clearly seen that both love/hate and reverse bandwagon attacks are pretty successful. Binary love/hate attack can reduce the 1's ratio of a target movie by 40.48%,

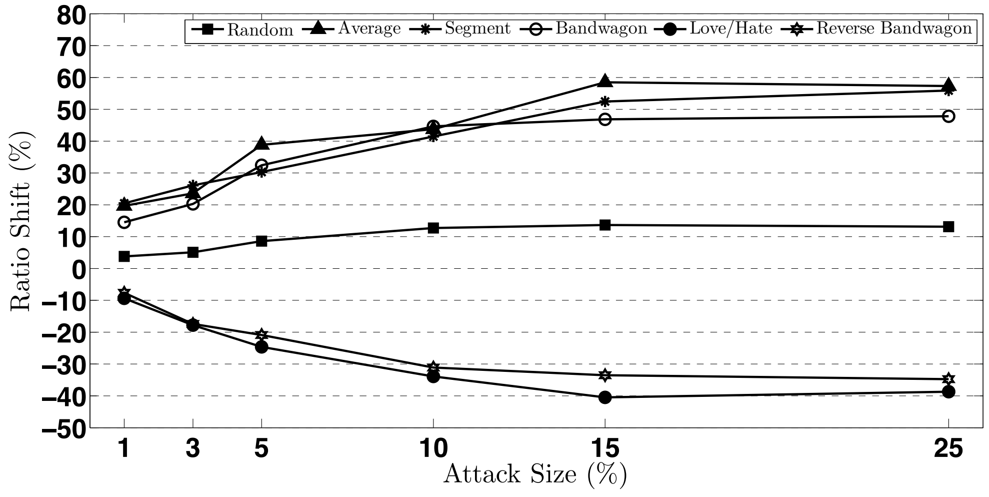


**Figure 5.8.** Ratio Shift Values for Varying Attack Size on Privacy Preserving Scheme Proposed by *Yakut and Polat (2012b)

otherwise if binary reverse bandwagon attack is applied to nuke a movie, negative ratio shift of 34.76% can be succeeded. For both binary push and nuke attacks, increasing the *AS* increases the impact of attacks. However, this condition remains until the *AS* reached its optimal value. Thereafter, enhancing the *AS* reduces the influence of binary attacks.

These experimental results indicates that malicious users can change the outcomes of NBC-based PPCFADD scheme. Similar to the results of the experiments on numeric datasets, all push attack models significantly effect the results, but random attack model. However, the success order of the push attack models on binary data are different from numeric data. While the most successful attack model on numeric data is segment attack, on binary data the most successful attack model is average attack. This is followed by segment, and bandwagon attacks, respectively. As on numeric data, both love/hate, and reverse bandwagon attacks achieve considerable negative ratio shift values on the results of NBC-based privacy-preserving scheme.

### 5.3.4. Discussion

In this section, robustness of three studies enabling data holders private collaboration on ADD are analyzed against well-known six shilling attack models. According to the experimental findings, it can be claimed that PPCFADD schemes for numeric and binary data can be subjected to shilling attacks. Although these schemes allow two parties to collaborate to offer more useful CF services without violating confidentiality, malicious users who knows the collaboration can insert shilling profiles to disrupt the results. There are several techniques proposed to detect attacks in recommender systems having numeric data. However, these existing methods work on centralized data. When data is arbitrarily distributed between two parties, each party can operate on only its own data. Therefore, by employing the existing detection methods on ADD, they cannot find the distributed attack profiles.

Question that should be explained is *"why existing attack detection methods cannot reveal shilling profiles on PPCFADD schemes"*. For instance, to apply the classification-based detection approaches, classification attributes should be generated based on the entire dataset. The underlying assumption behind the generic attributes proposed by Mobasher et al. (2007b) is that overall statistical signature of an attack profile will diverge significantly from a genuine profile. However, when data is distributed, both data of genuine profile, and the attack profile will also be distributed between the parts. Thus, parties cannot discover the statistical difference between genuine and attack profiles while working on their private data. This is same for the model-specific attributes, which are designed to detect the attack profiles by comparing the similarity of a profile with known attack models (Mobasher et al., 2007b). Since, some ratings of an attack profile are on one side, and the rest of the ratings are on the other side, parts individually cannot compute this similarity by having part of the attack profile at hand. Hence, current feature-based algorithms, which aim to select users with the maximum impact in attributes used, cannot be applied when data are distributed.

The defined challenge for classification-based attack detection methods is also case for other attack detection methods. The intuition behind time series-based attack detection approach proposed by Zhang et al. (2006a) is that each attack model may affect the rating distribution of target or possibly other items over time. Thus, to detect suspicious behaviour, changes in *sample average* and *sample entropy* values are analyzed within a period of time. Hence, to employ this method on ADD, *sample average* property which captures the change in an item's popularity, and *sample entropy* which captures the distributional change in an item's ratings need to be computed (Zhang et al., 2006a). However, since data is arbitrarily distributed between two parties, the ratings of items are also split between them. With partial data, parties cannot construct the time series of ratings for items. Therefore, they need to collaborate to figure out the attack profiles.

Researchers also applied statistical anomaly detection method as an alternative approach in attack detection, which relies on item average values. One such technique proposed by Bhaumik et al. (2006) is based on statistical control process, which used to monitor items as they are under attack, by observing the change in item's mean rating over time. However, in the case of ADD, the half of the item ratings will be in the other part, hence monitoring the item mean rating looking at the partial data will not be truthful.

Clustering based attack detection algorithms attempt to distinguish clusters of malicious users from clusters of real users. The method proposed by Mehta (2007) bases on the intuition that clusters containing profiles of malicious users will be tighter, in the sense that these profiles will be very similar to each other. However, when data is arbitrarily distributed, attacker will insert the attack profile by distributing the ratings in the profile between two parts. Thus, the similarity among the attack profiles can vary, and thereby, it may not be possible to distinguish the clusters of attack profiles from the clusters of authentic users.

Consequently, according to experimental results, introduced PPCFADD methods are not robust for shilling attacks. Although researchers propose solutions for enhancing robustness of non-private and private centralized data-based CF algorithms against profile injection attacks, these solutions cannot be directly applied to

PPCFADD schemes. Also, there is no proposed attack detection methods for binary shilling profiles. If robustness of the privacy-preserving distributed CF algorithms not guaranteed, e-companies having insufficient data may hesitate to collaborate with other data owners. Therefore, it is essential to discover the attack profiles, and to do this new detection methods, which can work on distributed data, or the distributed versions of the existing detection methods should be implemented.

## 5.4. Conclusions

In this chapter, robustness of privacy-preserving collaborative filtering schemes against shilling attacks is investigated. Shilling attack problem on arbitrarily distributed data-based collaborative filtering systems are explored. In order to insert shilling profiles into arbitrarily distributed numeric and binary data, a method, which can be applied in generation of the well-known shilling attack models on distributed data, is proposed. Three state-of-the-art privacy-preserving schemes proposed for arbitrarily distributed data are analyzed exposed to profile injection attacks, which are injected by the proposed attack strategy. Real data-based experiments are performed to extensively evaluate the robustness of these schemes against six well-known shilling attack models. Empirical results show that privacy-preserving schemes proposed for arbitrarily distributed data are defenceless against shilling attacks, despite privacy protection. The reasons of why existing detection methods proposed in the literature against malicious users cannot be directly applied on are discussed. Need for more intelligent and expert attack detection methods for arbitrarily distribute data is emphasised.

# 6. DETECTING SHILLING PROFILES PRIVATELY ON ARBITRARILY DISTRIBUTED DATA

In this chapter, distributed version of the classification-based shilling attack detection method described in details in Chapter 2 is developed to defend privacy-preserving collaborative filtering schemes proposed for arbitrarily distributed data against shilling attacks. To derive the required classification attributes collaboratively between two data holders, private protocols, which utilize homomorphic encryption and random filling methods, are proposed for secure computations. Moreover, the need for collaboration in attack detection on arbitrarily distributed data is experimentally shown up.

## 6.1. Introduction

According to the studies in literature, insufficient data problem might be accomplished by employing private protocols, which allow cooperation of data holders while preserving privacy. On the other hand, there also exist centralized solutions for shilling attack problem, and by utilizing the proposed methods it is possible to detect shilling attacks. However, as shown empirically in Chapter 5, shilling attack problem is also valid for PPDCF solutions (Yılmazel and Kaleli, 2016). PPDCF solutions proposed on ADD are defenceless against shilling attacks, hence malicious users, who know the collaboration, might insert distributed attack profiles into ADD to disrupt the results of these systems. Looking at the results, e-companies who are unsure of being subject to shilling attacks might refrain from cooperation even they need it to offer more useful CF services to their customers. Therefore, it is essential to detect the shilling profiles in PPDCF systems.

Even though there are many solutions proposed for enhancing robustness of CF algorithms against profile injection attacks, existing methods work on centralized data, and these solutions cannot be directly applied to PPCF algorithms proposed on arbitrarily distribute data (PPCFADD) (Yılmazel and Kaleli, 2016). When data is arbitrarily distributed between two parties, each party can operate on only its own data. However, existing attack detection methods utilize information about user-item matrix and a data owner controls whole attack detection process. Therefore, by employing the existing detection methods on ADD, they cannot find the distributed attack profiles. As a result, more intelligent and expert attack detection methods for ADD are required.

In nutshell, this chapter is focused on detecting shilling profiles in PPCFADD systems. Distributed version of a well-known classification-based attack detection method is proposed to defend PPCFADD schemes against shilling attacks that are injected by the attack strategy introduced in Chapter 5. Besides, it is empirically shown that even if parts have their own defending mechanisms based on their own data, distributed attack profiles on ADD might not be detected without collaboration.

## 6.2. Private Protocols for Finding Classification Attributes Required for Shilling Attack Detection on Arbitrarily Distributed Data

In this section, steps in deriving the required classification attributes privately between two data holders are described. First, the *data masking* that should be applied by the data holders to mask their own private data before the collaborative works is specified. Then, the *Revised Private Mean Estimation Protocol* that is designed to compute some essential primitive values, which are compulsory in calculation of the classification attributes, especially the generic ones, is defined. Finally, the *private protocols*, which are built up to derive the required generic and model-specific classification attributes for each distributed profile collaboratively between two data holders, are represented.

### 6.2.1. Data masking using random filling method

Before collaboration, collaborating parts, namely, Part A and Part B, need to mask their own data, in order to protect the confidentiality of their data sets. For data disguising, parts can use the method proposed by Yakut and Polat (2012a) for ADD, whose steps are described as follows:

i. As an initial step, each part determines the density of users' in their own databases, which is the number of rated items in a user profile.

ii. Next, each part uniformly randomly selects $\theta$ percent of their empty cells, where $\theta$ is determined as stated by Yakut and Polat (2012a).

iii. Then, parts fill such cells with fake ratings, and obtain their masked databases. Fake ratings can be determined by following one of the strategies specified by Yakut and Polat (2012a).

### 6.2.2. Revised private mean estimation protocol

After masking private databases by using fake ratings, parts need to calculate the classification attributes necessary for shilling attack detection by collaborating with each other. 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 mean rating of a user, and an item, or the total number of ratings in the system provided by a user, or provided for an item, need to be known. Let's first explain how necessary building blocks can be estimated without violating data owners' privacy.

Since data is arbitrarily distributed between Part A and B, parties need to collaborate with each other to estimate the user mean ratings and the item mean ratings, without deeply jeopardizing their privacy. In general, arithmetic mean, or simply mean or average, can be computed as the sum of all the numbers in the series divided by the count of all numbers in that series. Hence, $mean = sum/count$ is an example of algebraic measure that can be calculated by applying division function to distributive measures *sum* and *count*. More specifically, *sum* and *count* can be computed by partitioning the data into smaller sets, computing each measure for each subset, and finally merging them to obtain the final value (Yakut and Polat, 2012a). Let's look at the calculations of user mean ratings and item mean ratings distributively on ADD.

The mean of ratings provided for *item i* by all users, which is represented as $\overline{r_i}$, can be computed in distributive manner between Part A and B as follows:

$$\overline{r_i} = \frac{\sum_{u \epsilon U_i} r_{u,i}}{R_{U,i}} = \frac{\sum_{u \epsilon U_{i_{PartA}}} r_{u,i} + \sum_{u \epsilon U_{i_{PartB}}} r_{u,i}}{R_{U,i_{PartA}} + R_{U,i_{PartB}}} = \frac{sum_{PartA} + sum_{PartB}}{count_{PartA} + count_{PartB}} \tag{6.1}$$

where, $U_i$ is the set of users who have given rating for *item i*. $r_{u,i}$ is the rating given by *user u* to *item i*, where $u \epsilon U_i$, and $U_{i_{PartA}} \cup U_{i_{PartB}} = U_i$. $R_{U,i}$ is the total number of ratings in the system provided for *item i* by all users, which is especially the size of the set $U_i$. The total number of ratings held by Part A and B is shown by $R_{U,i_{PartA}}$ and $R_{U,i_{PartB}}$, respectively for *item i*.

Similarly, the mean of ratings provided by *user u* across all items, which is represented as $\overline{v_u}$, can be computed distributively between Part A and B as follows:

$$\overline{v_u} = \frac{\sum_{j \epsilon I_u} v_{u,j}}{N_u} = \frac{\sum_{j \epsilon I_{u_{PartA}}} v_{u,j} + \sum_{j \epsilon I_{u_{PartB}}} v_{u,j}}{N_{u_{PartA}} + N_{u_{PartB}}} = \frac{sum_{PartA} + sum_{PartB}}{count_{PartA} + count_{PartB}} \tag{6.2}$$

where, $I_u$ is the set of items that *user u* provided a rating. $v_{u,j}$ is the rating given by *user u* to *item j*, where $j \epsilon I_u$, and $I_{u_{PartA}} \cup I_{u_{PartB}} = I_u$. $N_u$ is the total number of ratings in the system provided by *user u*, which is especially the size of the set $I_u$, whereas $N_{u_{PartA}}$ and $N_{u_{PartB}}$ show the total number of ratings held by Part A and B, respectively for *user u*.

As can be seen from Eq. 6.1 and 6.2, calculation of both user and item mean ratings can be obtained in similar manners only by changing the input. While calculating the user mean rating, the user profile or the row of the user-item matrix need to be considered, on the other hand, while calculating the item mean rating, the ratings given by all the users to a specific item or the column of the user-item matrix is needed. Since calculation of both user and item mean ratings can be obtained similarly by changing the input vector, we revised Yakut and Polat (2012a)'s *Private Mean Estimation Protocol*, which is designed for estimating the user average ratings on ADD, to calculate both user and item mean ratings. Moreover, with some additional steps, it becomes possible to store some intermediate count values, which are necessary in calculation of the classification attributes.

Hence, the parties can estimate $\overline{r_i}$ values for all items, $i = 1, 2, ..., n$ (or $\overline{v_u}$ values for all users, $u = 1, 2, .., m$), in a distributive manner, as follows:

i. Part A and B compute partial *sum* and *count* values based on their own databases for all items (or users). Hence, while Part A computes $\sum_{u \epsilon U_{i_{PartA}}} r_{u,i}$ (or $\sum_{j \epsilon I_{u_{PartA}}} v_{u,j}$) and $R_{U,i_{PartA}}$ (or $N_{u_{PartA}}$), Part B computes $\sum_{u \epsilon U_{i_{PartB}}} r_{u,i}$ (or $\sum_{j \epsilon I_{u_{PartB}}} v_{u,j}$), and $R_{U,i_{PartB}}$ (or $N_{u_{PartB}}$).

ii. Then, Part A sends estimated sub-aggregates, which are *sum* and *count*, calculated for the items (or users) with odd indices to Part B, and Part B sends estimated sub-aggregates calculated for the items (or users) with even indices to Part A. By this way, parts exchange sub-aggregates for half of the items (or users).

iii. Next, by using the sub-aggregates that they own, and the other part sends, Part A and B estimate item (or user) mean ratings for even and odd indexed items (or users), respectively.

   For this calculations, Part A will perform the following steps:

   - Part A finds sum and count values of even indexed items (or users), which are $ItemSum_{even}$ (or $UserSum_{even}$), and $R_{U,i_{even}}$ (or $N_{u_{even}}$) in the rest of the equations.

- Part A stores $R_{U,i_{even}}$ (or $N_{u_{even}}$) value that will be used in the rest of the protocols.
- Part A estimates the mean ratings' of items (or users) with even indices, which is $\overline{r_{i_{even}}}$ (or $\overline{v_{u_{even}}}$).

Similarly, Part B will perform the following steps:

- Part B finds sum and count values of odd indexed items (or users), which are $ItemSum_{odd}$ (or $UserSum_{odd}$), and $R_{U,i_{odd}}$ (or $N_{u_{odd}}$) in rest of the equations.
- Part B stores $R_{U,i_{odd}}$ (or $N_{u_{odd}}$) value that will be used in the rest of the protocols.
- Part B estimates the mean ratings' of items (or users) with odd indices, which is $\overline{r_{i_{odd}}}$ (or $\overline{v_{u_{odd}}}$).

iv. Finally, parts exchange the estimated item (or user) mean ratings. Thus, at the end of this protocol, each part ends with the $\overline{r_i}$ (or $\overline{v_u}$) values for all items (or users).

Distributed computations require data sharing between the parties, thus the important issue is exchanging as few as possible amounts of data. With this protocol each part sends data to the other part for half of the items (or users). Therefore, the companies cannot figure out the sum of the ratings and the values given to such items (or users) during the protocol (Yakut and Polat, 2012a).

Parties need to apply *Revised Private Mean Estimation Protocol* twice, ones for estimating the item mean ratings, and ones for the user mean ratings. At the end, each part will have the following information about the data: Part A and B both learns $\overline{r_i}$ values for all items, and $\overline{v_u}$ values for all users. While Part A has a knowledge about $R_{U,i_{even}}$, and $N_{u_{even}}$, B knows $R_{U,i_{odd}}$, and $N_{u_{odd}}$. These values will be used in the calculation of the classification attributes necessary for shilling attack detection.

### 6.2.3. Private estimation of RDMA, WDMA, and WDA attributes

Generic attributes RDMA, WDA, and WDMA, whose equations are given in Eq. 2.1, 2.2, 2.3, can be calculated in distributed manner between Part A and B as shown in Eq. 6.4, 6.3, 6.5, respectively.

$$\begin{aligned} WDA_u = \sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r_i}|}{R_{U,i}} &= \sum_{i=0}^{N_{u_{PartA}}} \frac{|r_{u,i} - \bar{r_i}|}{R_{U,i}} + \sum_{i=0}^{N_{u_{PartB}}} \frac{|r_{u,i} - \bar{r_i}|}{R_{U,i}} \\ &= WDA_{u_{PartA}} + WDA_{u_{PartB}} \end{aligned} \tag{6.3}$$

$$\begin{aligned} RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i}}}{N_u} &= \frac{\sum_{i=0}^{N_{u_{PartA}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i}} + \sum_{i=0}^{N_{u_{PartB}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i}}}{N_u} \\ &= \frac{WDA_{u_{PartA}} + WDA_{u_{PartB}}}{N_u} \end{aligned} \tag{6.4}$$

$$\begin{aligned} WDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i}}^2}}{N_u} &= \frac{\sum_{i=0}^{N_{u_{PartA}}} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i}}^2} + \sum_{i=0}^{N_{u_{PartB}}} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i}}^2}}{N_u} \\ &= \frac{WDMA_{u_{PartA}} + WDMA_{u_{PartB}}}{N_u} \end{aligned} \tag{6.5}$$

To compute these three attributes, first of all each part should find $|r_{u,i} - \bar{r_i}|$. Since each of them knows the item means, $\bar{r_i}$, and the items the user rated on their sides, along with the given rating value, they can compute this absolute difference by themselves. According to the formulas of RDMA, WDA, and WDMA; this calculated absolute difference value, $|r_{u,i} - \bar{r_i}|$, should be divided by $R_{U,i}$, or ${R_{U,i}}^2$. However, Part A only knows $R_{U,i_{even}}$, in other words, the counts of even indexed items. Hence, for each user Part A can find $\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i}}$, and $\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i}}^2}$ value for even indexed items by herself. However, for the odd indexed items she should collaborate with Part B. This is same for Part B. Since, Part B only knows $R_{U,i_{odd}}$, the counts of odd indexed items, for each user Part B can find $\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i}}$, and $\sum_{i=0}^{N_u} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i}}^2}$ value for odd indexed items by herself. Whereas, for the even indexed

$$\begin{aligned} WDA_u &= \sum_{i=0}^{N_{u_{PartA}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i}} + \sum_{i=0}^{N_{u_{PartB}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i}} \\ &= \sum_{i=0}^{N_{u_{PartA_{even}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i_{even}}} + \sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i_{odd}}} + \sum_{i=0}^{N_{u_{PartB_{even}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i_{even}}} + \sum_{i=0}^{N_{u_{PartB_{odd}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{R_{U,i_{odd}}} \\ &= WDA_{u_{PartA_{even}}} + WDA_{u_{PartA_{odd}}} + WDA_{u_{PartB_{even}}} + WDA_{u_{PartB_{odd}}} \end{aligned} \tag{6.6}$$

$$\begin{aligned} WDMA_u &= \frac{\sum_{i=0}^{N_{u_{PartA}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i}}^2} + \sum_{i=0}^{N_{u_{PartB}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i}}^2}}{N_u} \\ &= \frac{\sum_{i=0}^{N_{u_{PartA_{even}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i_{even}}}^2} + \sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i_{odd}}}^2} + \sum_{i=0}^{N_{u_{PartB_{even}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i_{even}}}^2} + \sum_{i=0}^{N_{u_{PartB_{odd}}}} \frac{\left|r_{u,i}-\bar{r_i}\right|}{{R_{U,i_{odd}}}^2}}{N_u} \\ &= \frac{WDMA_{u_{PartA_{even}}} + WDMA_{u_{PartA_{odd}}} + WDMA_{u_{PartB_{even}}} + WDMA_{u_{PartB_{odd}}}}{N_u} \end{aligned} \tag{6.7}$$

items she should collaborate with Part A. Let's rewrite Eq. 6.3 and Eq. 6.5 according to $R_{U,i_{even}}$, and $R_{U,i_{odd}}$, as shown in Eq. 6.6 and Eq. 6.7, respectively.

Looking at the equation of WDMA attribute, it can be stated that the computation of numerator of the WDMA attribute is similar to the computation of WDA attribute, only difference is that there is a square in $R_{U,i}$. Thus, WDA and WDMA attributes can be calculated by applying similar steps.

i. Each part find $|r_{u,i} - \bar{r_i}|$ values by themselves.

ii. Then, each part find the required partial sum values according to the data they have.

   - Part A knows the counts of even indexed items, $R_{U,i_{even}}$, hence for each user Part A can find the following values by herself:
     - For WDA calculation: $\sum_{i=0}^{N_{u_{PartA_{even}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{even}}} = WDA_{u_{PartA_{even}}}$.
     - For WDMA calculation: $\sum_{i=0}^{N_{u_{PartA_{even}}}} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i_{even}}}^2} = WDMA_{u_{PartA_{even}}}$.
   - Similarly, Part B knows the counts of odd indexed items, $R_{U,i_{odd}}$, hence for each user Part B can find the following values by herself:
     - For WDA calculation: $\sum_{i=0}^{N_{u_{PartB_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}} = WDA_{u_{PartB_{odd}}}$.
     - For WDMA calculation: $\sum_{i=0}^{N_{u_{PartB_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{{R_{U,i_{odd}}}^2} = WDMA_{u_{PartB_{odd}}}$.

iii. Since the counts of odd indexed items, $R_{U,i_{odd}}$, are known by Part B, for Part A to calculate the remaining partial sum values (which are $WDA_{u_{PartA_{odd}}}$ and $WDMA_{u_{PartA_{odd}}}$), she should collaborate with Part B. Hence, Part A encrypts each $|r_{u,i} - \bar{r_i}|$ value that she calculated for the odd indexed items, and sends these encrypted values to Part B.

iv. Knowing the $R_{U,i_{odd}}$ values, Part B can get $1/R_{U,i_{odd}}$, and $1/{R_{U,i_{odd}}}^2$ exponent of the encrypted values, which are equal to $\varepsilon_K\left(\frac{x}{R_{U,i_{odd}}}\right)$, and $\varepsilon_K\left(\frac{x}{{R_{U,i_{odd}}}^2}\right)$ as shown in Eq. 6.8, and Eq. 6.9, where $x$ is the absolute difference $|r_{u,i} - \bar{r_i}|$ that Part A encrypted before sending to Part B.

$$\begin{aligned} \varepsilon_K\left(|r_{u,i} - \bar{r_i}|\right)^{1/R_{U,i_{odd}}} \; mod\, n^2 &\equiv \varepsilon_K\left(|r_{u,i} - \bar{r_i}| * \frac{1}{R_{U,i_{odd}}}\right) mod\, n \\ &= \varepsilon_K\left(\frac{|r_{u,i} - \bar{r_i}|}{R_{U,i_{odd}}}\right) \end{aligned} \tag{6.8}$$

$$\begin{aligned} \varepsilon_K\left(|r_{u,i} - \bar{r_i}|\right)^{1/R_{U,i_{odd}}{}^2} \; mod\, n^2 &\equiv \varepsilon_K\left(|r_{u,i} - \bar{r_i}| * \frac{1}{R_{U,i_{odd}}{}^2}\right) mod\, n \\ &= \varepsilon_K\left(\frac{|r_{u,i} - \bar{r_i}|}{R_{U,i_{odd}}{}^2}\right) \end{aligned} \tag{6.9}$$

v. Moreover, now Part B has the encrypted $\varepsilon_K\left(\frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}}\right)$, and $\varepsilon_K\left(\frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}{}^2}\right)$ values of each user, so she can multiply these encrypted values respectively to obtain their sum, again in encrypted form, which are $\varepsilon_K\left(\sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}}\right)$ and $\varepsilon_K\left(\sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}{}^2}\right)$ values.

vi. Then, Part B sends $\varepsilon_K\left(\sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}}\right)$ and $\varepsilon_K\left(\sum_{i=0}^{N_{u_{PartA_{odd}}}} \frac{|r_{u,i}-\bar{r_i}|}{R_{U,i_{odd}}{}^2}\right)$ values that she computed for each user back to Part A.

vii. Part A decrypts them, and hence obtains $WDA_{u_{PartA_{odd}}}$, and $WDMA_{u_{PartA_{odd}}}$ values. Besides, adds these values to the partial sum values that she already have for the even indexed items.

- Part A will add $WDA_{u_{PartA_{even}}}$ computed in step ii, and $WDA_{u_{PartA_{odd}}}$ values to get $WDA_{u_{PartA}}$.
- Part A will add $WDMA_{u_{PartA_{even}}}$ computed in step ii, and $WDMA_{u_{PartA_{odd}}}$ values to get $WDMA_{u_{PartA}}$.

viii. Now step iii through vii should be repeated ones more by switching the roles between the parts to compute $WDA_{u_{PartB}}$, and $WDMA_{u_{PartB}}$.

So far, for each user half of $WDA_u$ attribute ($WDA_{u_{PartA}}$), and half of the numerator of $WDMA_u$ attribute is known by Part A ($WDMA_{u_{PartA}}$), and the other halves are known by Part B. For instance, if parts sum up the partial values

$WDA_{u_{PartA}}$, and $WDA_{u_{PartB}}$, they can obtain $WDA_u$ value of each user. However, how should they share these partial $WDA_u$ values without violating privacy? Note that $RDMA_u$ attribute is calculated as $RDMA_u = \frac{WDA_{u_{PartA}}+WDA_{u_{PartB}}}{N_u} = \frac{WDA_u}{N_u}$, but Part A knows the even indexed users' $N_u$ values ($N_{u_{even}}$), and Part B knows the odd indexed users' $N_u$ values ($N_{u_{odd}}$). If both parts learn the $WDA_u$ and $RDMA_u$ values, then Part A may conclude $N_{u_{odd}}$, and Part B may infer $N_{u_{even}}$. To prevent this, let's share the $WDA_u$ values such that Part A will know the $WDA_u$ values of even indexed users, also since Part A knows $N_{u_{even}}$, she can easily calculate the $RDMA_u$ values of that users; and Part B will know the $WDA_u$ values of odd indexed users, also since Part B knows $N_{u_{odd}}$, she can easily calculate the $RDMA_u$ values of that users. By the same way $WDMA_u$ attribute can also be calculated. Hence, private protocol for estimating RDMA, WDMA, WDA classification attributes continues as follows:

ix. Part A sends $WDA_{u_{PartA}}$, and $WDMA_{u_{PartA}}$ values of odd indexed users to Part B, and Part B sends $WDA_{u_{PartB}}$, and $WDMA_{u_{PartB}}$ values of even indexed users to Part A.

x. Now, as shown in Eq. 6.3, Part A sums up $WDA_{u_{PartA}}$ values that she has for the even indexed users with the $WDA_{u_{PartB}}$ values that Part B has send in the previous step to obtain $WDA_u$ values of even indexed users. Similarly, Part B sums up $WDA_{u_{PartB}}$ values that she has for the odd indexed users with the $WDA_{u_{PartA}}$ values that Part A has send in the previous step to obtain $WDA_u$ values of odd indexed users.

   The numerator of the $WDMA_u$ attribute shown in Eq. 6.5, can be computed in the same manner. Each part add up the the partial numerator value that the other part sends with the partial value that she has. Thus, Part A can sum up $WDMA_{u_{PartA}}$ values that she has for the even indexed users with the $WDMA_{u_{PartB}}$ values that Part B has send to obtain the numerator of the $WDMA_u$ attribute of even indexed users. Likely, Part B will have the numerator of the $WDMA_u$ attribute of odd indexed users. According to the formula of the $WDMA_u$ attribute, this numerator should be divided by $N_u$. Thus, each part can compute the $WDMA_u$ attribute for her part. Part A knows the numerator of the $WDMA_u$ attribute of the even indexed users, and the $N_{u_{even}}$, Part B knows the numerator of the $WDMA_u$ attribute of the odd indexed users, and the $N_{u_{odd}}$. Hence, Part A can calculate $WDMA_{u_{even}}$, and Part B can compute $WDMA_{u_{odd}}$ by dividing the numerator values by the $N_u$ values they own.

Hence, at the end of this step, even indexed users WDA and WDMA attributes are known by Part A ($WDA_{u_{even}}$, $WDMA_{u_{even}}$), and odd indexed users WDA and WDMA attributes are known by Part B ($WDA_{u_{odd}}$, $WDMA_{u_{odd}}$). In other words, half of the attribute values are known by one part and the other half is known by the other.

xi. Now on, since each part knows half of the users $WDA_u$ and $N_u$ values, they can compute these users $RDMA_u$ attributes by themselves. The RDMA attribute shown in Eq. 6.4 can also be written as $WDA_u/N_u$. Part A knows $WDA_{u_{even}}$ and $N_{u_{even}}$, so for the even indexed users, Part A can calculate $RDMA_{u_{even}}$ by dividing $WDA_{u_{even}}$ by $N_{u_{even}}$, where $RDMA_{u_{even}} = \frac{WDA_{u_{even}}}{N_{u_{even}}}$. Likely, since Part B knows $WDA_{u_{odd}}$ and $N_{u_{odd}}$, for the odd indexed users Part B can calculate $RDMA_{u_{odd}}$ by dividing $WDA_{u_{odd}}$ by $N_{u_{odd}}$, where $RDMA_{u_{odd}} = \frac{WDA_{u_{odd}}}{N_{u_{odd}}}$. At the end of this step, even indexed users RDMA attributes are known by Part A ($RDMA_{u_{even}}$), and odd indexed users RDMA attributes are known by Part B ($RDMA_{u_{odd}}$).

### 6.2.4. Private estimation of DegSim attribute

*DegSim* attribute is based on the average similarity of a profile's top $k$ nearest neighbours. In order to calculate this attribute, as a first step, similarity values between a user and each of the remaining users need to be calculated. Then, from these similarity values highest $k$ of them should be chosen. In this study, user profiles are arbitrarily distributed between two parties, thus similarity computation between two users should be perform privately on ADD. For similarity computation, the *Private Similarity Computation Protocol (PrivateSims)* proposed by Memiş and Yakut (2014), which is designed for distributed Pearson similarity calculation on ADD between two parts, without jeopardizing privacy, is employed. With *PrivateSims* protocol, each part privately computes components of a similarity value ($w_{u,a}$), and end up with half of the total values between user $u$ and each of the remaining users $a$ (Memiş and Yakut, 2014). Hence, at the end, each part will have half of the similarities. Then, in order to calculate *DegSim* attribute, each part need to find top $k$ similar users, in other words, highest $k$ similarity values in that part, by using the $w_{u,a}$ values that they own, since one part might possess all the highest $k$ values. Thereafter, one part might send her top $k$ similarity values to the other part, which will only learn the value of the similarities, not the corresponding users. The part that receives all the highest $2k$ similarity values, finds highest $k$ of them, and calculates the *DegSim* attribute of user $u$.

For each user $u$ the following is performed:

i. *PrivateSims* protocol will be applied. Half of the similarities will be known by Part A, and the others will be known by Part B.

ii. Each part finds highest $k$ similarity values that she owns.

iii. Now to calculate $DegSim_u$ attribute parts need to collaborate. Hence if the user is an even indexed user;

    - Part A sends her $k$ highest similarity values to Part B.
    - Part B merges the similarity values that she owns with the ones that Part A sends.

- Among all, Part B choses the highest $k$ value, adds them up, and divides the sum by $k$ to obtain the $DegSim_u$ attribute.

iv. For the remaining users by switching roles, step iii will be repeated, and Part A will have the $DegSim_u$ attribute of the odd indexed users.

v. Finally, each part will have $DegSim_u$ attribute for half of the users.

Different from *DegSim* attribute, *DegSim'* attribute, which is equation is shown in Eq. 2.5, takes into account the number of co-rated items between two users. By modifying *PrivateSims* protocol this attribute can be calculated by following the steps of the above protocol. However, finding co-rated items of two users whose ratings are distributed between two parts is costy, and more than that this calculation might disrupt privacy. Since privacy is the main concern, we give up this classification attribute.

### 6.2.5. Private estimation of length variance attribute

For to calculate the *LengthVar* attribute for each user, the $\overline{\#ratings}$ (or, $\overline{N}$) value shown in Eq. 2.6 should be computed first. At the end of the *Revised Private Mean Estimation Protocol*, Part A learns the total number of ratings given by the even indexed users $N_{u_{even}}$ (or, $count_{even}$), and Part B learns the total number of ratings given by the odd indexed users $N_{u_{odd}}$ (or, $count_{odd}$). First of all, the average number of ratings in the system, $\overline{N}$, should be calculated, which is distributed calculation can be formulated as below, and after then, parts can calculate the *LengthVar* attribute.

$$\overline{N} = \frac{\sum_{i=0}^{N} N_u}{N} = \frac{\sum_{u \epsilon N_{u_{even}}} N_u + \sum_{u \epsilon N_{u_{odd}}} N_u}{N} = \frac{count_{even} + count_{odd}}{N} \tag{6.10}$$

i. Each part can add the number of ratings that they have. Hence, by adding the even indexed users $N_{u_{even}}$ values, Part A will have the $count_{even}$ value, which is the sum of the total number of ratings in the system given by even indexed users, and by adding the odd indexed users $N_{u_{odd}}$ values, Part B will have $count_{odd}$ value, which is the sum of the total number of ratings in the system given by odd indexed users.

ii. Then, parts exchange these count values. Part A sends $count_{even}$ value to Part B, and Part B sends $count_{odd}$ value to Part A.

iii. Since both parts knows $N$, by adding the count values $count_{even} + count_{odd}$, and dividing this sum to $N$, each part compute the $\overline{N}$ value.

iv. By knowing all the even indexed users $N_{u_{even}}$ values, Part A can calculate $\left|N_u - \overline{N}\right|$ difference for even indexed users, likewise, by knowing all the odd indexed users $N_{u_{odd}}$ values, Part B can calculate this absolute difference for odd indexed users by herself.

v. According to Eq. 2.6, $\left|N_u - \overline{N}\right|$ difference should be divided by $\sum_{i=0}^{N} \left(N_i - \overline{N}\right)^2$. This denominator can be calculated partially by each part. Thus, Part A can calculate partial sum for even indexed items, and Part B can calculate the partial sum for odd indexed items.

vi. Then, parts can exchange these partial sum values. Hence, Part A sends $\sum_{i=0}^{N_{even}} \left(N_i - \overline{N}\right)^2$ value to Part B, and Part B sends $\sum_{i=0}^{N_{odd}} \left(N_i - \overline{N}\right)^2$ value to Part A.

vii. By adding the partial sum value the part has with the partial sum value the other part sends, each part can find the denominator of the *LengthVar* attribute, on other words each part will have the $\sum_{i=0}^{N} \left(N_i - \overline{N}\right)^2$ value.

viii. Then, by dividing the $\left|N_u - \overline{N}\right|$ differences that they calculated to the $\sum_{i=0}^{N} \left(N_i - \overline{N}\right)^2$ value, Part A will have the $LengthVar_u$ attribute of the even indexed users, and Part B will have the $LengthVar_u$ attribute of the odd indexed users.

### 6.2.6. Private estimation of average attack model-specific attributes

To find the average attack model specific attributes, namely *FMV*, *FMD*, and *Profile Variance*, described thoroughly in Section 2.3.2, first of all we need to find the optimal partitioning, which is the one where the mean variance is minimized, and calculated as in Eq. 2.7.

For each user $u$ the following steps will be performed:

i. Each part can create her own possible target item set by herself.

   - By examining user $u$'s profile that she owns, Part A constructs the set of possible target items $(P_{u,T_A})$ from the rated items in the profile having maximum rating $(r_{max})$.
   - Similarly, Part B constructs the set of possible target items $(P_{u,T_B})$ for her own part.

ii. Now, each part need to calculate the $MeanVar(u, p_{target})$ attribute for each of the items in their possible target item sets' iteratively. Through *Revised Private Mean Estimation Protocol* both parts know the average rating of each item, thus the $\overline{r_i}$ values. According to Eq. 2.7, parts need to calculate the square of the difference of the filler ratings, but some of the filler ratings are in one part, and the other are in the other part. Hence, Eq. 2.7 can be calculated in distributed manner as follows:

$$MeanVar\,(u, p_{target}) = \frac{\sum_{i \epsilon P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2 + \sum_{i \epsilon P_{u,F_B}} (r_{u,i_B} - \overline{r_i})^2}{|P_{u,F_A}| + |P_{u,F_B}|} \tag{6.11}$$

Thus, for calculating the MeanVar values of the possible target items in $P_{u,T_A}$, Part A should consider all the rated items in Part B along with the ones that were rated in her side except the chosen target item as her filler item set, $P_{u,F_A}$; and similarly Part B should consider all the rated items in Part A along with her $P_{u,F_B}$ set as her filler items in calculating the MeanVar values of the possible target items in $P_{u,T_B}$. By looking at all the rated items they have, parts needs to compute $\sum_{i \epsilon P_{u,F}} (r_{u,i} - \overline{r_i})^2$ value for their part, and then, exchange these values with each other.

- Part A computes $\sum_{i \epsilon P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2$ value, let's name as $MeanVarNumerator_A$, and sends to Part B.
- Part B computes $\sum_{i \epsilon P_{u,F_B}} (r_{u,i_B} - \overline{r_i})^2$ value, let's name as $MeanVarNumerator_B$, and sends to Part A.

iii. Iterating through the possible target item set's, each part calculates the numerator of the $MeanVar\,(u, p_{target})$ values' in her part.

(a) For each of the elements in $P_{u,T_A}$, iteratively take one of them as the suspected target item, and find the numerator of the $MeanVar(u, p_{target})$ values'.

- Construct the $P_{u,F_A}$, which is the rest of the rated items in user $u$'s profile own by Part A, plus the ones in Part B.
- Calculate $\sum_{i \epsilon P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2$, which will be the $MeanVarNumerator_A$ value for the corresponding suspected target item.
- Add the obtained $MeanVarNumerator_A$ with $MeanVarNumerator_B$ value that Part B send in step ii, where the result will be 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 optimal partitioning according to Part A. Even though 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.

(c) Steps (a), (b) will repeated by Part B. So at the end of these steps Part B also learns the optimal partitioning according to Part B.

iv. Now, there are two values for optimal partitioning. Among these the one having the minimum value is the optimal partitioning for user $u$. So, parts exchange these values, and both learn if the optimal partitioning is in her part or the other. The part which has the optimal partitioning, increases by one the TMF value of the target item that gives this partitioning.

v. Now optimal partitioning is in which part is known. However, actual *MeanVar* attribute is not calculated yet. Only the numerator is obtained. To obtain the *MeanVar* value, this numerator should be divided by the size of the filler items, $|P_{u,F_A}| + |P_{u,F_B}|$. This is actually equal to the total number of items rated by the user in both Part A, and Part B, which is the $N_u$ value that we calculated in the *Private Mean Estimation Protocol*, minus 1, indicating the target item. However, at the end of this protocol, Part A knows $N_u$ value of the even indexed users, and Part B knows $N_u$ value of the odd indexed users.

- If optimal partitioning is in Part A, and the working user is an odd indexed user:
    - For calculating $FMV_u$ attribute, Part A sends the numerator value she has to Part B. Part B already knows the $N_u$ value of the user, hence calculates the $FMV_u$ of the user. Part B cannot know which item is the possible target, can only learn the value of the attribute.
- If optimal partitioning is in Part B, and the working user is an even indexed user:
    - For calculating $FMV_u$ attribute, Part B sends the numerator value she has to Part A. Part A already knows the $N_u$ value of the user, hence calculates the $FMV_u$ of the user. Part A cannot know which item is the possible target, can only learn the value of the attribute.
- If optimal partitioning is in Part A, and the working user is an even indexed user:
    - For calculating $FMV_u$ attribute, since Part A already knows the $N_u$ value of the user, she calculates the $FMV_u$ of the user.
- If optimal partitioning is in Part B, and the working user is an odd

indexed user:

- For calculating $FMV_u$ attribute, since Part B already knows the $N_u$ value of the user, she calculates the $FMV_u$ of the user.

*FMD* and *Profile Variance* are the other attributes that need to be computed based on the optimal partitioning, which is found in step iv of the above protocol. These attributes can also be computed in distributed manner. Eq. 2.8 can be written as follows:

$$FMD_u = \frac{\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}| + \sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|}{|P_{u,F_A}| + |P_{u,F_B}|}$$
$$= \frac{\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}| + \sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|}{N_u - 1} \tag{6.12}$$

In Eq. 2.9, *PMean* is the average of ratings provided by a user across all items in both parts. This is actually the $\overline{v_u}$ value that we have calculated in the *Revised Private Mean Estimation Protocol.* At the end of this protocol, each part learns the $\overline{v_u}$ values of all users, hence each part already knows *PMean.* Moreover, $i \in P_{u,T} \cup P_{u,F}$ means all the items that have rated in the profile, both in Part A, and B. Hence, unlike the formula of *FMD*, and *MeanVar*, in *Profile Variance* the possible target item is also take into consideration. So, $i \in P_{u,T} \cup P_{u,F}$ is same as $i \in P_u - P_{u,\emptyset}$. $|P_{u,T} \cup P_{u,F}|$ in the denominator is actually $|P_{u,F}| + |P_{u,T}|$. Since, $|P_{u,T}|$ is 1, it turned to be $|P_{u,F}| + 1$, which is equal to $N_u$ value. As stated above, Part A knows even indexed users $N_u$ value, and Part B knows odd indexed users $N_u$ value. Eq. 2.9 can be written in distributed manner as follows:

$$ProfileVariance_u = \frac{\sum\limits_{i_A \epsilon P_{u_A} - P_{u,\emptyset_A}} (r_{u,i_A} - PMean)^2 \quad + \sum\limits_{i_B \epsilon P_{u_B} - P_{u,\emptyset_B}} (r_{u,i_B} - PMean)^2}{N_u} \tag{6.13}$$

These two attributes can also be computed concurrently while calculating the $FMV_u$ attribute of the user. In the final step of the above private protocol, we can also calculate *FMD*$_u$ and *Profile Variance*$_u$ attributes.

- If optimal partitioning is in Part A, and the working user is an odd indexed user:

  - For calculating *FMD*$_u$ attribute, Part A calculates $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part A except the target item, and sends this value to Part B. Part B calculates $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part B. Part B adds up these partial sums, and then divides by $N_u - 1$ to obtain the *FMD*$_u$ attribute.
  - For calculating *Profile Variance*$_u$, Part A calculates $\sum_{i_A \epsilon P_{u_A} - P_{u,\varnothing_A}} (r_{u,i_A} - PMean)^2$ value for all the rated items that she has, including the target item, and sends this value to Part B. Part B calculates $\sum_{i_B \epsilon P_{u_B} - P_{u,\varnothing_B}} (r_{u,i_B} - PMean)^2$ value on all the rated items in her part. Part B adds up these partial sums, and then divides by $N_u$ to obtain the *Profile Variance*$_u$ attribute.

- If optimal partitioning is in Part B, and the working user is an even indexed user:

  - For calculating *FMD*$_u$ attribute, Part B calculates $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part B except the target item, and sends this value to Part A. Part A calculates $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part A. Part A adds up these partial sums, and then divides by $N_u - 1$ to obtain the *FMD*$_u$ attribute.
  - For calculating *Profile Variance*$_u$, Part B calculates $\sum_{i_B \epsilon P_{u_B} - P_{u,\varnothing_B}} (r_{u,i_B} - PMean)^2$ value for all the rated items that she has, including the target item, and sends this value to Part A. Part A calculates $\sum_{i_A \epsilon P_{u_A} - P_{u,\varnothing_A}} (r_{u,i_A} - PMean)^2$

value on all the rated items in her part. Part A adds up these partial sums, and then divides by $N_u$ to obtain the *Profile Variance*$_u$ attribute.

- If optimal partitioning is in Part A, and the working user is an even indexed user:

  - For calculating $FMD_u$ attribute, Part B calculates $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part B, and sends this value to Part A. Part A calculates $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part A except the target item. Part A adds up these partial sums, and then divides by $N_u - 1$ to obtain the $FMD_u$ attribute.
  - For calculating *Profile Variance*$_u$ attribute, Part B calculates $\sum_{i_B \epsilon P_{u_B} - P_{u,\emptyset_B}} (r_{u,i_B} - PMean)^2$ value for all the rated items that she has, and sends this value to Part A. Part A calculates $\sum_{i_A \epsilon P_{u_A} - P_{u,\emptyset_A}} (r_{u,i_A} - PMean)^2$ value on all the rated items in her part, including the target item. Part A adds up these partial sums, and then divides by $N_u$ to obtain the *Profile Variance*$_u$ attribute.

- If optimal partitioning is in Part B, and the working user is an odd indexed user:

  - For calculating $FMD_u$ attribute, Part A calculates $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part A, and sends this value to Part B. Part B calculates $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$ value for her filler item set, which include all the items that have rating on Part B except the target item. Part B adds up these partial sums, and then divides by $N_u - 1$ to obtain the $FMD_u$ attribute.
  - For calculating *Profile Variance*$_u$ attribute, Part A calculates $\sum_{i_A \epsilon P_{u_A} - P_{u,\emptyset_A}} (r_{u,i_A} - PMean)^2$ value for all the rated items that she has, and sends this value to Part B. Part B calculates $\sum_{i_B \epsilon P_{u_B} - P_{u,\emptyset_B}} (r_{u,i_B} - PMean)^2$

value on all the rated items in her part, including the target item. Part B adds up these partial sums, and then divides by $N_u$ to obtain the *Profile Variance*$_u$ attribute.

As well as learning the $FMV_u$ attribute, parts also learn the value of $FMD_u$, and *Profile Variance*$_u$ attribute for half of the users. This protocol needs to be computed twice; ones for push attack, where the set $P_{u,target}$ contains items having rating $r_{max}$, and ones for nuke attacks, where the set $P_{u,target}$ contains items having rating $r_{min}$. Thus, by choosing the minimum ratings of the profile as the target set, these metrics need to be computed ones more. Hence, the above protocol written for calculating the average attack model-specific attributes can be used to find the corresponding attributes for nuke attacks.

### 6.2.7. Private estimation of bandwagon attack model-specific attributes

As in random attack, *FAC*, and *FMD* attributes need to be generated, but different from random attack, the partitioning shown in Eq. 2.10 will be used in calculations. Hence, all items in $P_u$ that are given the maximum rating (or the minimum rating for nuke attack) in 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 average ratings of items is known by both parts, this correlation value can be calculated distributively between two parts, as shown in Eq. 6.14.

$$FAC_u = \frac{X \cdot Y}{\sqrt{X^2} \cdot \sqrt{Y^2}} = \frac{X_A \cdot Y_A + X_B \cdot Y_B}{\sqrt{{X_A}^2 + {X_B}^2} \cdot \sqrt{{Y_A}^2 + {Y_B}^2}} \tag{6.14}$$

where, $X_A$, and $X_B$ are the vectors having the ratings given to filler items in Part A, and B, respectively. Whereas, $Y_A$, and $Y_B$ are the vectors containing the average ratings of the corresponding items, in order of Part A, and B.

Calculation of the $FAC$ attribute, which is distributed equation is given in Eq. 6.14 can be obtained privately for bandwagon attack model as follows:

For each user $u$ the following steps will be performed:

i. Looking at their own data, each part constructs the filler item and target item sets of user $u$.

   - Part A forms the set $P_{u,F_A}$, and Part B forms the set $P_{u,F_B}$.
   - The ratings given by user $u$ to the items in $P_{u,F_A}$ is actually the $X_A$ vector, and similarly the ratings given to items in $P_{u,F_B}$ is the $X_B$ vector. Hence, parts form $X_A$, and $X_B$ vectors by themselves.
   - Each part increases the TMF value of the items in her target item list by one.

ii. Since parts already knows the item means, they can form $Y_A$, and $Y_B$ vectors, which will contain the corresponding item mean values of the filler items in that part.

iii. Now, by knowing $X_A$ and $Y_A$, Part A can calculate $X_A \cdot Y_A$, ${X_A}^2$, and ${Y_A}^2$. Similarly, Part B calculates $X_B \cdot Y_B$, ${X_B}^2$, and ${Y_B}^2$.

*If the user is an even indexed user, then the following steps are obeyed.*

iv. Part A sends the value of ${Y_A}^2$ to Part B. By using ${Y_B}^2$ value that she has, Part B can calculate $\sqrt{{Y_A}^2 + {Y_B}^2}$.

v. Part B sends the value of ${X_B}^2$ to Part A. By using ${X_A}^2$ value that she has, Part A can calculate $\sqrt{{X_A}^2 + {X_B}^2}$.

vi. Part B encrypts the $\sqrt{{Y_A}^2 + {Y_B}^2}$ value that she obtained in step iii with her key, and sends $\xi\left(\sqrt{{Y_A}^2 + {Y_B}^2}\right)$ to Part A.

vii. According to Eq. 6.14, the denominator of $FAC_u$ is the product of $\sqrt{{X_A}^2 + {X_B}^2}$ and $\sqrt{{Y_A}^2 + {Y_B}^2}$. Part A knows the value of $\sqrt{{X_A}^2 + {X_B}^2}$. Moreover, in the previous step Part B send the encrypted value of $\sqrt{{Y_A}^2 + {Y_B}^2}$, which is $\xi\left(\sqrt{{Y_A}^2 + {Y_B}^2}\right)$ to Part A. Hence, Part A can get the $\sqrt{{X_A}^2 + {X_B}^2}$ exponent of this encrypted value to obtain the encrypted value of their product, which is the denominator.

$$\xi_K\left(\sqrt{{Y_A}^2 + {Y_B}^2}\right)^{\left(\sqrt{{X_A}^2 + {X_B}^2}\right)} \; mod\, n^2 \equiv \xi_K\left(\sqrt{{X_A}^2 + {X_B}^2} \cdot \sqrt{{Y_A}^2 + {Y_B}^2}\right) \; mod\, n \tag{6.15}$$

viii. According to Eq. 6.14, the numerator of $FAC_u$ is the sum of $X_A \cdot Y_A$ and $X_B \cdot Y_B$. However, $X_A \cdot Y_A$ is known by Part A, and $X_B \cdot Y_B$ is known by Part B. Hence, Part B encrypts the $X_B \cdot Y_B$ value that she obtained in step i with her key, and sends $\xi\left(X_B \cdot Y_B\right)$ to Part A.

ix. Part A encrypts the value of $X_A{\cdot}Y_A$, and multiplies $\xi\left(X_A \cdot Y_A\right)$ with $\xi\left(X_B \cdot Y_B\right)$ in order to obtain their encrypted sum, which is $\xi\left(X_A \cdot Y_A + X_B \cdot Y_B\right)$.

x. Now Part A has both the numerator and denominator value of the $FAC_u$ attribute in encrypted format. Thus, by using *Secure Division Protocol* (Dahl et al., 2012), Part B can divide $\xi\left(X_A \cdot Y_A + X_B \cdot Y_B\right)$ to $\xi_K\left(\sqrt{{X_A}^2 + {X_B}^2} \cdot \sqrt{{Y_A}^2 + {Y_B}^2}\right)$, and send the result to Part B.

xi. Part B decrypts the result, and gets the $FAC_u$ attribute of the even indexed users.

xii. By changing places in step iv to xi, Part A can calculate the $FAC_u$ attribute of the odd indexed users (*If the user is an odd indexed user, then this step is obeyed.*).

Calculation of the *FMD* attribute, which is distributed equation is given in Eq. 6.12 can be obtained privately for bandwagon attack model as follows:

For each user $u$ the following steps will be performed:

i. Each part constructs her filler item and target item sets according to her own data.

   - By constructing the filler sets, each part also learns the size of their filler set. Thus, Part A forms the set $P_{u,F_A}$, and learns $|P_{u,F_A}|$. Similarly, Part B forms the set $P_{u,F_B}$, and learns $|P_{u,F_B}|$.
   - Each part increases the TMF value of the items in her target item list by one.

ii. Parts calculate $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$, and $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$ values by themselves based on their own filler item set's.

iii. According to Eq. 6.12 partial sum values need to be add up, and then divided by the sum of the size (count) of the filler item sets, $|P_{u,F_A}| + |P_{u,F_B}|$. Hence, if the user is an even indexed user;

   (a) Part A encrypts the partial sum value obtained from $\sum_{i_A \epsilon P_{u,F_A}} |r_{u,i_A} - \overline{r_i}|$, and the size of her filler item set, $|P_{u,F_A}|$. Then, sends these encrypted values to Part B.

   (b) Part B encrypts her partial sum value obtained from $\sum_{i_B \epsilon P_{u,F_B}} |r_{u,i_B} - \overline{r_i}|$, and the size of her filler item set, $|P_{u,F_B}|$.

   (c) Part B multiplies these encrypted partial sum values, and obtains sum of them in encrypted form. Similarly, by multiplying the encrypted count values, Part B obtains the sum of the counts in encrypted form.

   (d) By using *Secure Division Protocol* (Dahl et al., 2012), Part B can divide encrypted sum to encrypted count, and send the result to Part A.

   (e) Part A decrypts the result, and gets the $FMD_u$ attribute of the even indexed user.

iv. By changing places in step iii, Part B can calculate the $FMD_u$ attribute of the odd indexed users.

By choosing the minimum ratings of the profile as the target set, *FAC*, and *FMD* attributes need to be computed ones more. Hence, the above protocol written for calculating the bandwagon attack model-specific attributes can be used to find the corresponding attributes for nuke attacks.

These same protocols can be used for calculating the required random attack model-specific attributes. The only difference is the target and filler item sets that is operated on. Hence, these protocols need to be computed in a loop, and the partitioning which gives the minimum *FAC* value needs to be chosen as the optimal partitioning. *FAC*, and *FMD* attributes of the corresponding partitioning is taken as the random attack model-specific attributes.

### 6.2.8. Private estimation of segment attack model-specific attributes

For segment attacks, *FMTD*, and *GFMV* attributes needs to be generated based on the partitioning shown in Eq. 2.10, as in bandwagon attack. The *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 calculated as given in Eq. 2.11. The formula of *GFMV* attribute is given in Eq. 2.12. These attributes can also be calculated in distributed manner between Part A and B.

$$FMTD_u = \left| \left( \frac{\sum_{k_A \in P_{u,T_A}} r_{u,k_A} + \sum_{k_B \in P_{u,T_B}} r_{u,k_B}}{|P_{u,T_A}| + |P_{u,T_B}|} \right) - \left( \frac{\sum_{i_A \in P_{u,F_A}} r_{u,i_A} + \sum_{i_B \in P_{u,F_B}} r_{u,i_B}}{|P_{u,F_A}| + |P_{u,F_B}|} \right) \right| \tag{6.16}$$

$$GFMV_u = \frac{\sum_{i_A \in P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2 + \sum_{i_B \in P_{u,F_B}} (r_{u,i_B} - \overline{r_i})^2}{|P_{u,F_A}| + |P_{u,F_B}|} \tag{6.17}$$

For each user $u$ the following steps will be performed:

i. Each part constructs her filler item and target item sets according to her own data.

ii. Each part increases the TMF value of the items in her target item list by one.

iii. Parts calculate some necessary partial sum values looking at their own data.

- For calculating $FMTD_u$ attribute, Part A calculates $\sum_{k_A \in P_{u,T_A}} r_{u,k_A}$, $|P_{u,T_A}|$, $\sum_{i_A \in P_{u,F_A}} r_{u,i_A}$, $|P_{u,F_A}|$. Part B calculates $\sum_{k_B \in P_{u,T_B}} r_{u,k_B}$, $|P_{u,T_B}|$, $\sum_{i_B \in P_{u,F_B}} r_{u,i_B}$, $|P_{u,F_B}|$.
- For calculating $GFMV_u$ attribute, Part A calculates $\sum_{i_A \in P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2$, and Part B calculates $\sum_{i_B \in P_{u,F_B}} (r_{u,i_B} - \overline{r_i})^2$.

iv. Now these partial sums need to be added up, and then divided by some count value according to the equations of the attributes (*If the user is an even indexed user, then the following steps are obeyed.*).

- For calculating $FMTD_u$ attribute, Part A encrypts $\sum_{k_A \in P_{u,T_A}} r_{u,k_A}$, $\sum_{i_A \in P_{u,F_A}} r_{u,i_A}$, $|P_{u,T_A}|$, and $|P_{u,F_A}|$ values.
- For calculating $GFMV_u$ attribute, Part A encrypts $\sum_{i_A \in P_{u,F_A}} (r_{u,i_A} - \overline{r_i})^2$.

v. Part A sends these encrypted values to Part B.

vi. Part B encrypts her own values, namely $\sum_{k_B \in P_{u,T_B}} r_{u,k_B}$, $\sum_{i_B \in P_{u,F_B}} r_{u,i_B}$, $|P_{u,T_B}|$, $|P_{u,F_B}|$, and $\sum_{i_B \in P_{u,F_B}} (r_{u,i_B} - \overline{r_i})^2$.

vii. According to Eq. 6.16, and 6.17, Part B performs the following computations.

- For calculating $FMTD_u$ attribute,
  (a) Part B multiplies $\varepsilon\left(\sum_{k_A \in P_{u,T_A}} r_{u,k_A}\right)$ with $\varepsilon\left(\sum_{k_B \in P_{u,T_B}} r_{u,k_B}\right)$, and obtains $\varepsilon\left(\sum_{k_A \in P_{u,T_A}} r_{u,k_A} + \sum_{k_B \in P_{u,T_B}} r_{u,k_B}\right)$.
  (b) Part B multiplies $\varepsilon\left(|P_{u,T_A}|\right)$ with $\varepsilon\left(|P_{u,T_B}|\right)$, and obtains $\varepsilon\left(|P_{u,T_A}| + |P_{u,T_B}|\right)$.

(c) Part B multiplies $\varepsilon\left(\sum_{i_A \in P_{u,F_A}} r_{u,i_A}\right)$ with $\varepsilon\left(\sum_{i_B \in P_{u,F_B}} r_{u,i_B}\right)$, and obtains $\varepsilon\left(\sum_{i_A \in P_{u,F_A}} r_{u,i_A} + \sum_{i_B \in P_{u,F_B}} r_{u,i_B}\right)$.

(d) Part B multiplies $\varepsilon\left(|P_{u,F_A}|\right)$ with $\varepsilon\left(|P_{u,F_B}|\right)$, and obtains $\varepsilon\left(|P_{u,F_A}| + |P_{u,F_B}|\right)$.

(e) By using *Secure Division Protocol* (Dahl et al., 2012), Part B divides encrypted sum that she obtains in step (a) to encrypted count value that is found in step (b), and obtains $\varepsilon\left(\frac{\sum_{k_A \in P_{u,T_A}} r_{u,k_A} + \sum_{k_B \in P_{u,T_B}} r_{u,k_B}}{|P_{u,T_A}| + |P_{u,T_B}|}\right)$.

(f) By using *Secure Division Protocol* (Dahl et al., 2012), Part B divides encrypted sum that she obtains in step (c) to encrypted count value that is found in step (d), and obtains $\varepsilon\left(\frac{\sum_{i_A \in P_{u,F_A}} r_{u,i_A} + \sum_{i_B \in P_{u,F_B}} r_{u,i_B}}{|P_{u,F_A}| + |P_{u,F_B}|}\right)$.

(g) Part B subtracts the value obtained in step (f) from the value obtained in step (e), by first multiplying the result in step (e) with -1, and then multiplying with the result in step (f) to get their sum (Parkes et al., 2008). Hence, obtains

$$\varepsilon\left(\frac{\sum_{k_A \in P_{u,T_A}} r_{u,k_A} + \sum_{k_B \in P_{u,T_B}} r_{u,k_B}}{|P_{u,T_A}| + |P_{u,T_B}|} - \frac{\sum_{i_A \in P_{u,F_A}} r_{u,i_A} + \sum_{i_B \in P_{u,F_B}} r_{u,i_B}}{|P_{u,F_A}| + |P_{u,F_B}|}\right)$$

(h) Then, Part B sends this result to Part A.

(i) Part A decrypts the result and obtains $FMTD_u$ attribute of the user.

- For calculating $GFMV_u$ attribute,

(a) Part B multiplies $\varepsilon\left(\sum_{i_A \in P_{u,F_A}} \left(r_{u,i_A} - \overline{r_i}\right)^2\right)$ by $\varepsilon\left(\sum_{i_B \in P_{u,F_B}} \left(r_{u,i_B} - \overline{r_i}\right)^2\right)$, and obtains their encrypted sum.

(b) Part B multiplies $\varepsilon\left(|P_{u,F_A}|\right)$ by $\varepsilon\left(|P_{u,F_B}|\right)$, and obtains $\varepsilon\left(|P_{u,F_A}| + |P_{u,F_B}|\right)$.

(c) By using *Secure Division Protocol* (Dahl et al., 2012), Part B divides the encrypted value that she obtains in step (a) to $\varepsilon\left(|P_{u,F_A}| + |P_{u,F_B}|\right)$, which is obtained in step (b).

(d) Then, Part B sends the result to Part A.

(e) Part A decrypts the result and obtains $GFMV_u$ attribute of the user.

viii. By changing places in step iii, Part B can calculate the $FMTD_u$ and $GFMV_u$ attributes of the odd indexed users (*If the user is an odd indexed user, then this step is obeyed.*).

## 6.3. Classification Approach for Detection of Shilling Attacks on Arbitrarily Distributed Data

One of the challenges that separates attack classification from traditional classification problems is stated by Williams et al. (2007) as follows: "The exponential number of combinations of attack types, possible attack targets, and selection of segment and filler items makes it infeasible to enumerate a training set using the ratings profiles alone". Hence, in order to generalize the idea of an authentic or attack profile beyond the raw ratings data, Williams et al. (2007) try to capture the statistical features together with some other detection attributes that describe the signature of a profile. Therefore, their detection model is based on the construction of detection attributes that are calculated for each profile in the database, which are then used to build a classifier by using supervised learning methods (Burke et al., 2006a).

In order to apply this shilling attack detection approach on ADD, the derived classification attributes, which are described in details in Section 2.3, should be calculated collaboratively between two parties without revealing privacy. However, in calculation of some of these attributes, especially the generic attributes, at first, some of the basic primitives, such as user and item mean, should be known by the collaborative parts. Due to privacy constraints, parts primarily mask their own data by using the RF method. After that, they compute these basic primitives by applying the Revised Mean Estimation Protocol. As a result, even indexed users' and items' primitive values are known by Part A, and the odd ones are known by Part B. All the necessary detection attributes are then generated for each user profile distributively on ADD by performing each private attribute estimating protocol consecutively. These private protocols do not compromise on security, since secure computations based on homomorphic encryption techniques are used. At the end, some of the classification attributes calculated for a user are obtained by one part, and the remaining attributes are acquired by the other part. According to the chosen classifier learning algorithm, which is $k$-NN in this study, the classifier model is constructed by exchanging the attributes. In this point, since all the attributes are known by both parts, they can build the model individually. Even though both parts will have the model, in order to classify a new instance, collaboration between the parties is essential. In other words, for parts to use the model to test for new instances, they must collaborate, and calculate these classification attributes for those instances by applying the private protocols, which were used in training.

## 6.4. Experimental Evaluation

Real data-based experiments are presented to evaluate the performance of the classification based attack detection method, which utilizes the proposed private protocols for finding the required classification attributes, in identification of distributed shilling attacks privately on ADD. Moreover, the need for collaboration in attack detection on ADD is shown up empirically.

### 6.4.1. Data set and evaluation criteria

In the experiments, publicly available MovieLens 100K[1] dataset, which has been used in most research on attack detection, is applied. This dataset consists of 100,000 ratings on 1,682 movies by 943 users, such that at least 20 movies are rated by each user. Within the dataset, all ratings are integer values between one and five, where one is the lowest rating (disliked) and five is the highest (most liked).

In the literature, there are various measures that are used to compare different detection algorithms (Burke et al., 2015). Among all, for measuring the classification performance of the classifiers, standard measurements of *Recall* and *Precision*, which

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

**Table 6.1.** Confusion Matrix

| | | **Actual Condition** | |
|---|---|---|---|
| | | **Attack** | **Authentic** |
| **Predicted Condition** | **Attack** | TP | FP |
| | **Authentic** | FN | TN |

$$Recall \equiv Sensitivity = \frac{TP}{TP + FN} \tag{6.18}$$

$$Precision \equiv PPV = \frac{TP}{TP + FP} \tag{6.19}$$

are commonly used performance measures in information retrieval area, are chosen. Since *Recall* and *Precision* metrics enable the comparison of any two classifiers having the same target output, comparison of the obtained results with the ones in the literature become possible. The interest in shilling attack detection is how well the classification algorithms detect attacks, therefore both of these metrics are utilized with respect to attack identification (Burke et al., 2006a). More specifically, in attack detection while taking a "positive" classification means labelling a profile as an *Attack*, taking a "negative" classification means labelling a profile as an *Authentic* (Burke et al., 2015). Table 6.1 shows the corresponding confusion matrix. In the confusion matrix, *True Positive (TP)* shows the profiles that are correctly classified as *Attack*, whereas *True Negative (TN)* shows the profiles that are correctly classified as *Authentic*. Similarly, *False Positive (FP)* shows the profiles that are incorrectly classified as *Attack*, and *False Negative (FN)* shows the profiles that are incorrectly classified as *Authentic*.

On the basis of the confusion matrix shown in Table 6.1, the equations of *Recall*, and *Precision* can be given as in Eq. 6.18, and Eq. 6.19, respectively. As can be seen from the equations, *Recall*, and *Precision* are used for measuring the performance of the classifier in identifying attack profiles.

*Recall* metric, which is also called as *Sensitivity*, measures the number of attack profiles correctly classified as a fraction of the total number of actual attacks in the system. *Precision* metric, which is also named as the *Positive Predictive Value* (PPV), measures the number of attack profiles correctly classified as a fraction of the total number of profiles labelled as attack.

### 6.4.2. Experimental methodology

Currently, there exists no arbitrarily distributed data available in the literature. MLP is also a centralized dataset. Hence, in order to obtain an arbitrarily distributed data, this dataset is separated into two parts. During the partitioning process, a random number (r) is selected for each user. According to this number, r of the user ratings are put into first part (Part A), and the remaining ones are put into the other part (Part B). Consequently, all the users and all the items of MLP dataset take part in both of the obtained datasets. Hence, from the centralized $943 \times 1,682$ MLP dataset consisting of 100,000 ratings, two $943 \times 1,682$ datasets having totally 100,000 ratings, are obtained. This approach yields randomly distributed data between Part A and Part B. In all the experiments, this partitioning is used.

#### 6.4.2.1 *Experimental setup 1: Can collaborating parties detect distributed attacks alone?*

In this experiment, it is assumed that Part A and Part B are collaborating on ADD to provide predictions for their customers, and one of the proposed PPDCF schemes for ADD is employed between them. For attack detection, it is assumed that Part A and Part B are not collaborating, instead they have their own detection mechanisms with respect to their own data. As a defence to shilling attacks, each part utilizes one of the existing detection methods proposed for centralized data, where classification-based attack detection method introduced by Williams et al. (2007) is considered in the following experiments.

Under these circumstances, the question is whether Part A and Part B can detect the distributed attack profiles on ADD individually. More specifically, if distributed attack profiles are injected into the system by a malicious user who knows the collaboration of the parts, whether the detection algorithm of Part A can detect the distributed attack profiles by herself. Similarly, whether the detection algorithm of Part B, which is trained based on only Part B's data, can be successful in detecting distributed attacks.

In order to make comparisons, outcomes of the classification-based attack detection method introduced by Williams et al. (2007) is chosen as the baseline. In the baseline case, alias in the experiments executed by Williams et al. (2007), data is centralized and all attack detection process is under control of a single data holder. To replicate the baseline, experimental methodology similar to the one conducted by the authors is employed on all MLP dataset (*MLP*). Same train and test processes are applied to *MLP* as the authors, thus all the classification attributes are derived based on whole data.

For the case of baseline, the following methodology is employed on *MLP* dataset. In order to have distinct training and test sets for attack detection and response experiments, one-third of the user profiles in the *MLP* dataset are chosen as test profiles, and the remaining as train profiles. User profiles chosen for training are used to create training data of the Baseline classifier. For each test the second part of the data, which is chosen for testing, is injected with attack profiles. Then run through the Baseline classifier that had been trained on the augmented first part. The training data of the Baseline classifier is created by inserting a mix of the attack models for both push and nuke attacks at various filler sizes that ranged from 3 to 100% and attack sizes between 0.5 and 1%. The attacked movies in the training set is chosen at random from movies that had between 80 and 100 ratings. Specifically the training data is created by inserting the first attack at a particular filler size, and generating the detection attributes for the authentic and attack profiles. This process is repeated 21 more times for additional attack types and/or filler sizes, and generating the detection attributes separately. For all these subsequent

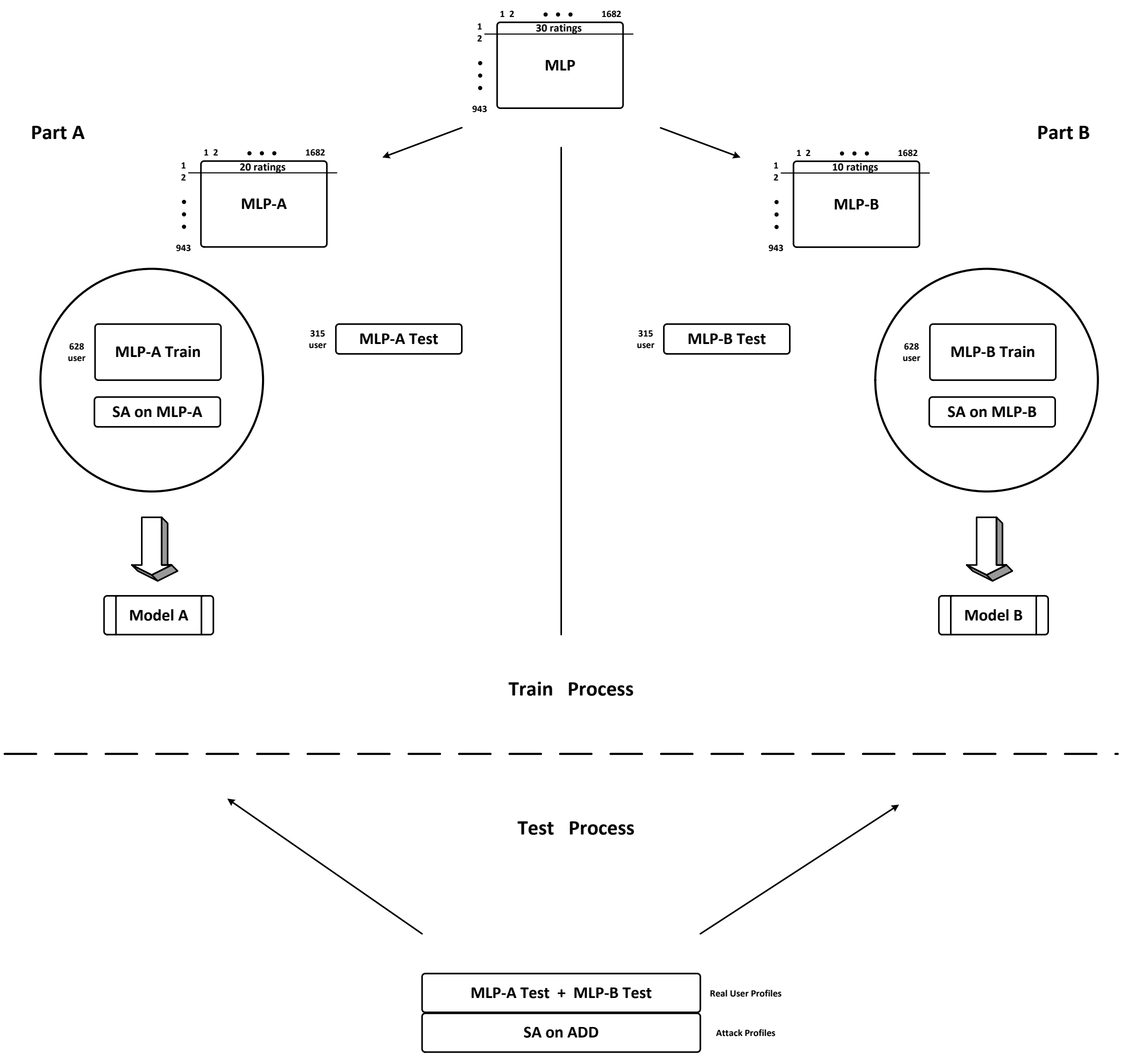


**Figure 6.1.** Experimental Setup 1: Can Collaborating Parties Detect Distributed Attacks Alone?

attacks, the detection attributes of only the attack profiles are then added to the original detection attribute dataset. Approximately same results are obtained as Williams et al. (2007), which are named as "Baseline" in the empirical results given in Section 6.4.3.1.

The same experimental methodology is also employed separately for Part A's and Part B's data, which are the ADD case of MLP and obtained by random partitioning of MLP dataset as explained at the beginning. Fig. 6.1 demonstrates train and test processes carried out by each part, namely Part A and PartB. Part A's dataset is shown as *MLP-A* and Part B's dataset is shown as *MLP-B* in Fig. 6.1.

*MLP-A* and *MLP-B* are the arbitrarily distributed version of *MLP* dataset between Part A and Part B. Similar to the experimental methodology employed in baseline, each part builds her own model; however differently from baseline, not based on all data, but based on her own data. Hence, Part A constructs *Model A* and Part B constructs *Model B* by employing same train process with the baseline, which are indicated as "Part A" and "Part B" in the empirical results given in Section 6.4.3.1, respectively.

For *MLP-A* and *MLP-B* datasets the following methodology is employed. Similar to Baseline, one-third of the user profiles are chosen as test profiles, and the remaining as train profiles for both of the datasets. *MLP-A* and *MLP-B* forms ADD case, in other words, the profiles in these datasets are distributed user profiles that are partitioned between Part A and Part B. Therefore, users that are chosen for test are same for both parts. Similarly, train users are also same. As can be seen from Fig. 6.1, Part A divides *MLP-A* dataset as *MLP-A Train* and *MLP-A Test.* Similarly, Part B divides *MLP-B* dataset as *MLP-B Train* and *MLP-B Test.* As the training process, both parts separately employ same methodology with the baseline, however base on their own data. For instance, user profiles in *MLP-A Train* are used by Part A to create training data of the *Model A* classifier. Part A's training data is created by inserting a mix of the attack models for both push and nuke attacks at various filler sizes that ranged from 3 to 100% and attack sizes between 0.5 and 1% based only on Part A's data, which is *MLP-A*. The attacked movies in the training set of Part A are selected at random from movies that had average ratings according to the data distribution of Part A. Similar to baseline, the training data of Part A is created by inserting the first attack at a particular filler size, and generating the detection attributes for the authentic and attack profiles, where this process is repeated 21 more times for additional attack types and/or filler sizes, and generating the detection attributes separately. For all these subsequent attacks, the detection attributes of only the attack profiles are then added to the original detection attribute dataset. The training process described for Part A is also employed individually by Part B to construct the *Model B* classifier. Differently from the baseline, the test profiles are generated based on MLP dataset, not on partial data, and then distributed between the parts. Hence, for each test, data is injected with distributed attack profiles injected by the proposed design strategy for ADD in Part A and PartB. Then run through corresponding classifier, either *Model A* or *Model B* classifier. In order to compare the results with the baseline, full attack profiles are generated, and distributed between the parts so that while some amount of the profile is injected to Part A, the remaining is injected to Part B. By this way, an attack profile with 10% filler size is injected to Part A, and the remaining, which is an attack profile with 90% filler size, is injected to Part B. Experimented filler size pairs in the trials for PartA%-PartB% are 3%-97%, 5%-95%, 10%-80%, 15%-75%, 20%-80%, 30%-70%, 40%-60%, 50%-50%, 60%-40%, 70%-30%, 80%-20%, 75%-15%, 80%-10%, 95%-5%, 97%-3%.

As Williams et al. (2007), for training the classifiers, 25 detection attributes are used:

- 6 Generic Attributes: WDMA, RDMA, WDA, LengthVar, DegSim (k = 450), and DegSim' (k = 2, d = 963)
- 6 Average Attack Model-Specific Attributes (3 for push, 3 for nuke): FMV, FMD, ProfileVar
- 4 Random Attack Model-Specific Attributes (2 for push, 2 for nuke): FMD, FAC
- 4 Group Attack Model-Specific Attributes for Bandwagon Attack (2 for push, 2 for nuke): FMD, FAC
- 4 Group Attack Model-Specific Attributes for Segment Attack (2 for push, 2 for nuke): FMTD, GFMV

- 1 Target Detection Model-Specific Attribute: TMF

The empirical values applied by Williams et al. (2007) are used in all the experiments to make the results comparable with the results reported by the authors. Hence, in order to classify unseen profiles with $k$-NN, the detection attributes of the profiles are used to find 9 nearest neighbours in the training set to determine the class using one over Pearson correlation distance weighting. All classifiers and classification results are created using Weka (Witten and Frank, 2005).

In generating the segment attack profiles, which focus on a particular group of items that are similar to each other and likely to be popular among a similar group of users, movies with Harrison Ford as a star (Harrison Ford segment) are used for attack training, whereas for attack testing popular horror movies (Horror segment) are used.

#### 6.4.2.2 *Experimental setup 2: Classification-based attack detection on arbitrarily distributed data with privacy*

In this experiment it is assumed that Part A, and Part B are collaborating both for providing predictions to their customers, and detecting shilling attacks on ADD. For shilling attack detection, parts are employing the distributed version of the classification-based detection algorithm proposed for ADD, as shown in Fig. 6.2.

Before collaboration, each part masks her own data by applying Random Filling. Then, similar to Williams et al. (2007)'s methodology, to obtain separate train and test sets for attack detection and response experiments, one-third of the distributed user profiles are chosen as test profiles, and the other half for train profiles. Distributed user profiles in the first part is used to create training data for the proposed attack detection algorithm. For each test the second part of the data is injected with distributed attack profiles, injected by the proposed design strategy, and then run through the classifier, namely *Model C* in Fig. 6.2.

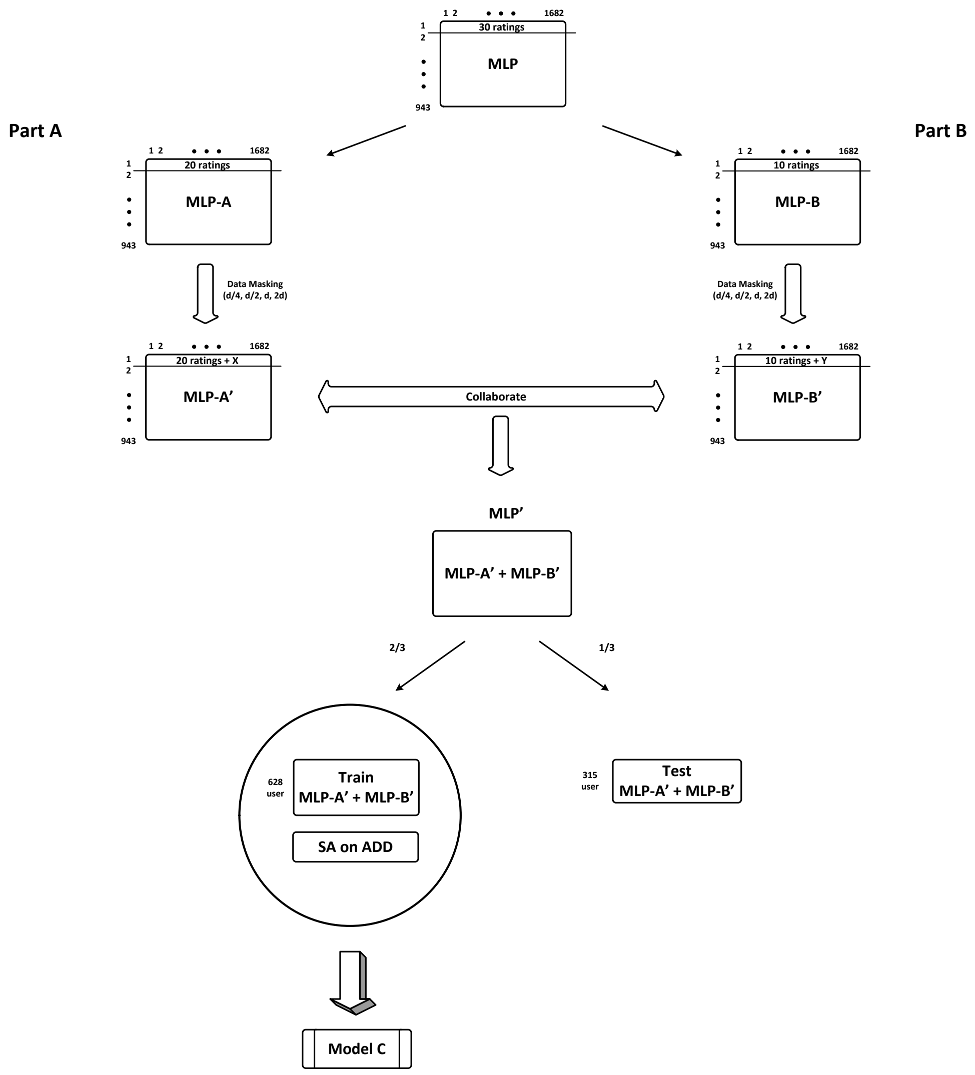


**Figure 6.2.** Experimental Setup 2: Classification-Based Attack Detection on Arbitrarily Distributed Data with Privacy

The detection attributes are generated collaboratively between two parts, by employing the proposed private protocols, and a class attribute, as either "Authentic" or "Attack", is added. For these experiments, all the detection attributes proposed by Williams et al. (2007) is employed, except *DegSim'* and *WDA*, whose calculations might cause privacy issues.

### 6.4.3. Empirical results

Two groups of experiments are conducted to evaluate the classification performance of the classifiers. First group of experiments are aimed to show that parts alone

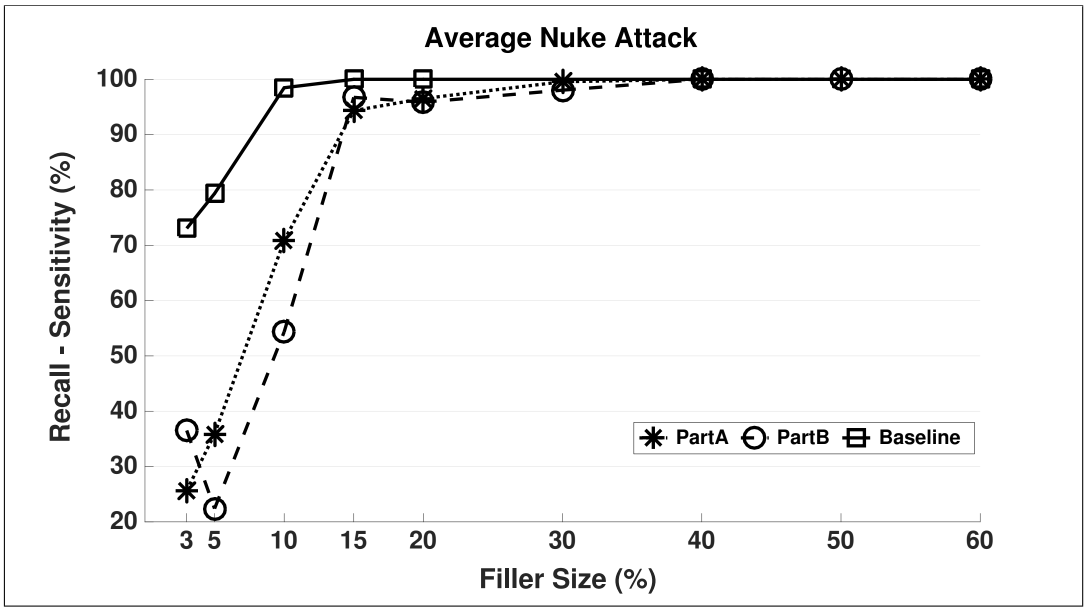


**Figure 6.3.** Recall vs. Filler Size for 1% Average Nuke Attacks on Part A, Part B, and Baseline

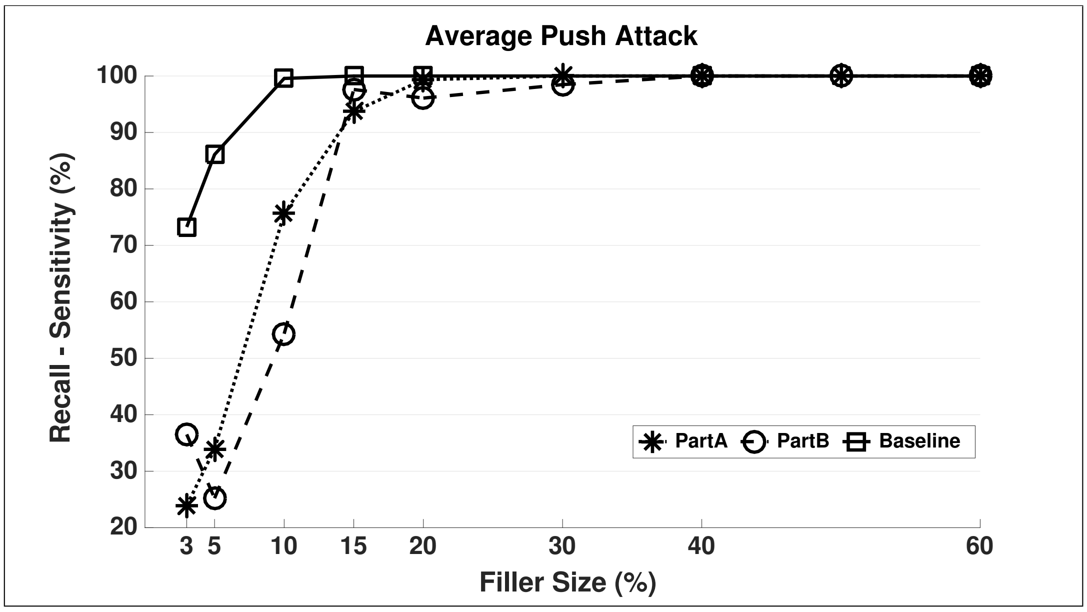


**Figure 6.4.** Recall vs. Filler Size for 1% Average Push Attacks on Part A, Part B, and Baseline

cannot detect distributed attack profiles, and the second group of experiments are demonstrated the performance of the distributed version of the classification-based detection algorithm on ADD.

#### 6.4.3.1 *Experiment 1: Can collaborating parties detect distributed attacks alone?*

By employing one of the existing detection methods for centralized data, which is $k$-NN for this study, each part builds her own classifier model based on her own dataset as explained in Section 6.4.2.1. A malicious user, who knows that two parts are collaborating on ADD to produce referrals, is injecting distributed attack profiles into the system. The question that needs to be answered in the following experiments is that can collaborating parts detect the distributed attack profiles on ADD by themselves. In other word, can the classifiers trained by each part identify the distributed attacks as accurately as the baseline, which is the work proposed by Williams et al. (2007). Their experiments are repeated, and similar results are obtained as the Baseline.

In order to demonstrate performance of the three classifiers, Recall versus Filler Size graphs of the attack models for 1% attack size are examined. Fig. 6.3 and Fig. 6.4 show the classification performance of PartA, Part B, and Baseline for Average Nuke and Average Push attack models, respectively. As can be seen from the figures it is very difficult for Part A and Part B to detect distributed average nuke and push attacks for filler sizes smaller than 15%. While the Recall value of the Baseline algorithm is 72% for average nuke attacks with 3% filler size, Part A and Part B cannot detect distributed average nuke attacks with the classifiers that they build based on their own datasets for this filler size.

Fig. 6.5 and Fig. 6.6 show the classification performance of PartA, Part B, and Baseline for Random Nuke and Random Push attack models, respectively. Even for small filler sizes Baseline classifier detects random nuke and push attacks with

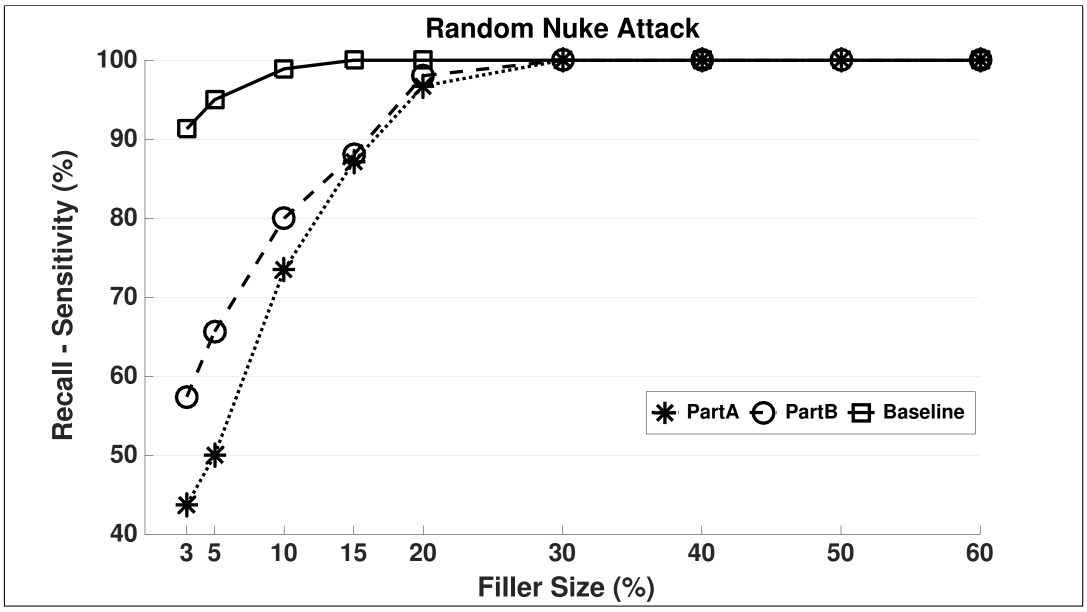


**Figure 6.5.** Recall vs. Filler Size for 1% Random Nuke Attacks on Part A, Part B, and Baseline

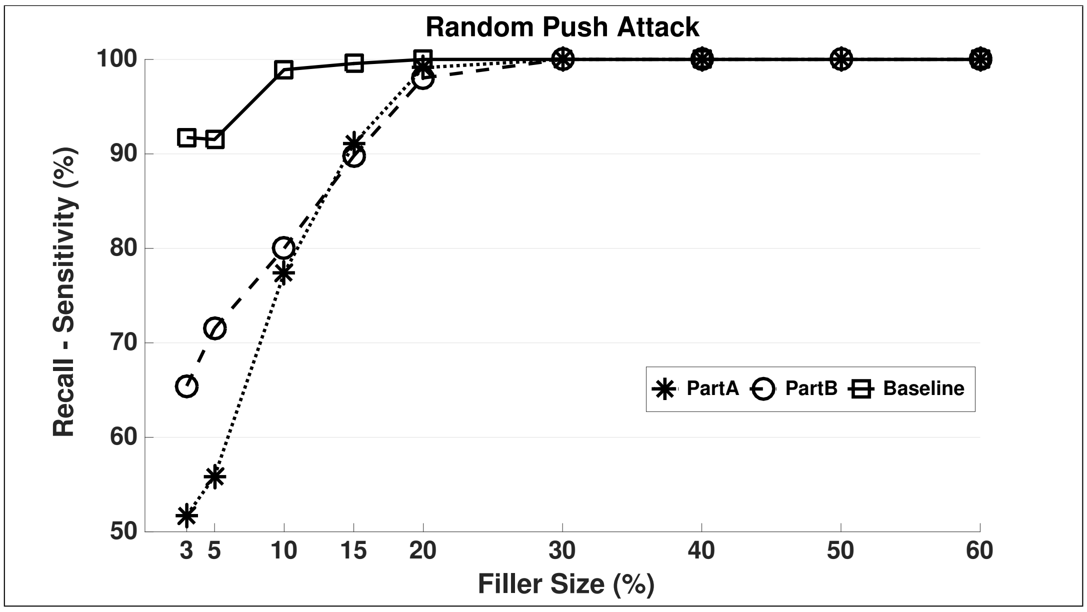


**Figure 6.6.** Recall vs. Filler Size for 1% Random Push Attacks on Part A, Part B, and Baseline

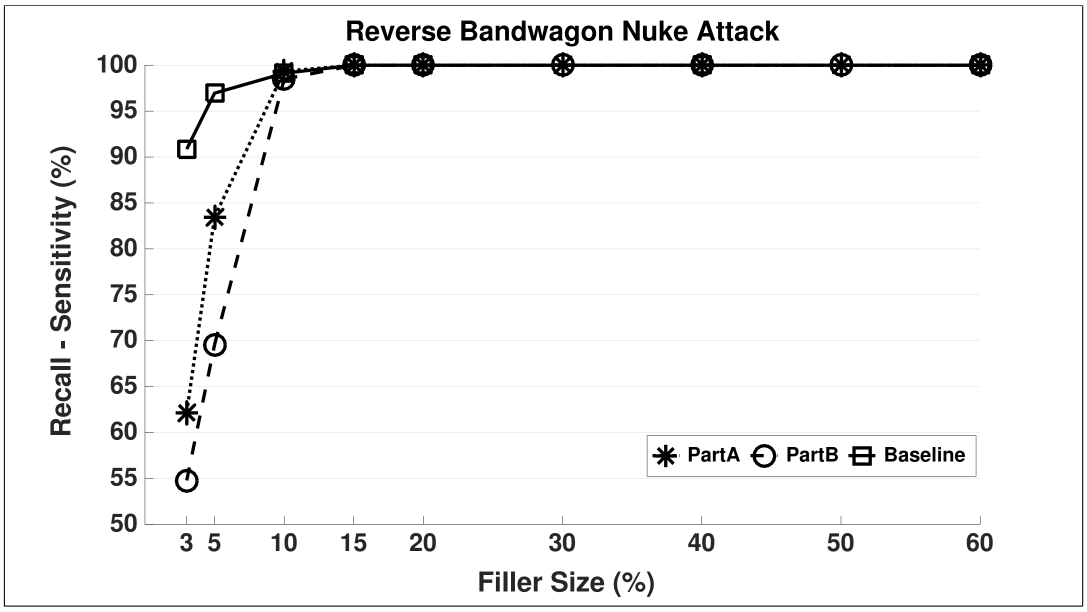


**Figure 6.7.** Recall vs. Filler Size for 1% Reverse Bandwagon Attacks on Part A, Part B, and Baseline

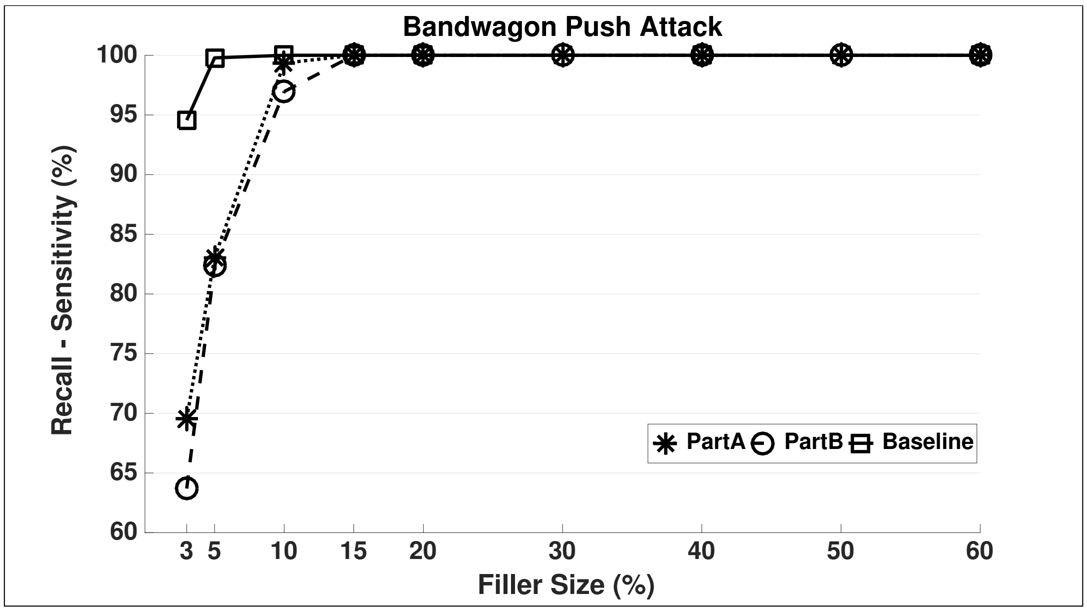


**Figure 6.8.** Recall vs. Filler Size for 1% Bandwagon Attacks on Part A, Part B, and Baseline

90% Recall value. Filler size of 10% is a critical value for random nuke and push attack models for the Baseline. For filler sizes equal to or greater than 10%, Baseline classifiers detect random nuke and push attacks with a 100% Recall. However, this is not the case for Part A and Part B. As can be seen from the figures up to 15% it is difficult for Part A and Part B to detect distributed random nuke and push attacks. Results obtained for Part A and Part B reach approximately 100% Recall when the filler size is equal to or greater than 20%. Up to this filler size, Part A and Part B cannot detect distributed random attack profiles as successful as the Baseline.

Fig. 6.7 and Fig. 6.8 indicate the classification performance of PartA, Part B, and Baseline for Reverse Bandwagon Nuke and Bandwagon Push attack models, respectively. Compared to Average and Random attack models, Part A and Part B achieve better Recall results for small filler size values, when Reverse Bandwagon Nuke and Bandwagon Push attack models are considered. However, these Recall values are below the Baseline results, since even for 3% filler size Baseline reaches 91% Recall. Similar to Baseline results, Part A and Part B reach approximately 100% Recall when the filler size is equal to or greater than 15%.

Classification performance of PartA, Part B, and Baseline for Love/Hate nuke attack model is given in Fig. 6.9. The 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 greater than 10%. When the Recall results obtained for Part A and Part B are examined, it can be stated that while one part, which is Part B in Fig. 6.9, detects distributed Love/Hate attack profiles with a 100% Recall value, which means more successfully than the Baseline for small filler sizes, the other part, which is Part A in the figure, cannot detect them as successfully as Part B and Baseline up to filler size of 10%. Hence, for distributed Love/Hate attacks on ADD, while one part might find the attack profiles, the other part might not find them for same filler size values. This may be due to random distribution of the generated Love/Hate attack profiles between the parts.

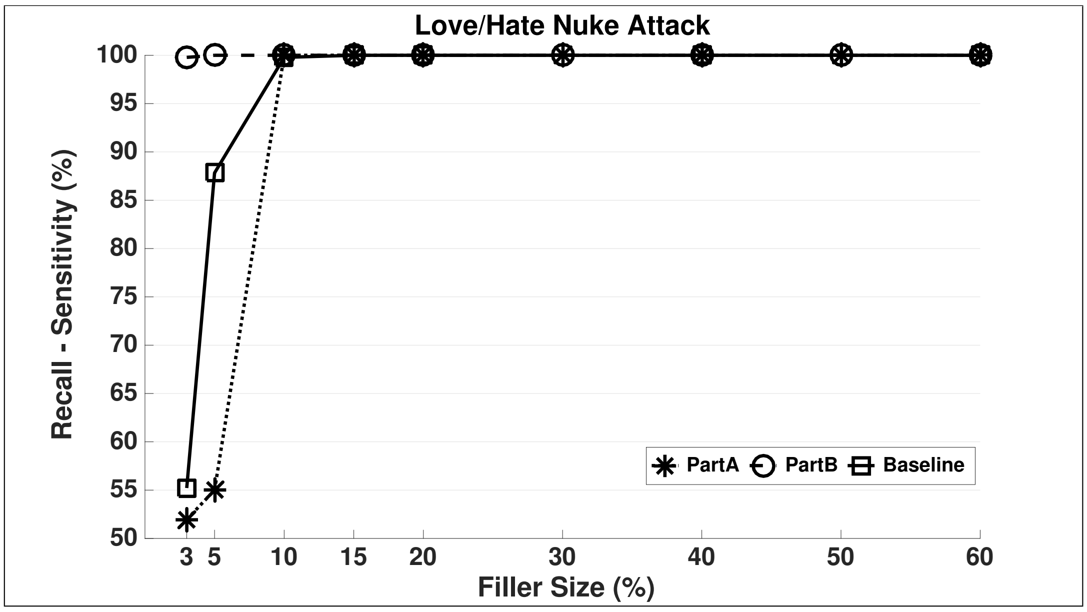


**Figure 6.9.** Recall vs. Filler Size for 1% Love/Hate Attacks on Part A, Part B, and Baseline

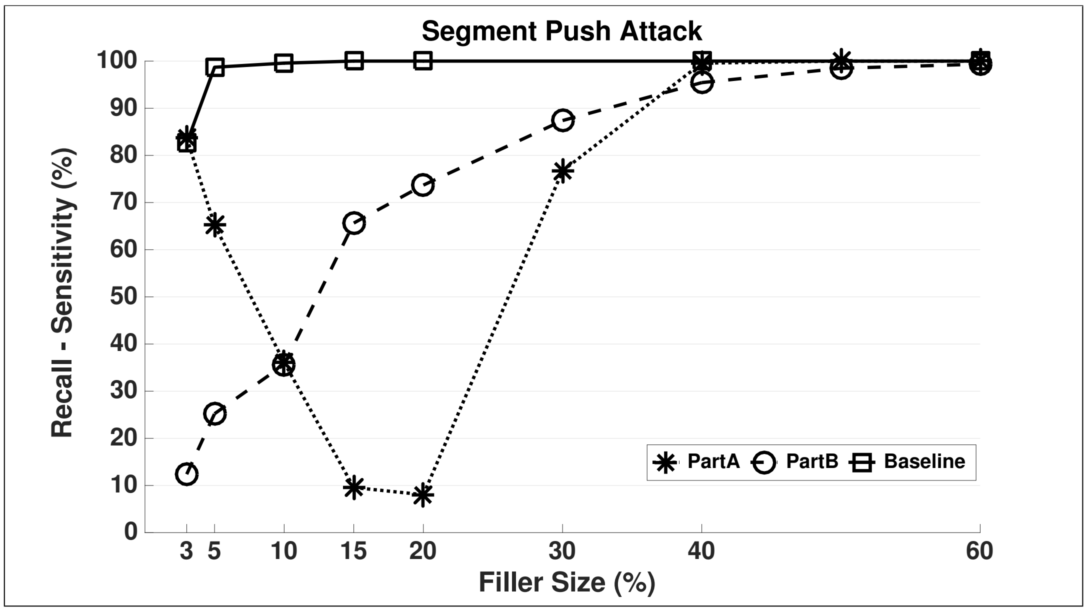


**Figure 6.10.** Recall vs. Filler Size for 1% Segment Attacks on Part A, Part B, and Baseline

Fig. 6.10 shows the classification performance of PartA, Part B, and Baseline for Segment Attack model, which aims to push an item to a targeted group of users with specific interest. As can be seen from the figure, the results obtained for Segment Attack model are significantly different from the Recall results obtained for other attack models. When the 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 filler size values equal to or greater than 5% Recall of 100% is provided. For Part A and Part B to achieve this Recall value the filler size has to be equal or greater than 40%, which is too high for distributed attack profiles. For smaller filler size values, while one part can detect the distributed attack profiles, the other part almost cannot detect the attacks. There are three critical filler size values for Part A and Part B, which are 10%, 20% and 30%. Up to 20% filler size, while Recall of Part A shows a decreasing behaviour, Part B's Recall results are increasing. Between 20% and 30% filler sizes Recall of both Part A and Part B are increasing, and these results reach Baseline classifier when filler size is 40%.

When the above Recall results are compared among each other, results obtained for Part A and Part B are seen as successful as the Baseline for Bandwagon and Love/Hate Attack models. However, Precision is particularly a problem for Part A and Part B classifiers, especially for these attack models. Fig. 6.11 compares Precision results for PartA, Part B, and Baseline for Bandwagon Push Attack on various filler size values. As the figure indicates Precision values obtained for Part A and Part B are far below the Baseline case even for large filler sizes. Hence, many false positive identifications are made by Part A and Part B compared to Baseline, which means that classifiers on Part A and Part B label significant amounts of authentic profiles as an attack. This outcome is also true for Love/Hate Attack model. In Fig. 6.12 Precision results obtained for PartA, Part B, and Baseline forLove/Hate Attack model on different filler size values are shown. Precision results of Part B are far below the Baseline. Even though the Precision values obtained by Part A

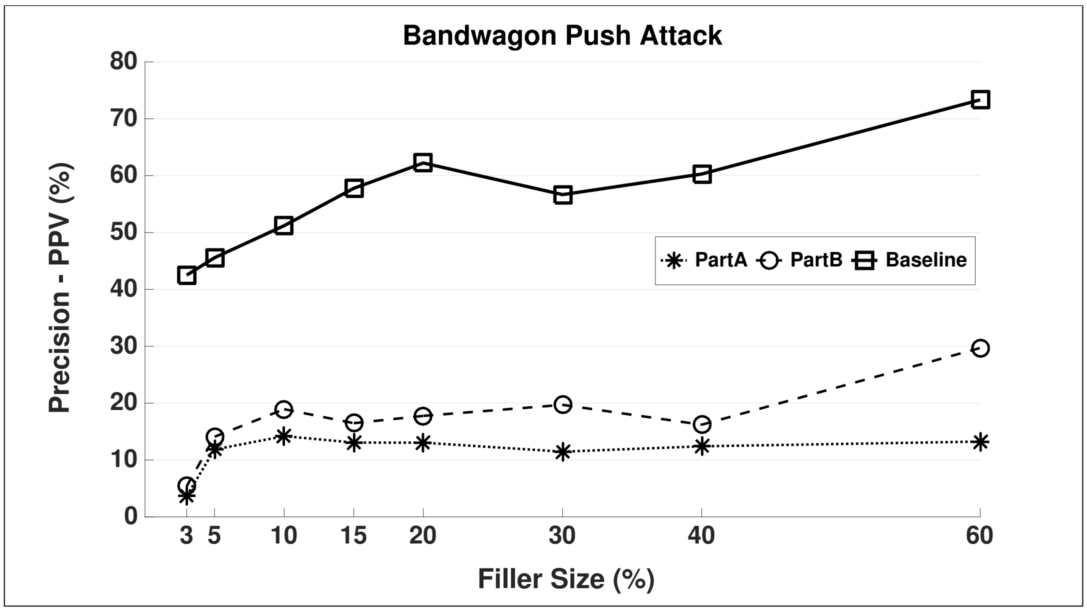


**Figure 6.11.** Precision - PPV vs. Filler Size for 1% Bandwagon Attacks on Part A, Part B, and Baseline

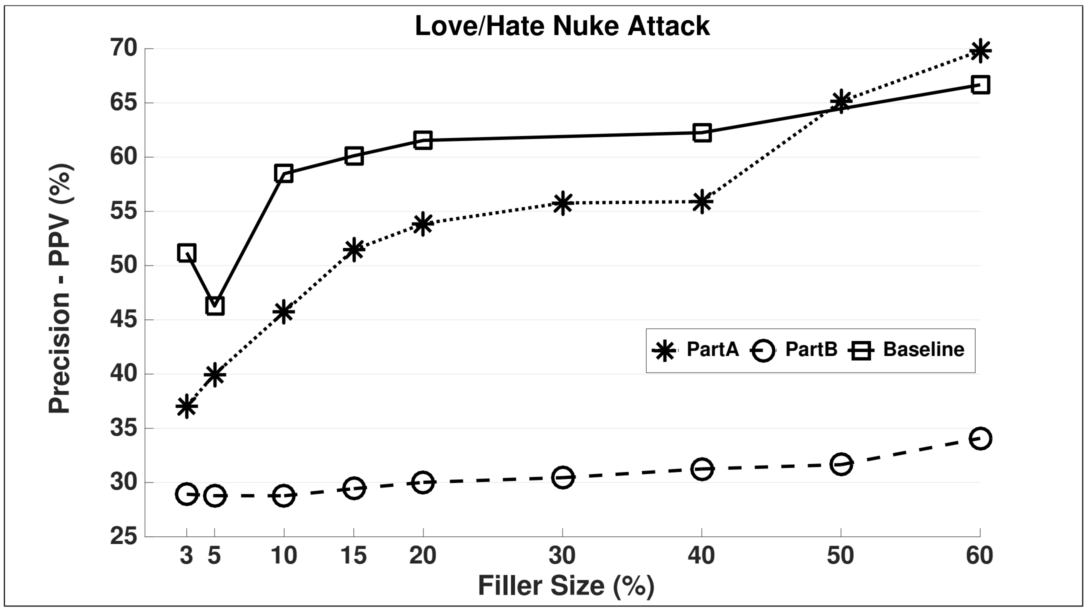


**Figure 6.12.** Precision - PPV vs. Filler Size for 1% Love/Hate Attacks on Part A, Part B, and Baseline

are better than Part B, these results still cannot reach the Baseline. Therefore, Part A and Part B classifiers misclassify some authentic profiles, and label them as an attack.

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 own detection mechanisms, since these classifiers are trained based on partial data, and not on all data, they cannot detect distributed shilling attacks. Therefore, collaboration of parts is necessary in detecting distributed shilling attacks on ADD without jeopardizing privacy.

#### 6.4.3.2 *Experiment 2: Classification-based attack detection on arbitrarily distributed data with privacy*

Since for parts to detect distributed shilling profiles by employing centralized detection methods based on their own dataset is not possible, collaboration of parts for detecting attacks, as well as providing predictions on ADD is studied in the following experiments. For shilling attack detection, the proposed classification-based shilling attack detection method for ADD is employed between Part A and Part B. A variety of experiments are conducted to investigate whether both confidentiality and classification accuracy can be achieved simultaneously by utilizing the proposed classification-based shilling attack detection method for ADD.

For detecting distributed shilling attacks on ADD without revealing privacy, parts need to mask their own data by selectively or uniformly randomly choosing some of their empty cells, and filling them with fake or default ratings. Hence, effects of different methods for determining fake ratings, and the level of perturbation on classification accuracy should be analyzed. Non-personalized ratings, personalized ratings and rating distribution are the three major methods that have been proposed for fake rating determination. Yakut and Polat (2012a) experimentally show that for MLP dataset, user mean and overall mean methods give slightly better results compared to other methods. Similarly, experiments conducted through this study also indicate that among user mean, item mean, overall mean and user rating distribution methods, user mean is the one which gives best results for shilling attack classification. Therefore, user mean method is chosen to determine fake ratings. To capture the balance between privacy and classification accuracy, the number of filled cells is an important matter to be considered. Even though as this number increases privacy of each part also increases, in fact actual data is also corrupted. Thus, increasing this amount definitely affects accuracy (Yakut and Polat, 2012a).

In order to explore the effects of number of unrated cells to be filled on classification accuracy, trials are done using different level of perturbation values, where user mean method is utilized for determining fake ratings. $d/4$, $d/2$, $d$, $2d$, and 0 (which indicates the base case where there is no data masking) are the perturbation values that are assessed, where $d$ represents the density of the user. Fig. 6.13 and Fig.6.14 show Perturbation Amount versus Recall graphs of nuke and push attack models with 5% filler size and 1% attack size, respectively. Looking at Fig. 6.13, it can be seen that for nuke attack models with 5% filler size, adding small amount of noise, which is the case in $d/4$, increases Recall results of the classifier compared to the base results. When the level of perturbation increases to $d/2$, except for the Love/Hate attack models, which is still better than the base case, but slightly worse than the results obtained with $d/4$, Recall of the classifier for all the nuke attack models become better. The negative effect of noise on the classification results are begin to appear slightly after $d/2$, and exactly after $d$. Even though increase in the perturbation amount from $d/2$ to $d$ decreases the classification performance with a certain extent compared to base case, perturbation amount of $d$ still gives acceptable Recall values for nuke attack models with 5% filler size. However, after $d$, increase in perturbation amount completely worsen the Recall results, as seen in $2d$. Similar outcomes can be inference for push attack models with 5% filler size as shown in Fig.6.14. When the perturbation amount is chosen as $d/4$, Recall results of the clas-

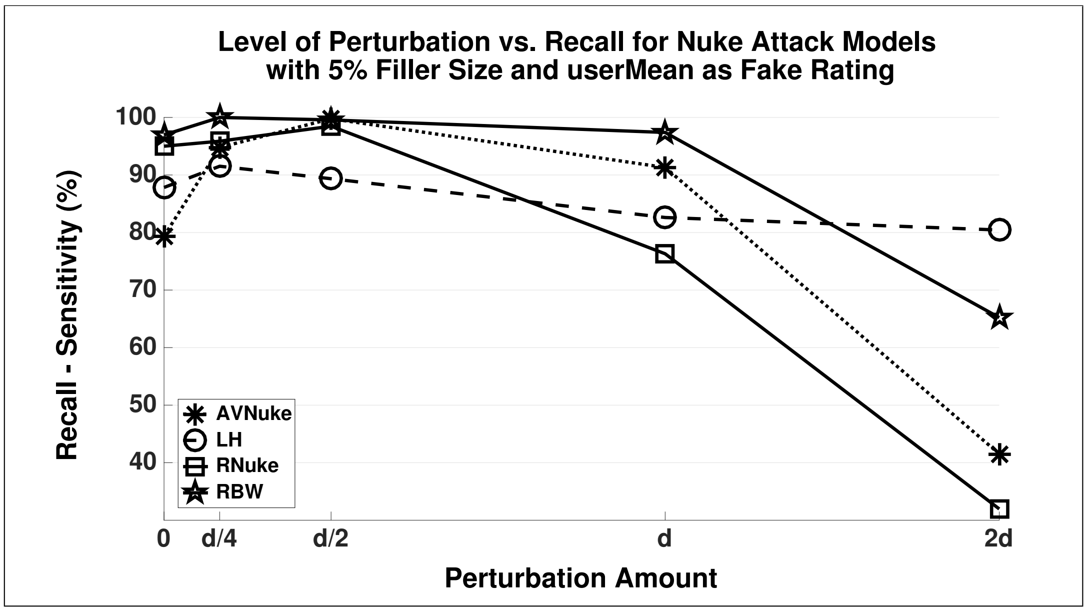


**Figure 6.13.** Perturbation Amount vs. Recall for Nuke Attack Models with 5% Filler Size and User Mean

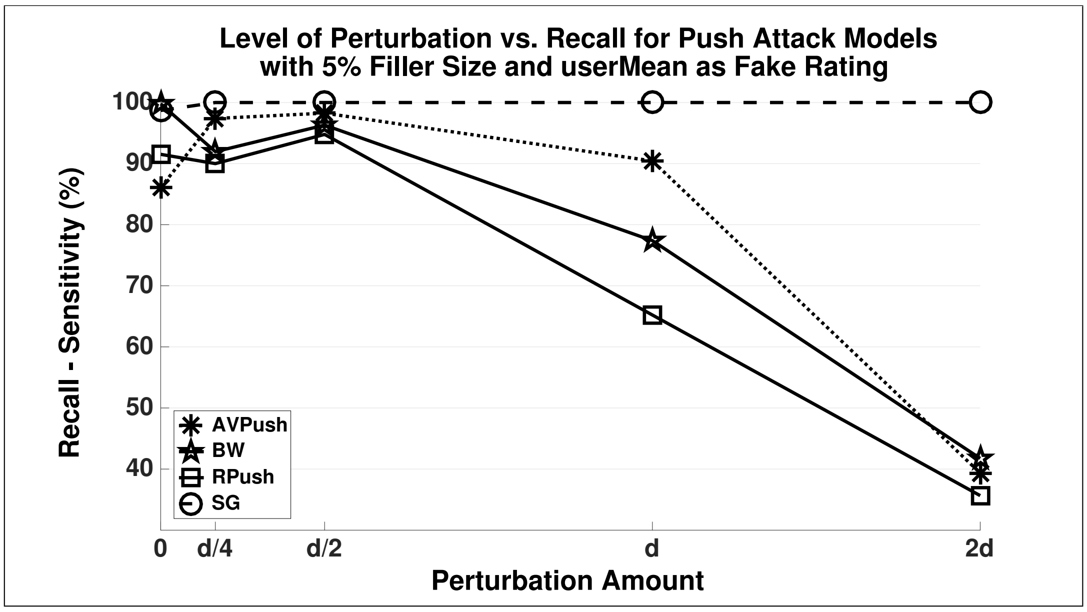


**Figure 6.14.** Perturbation Amount vs. Recall for Push Attack Models with 5% Filler Size and User Mean

sifier for Segment and Average Push attack models become slightly better, where as for Bandwagon and Random Push attack models Recall results become slightly worse compared to base results. For all of the push attack models best Recall results are achieved when the level of perturbation increases to $d/2$. Like the nuke attack models, the negative 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, it can also 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 level of perturbation.

Fig. 6.15 and Fig.6.16 show Perturbation Amount versus Recall graphs of nuke and push attack models with 10% filler size and 1% attack size, respectively. Due to large filler size, all base case Recall results of the classifier for all attack models have 100% as the initial value. Looking at Fig. 6.15, it can be seen that adding small amount of noise slightly decrease the Recall results of the classifier compared to the base results, as the case in $d/4$, yet these results are still accurate for nuke attack models with large filler size. Best Recall results with some noise are obtained when the level of perturbation increases to $d/2$ for nuke attack models. Even though increasing the level of perturbation from $d/2$ to $d$, decreases Recall of the classifier for nuke attack models with 10% filler size, these Recall results are acceptable compared to base case, especially if the privacy level needs to be high. On the other hand, after $d$, increase in perturbation amount worsen the Recall results, as seen in $2d$, especially for Reverse Bandwagon and Average Nuke attack models. Recall versus perturbation amount results of push attack models indicates similar outcomes as the nuke attack models with 10% filler size. As seen in Fig.6.16, initial Recall for all nuke attack models is 100%. Adding some noise slightly decrease the initial results as shown for perturbation amount of $d/4$. While Segment and Average

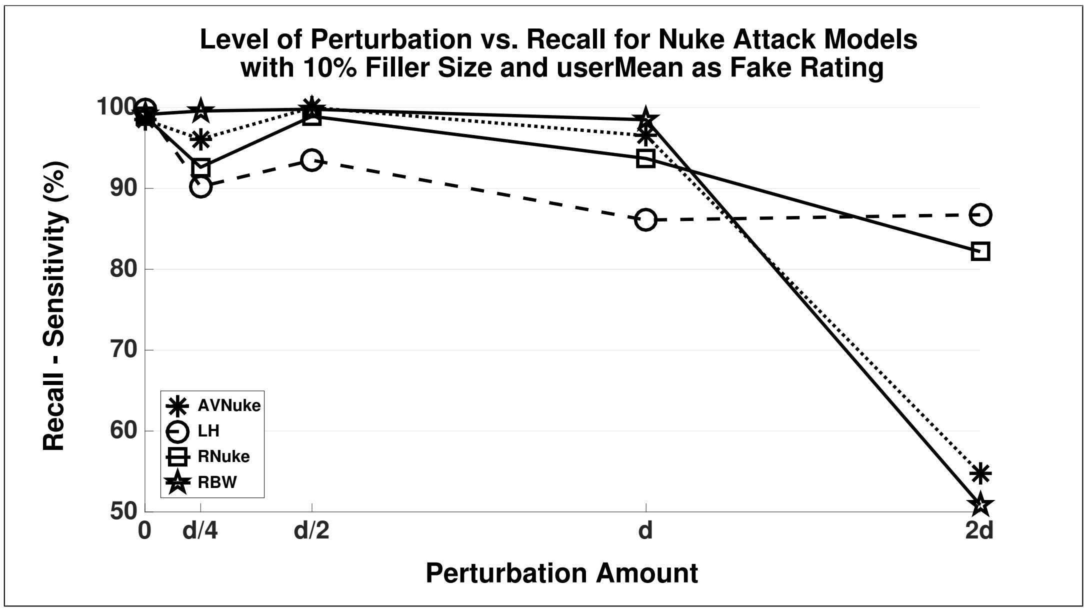


**Figure 6.15.** Perturbation Amount vs. Recall for Nuke Attack Models with 10% Filler Size and User Mean

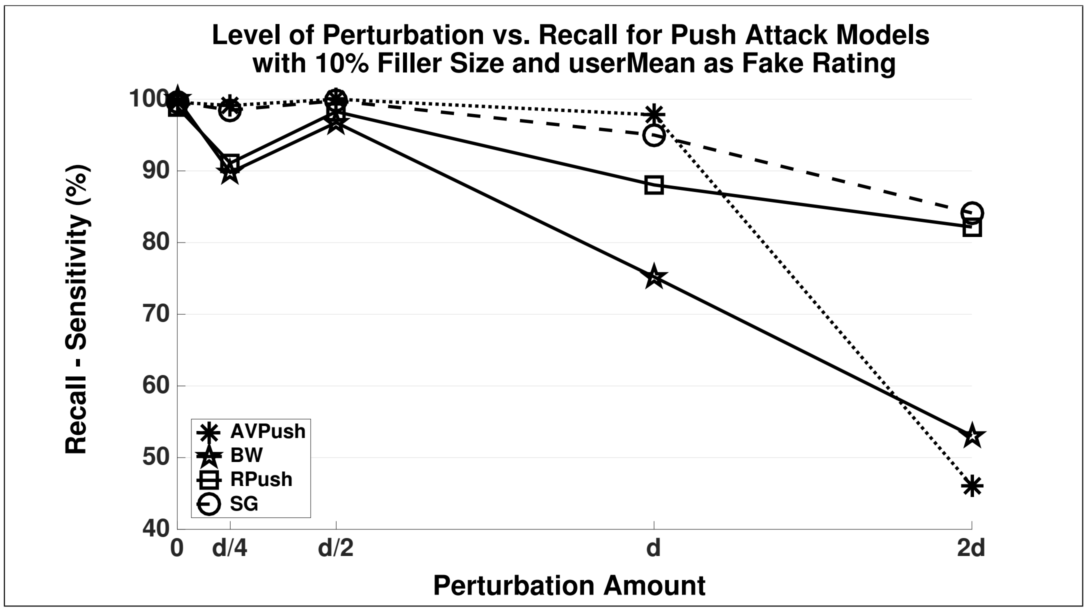


**Figure 6.16.** Perturbation Amount vs. Recall for Push Attack Models with 10% Filler Size and User Mean

Push attack models are not effecting by this noise, Bandwagon and Random Push attacks are slightly effecting. As the case for nuke attack models, when the level of perturbation set to $d/2$, best Recall results for nuke attack models are obtained, which are very close to base results. The negative 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 $2d$ indicates, increase in perturbation amount worsen 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 level of perturbation, which is identical to the results obtained for attack models with 5% filler size.

These empirical results demonstrate that equilibrium between privacy and classification accuracy can be assured for well-known shilling attack models with both small and large filler sizes. Besides by utilizing the proposed classification-based attack detection method for ADD, it is possible to detect distributed shilling attacks on ADD effectively, without jeopardizing data owners's privacy.

## 6.5. Conclusions

In this chapter, in order to protect privacy preserving distributed collaborative filtering algorithms against shilling attacks, and to keep up collaboration of online vendors on arbitrary data, distributed version of a classification based attack detection method, which can operate on arbitrarily distributed data while preserving privacy, 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 confidentiality of data holders. By operating the proposed private attribute estimation protocols consecutively, classification attributes are derived off-line among the parts. At the end, half of the derived attributes are known by each part. Then, by exchanging the attributes the classification model is constructed with k-NN algorithm. At this point, even though both parts have the model, 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 effectively, without jeopardizing data owners's privacy. Moreover, it is also experimentally shown that even if collaborative parts have their own detection mechanisms working on their own sides based on their own data, distributed attacks, which are injected by the proposed attack generation strategy for arbitrarily distributed data, cannot be detected. Hence, for detecting shilling profiles on arbitrarily distributed data, collaboration of parts is required, and with experimental studies the need for collaboration in detection of distributed shilling attacks on arbitrarily distributed data is exposed.

# 7. CONCLUDING REMARKS

This dissertation has focused on shilling attacks against privacy-preserving distributed collaborative filtering algorithms. In this final chapter, results attained in the work are summarized, and some directions in which this research might be extended in the future are highlighted.

## 7.1. Conclusions

In this dissertation, privacy preserving distributed collaborative filtering algorithms proposed on arbitrarily distributed data are investigated in terms of robustness against shilling attacks. Attack strategies developed so far are designed for central server-based systems, hence cannot be directly employed for arbitrarily distributed data-based recommender systems. For such recommender systems, data distribution is an another weakness which must be considered by an attack designer. Therefore, in order to manipulate the outcomes of privacy preserving distributed collaborative filtering algorithms proposed on arbitrarily distributed data, a new attack strategy, which injects fake profiles into these systems by partitioning the profile between data holders, is developed. Introduced attack strategy is used in generation of distributed adaptations of formerly proposed six-well known shilling attack models for cases of both numeric and binary data. Real data-based experiments confirm that attacks generated by the proposed strategy are capable of biasing the outcomes of the state-of-the-art privacy preserving distributed collaborative filtering schemes on arbitrarily distributed data, hence possibility of mounting successful attacks against privacy preserving distributed collaborative filtering algorithms proposed on arbitrarily distributed data are shown. Empirical outcomes also proven the vulnerability of these algorithms against shilling attacks.

Robustness analyzes of preserving distributed collaborative filtering algorithms proposed on arbitrarily distributed data, are also revealed the need for defending mechanisms against attacks. Even though these algorithms are beneficial for online vendors to offer more accurate and reliable collaborative filtering services without jeopardizing privacy, being subject to shilling attacks run the risk of turning these systems into promotion tools for malicious users, vendors, or retailers. Hence, for online vendors to continue to collaborate on distributed data, detection of shilling attacks is essential.

In literature there are several solutions proposed for detection of shilling attacks; however existing solutions are for the case where data is collected in one center. In other words, existing attack detection methods work on centralized data, and whole detection process is controlled by a single data holder, who knows all the statistics of the data. On the other hand, when data is arbitrarily distributed between two parties, due to privacy concerns, parts cannot have the control of all data, but can only operate on their own parts. Because of these reasons, existing attack detection methods cannot be directly employed on distributed data. New attack detection algorithms, or distributed versions of the existing attack detection solutions, which can be employed on arbitrarily distributed data with privacy, are required.

To protect privacy preserving distributed collaborative filtering algorithms against shilling attacks, and to keep up collaboration of online vendors, distributed version of a well known classification based attack detection method which can operate on arbitrarily distributed data while preserving privacy, is presented. In order to derive the required generic and model-specific classification attributes for each distributed profile collaboratively between two parties, private protocols are developed. Homomorphic encryption, and random filling techniques are used in achieving confidentiality of the parties. By employing the proposed private protocols off-line, half of the derived attributes are known by each part. Then according to the classifier learning algorithm that is decided, which is k-nn for this dissertation, the classifier model is constructed by exchanging the attributes. In this point, even though both parts learn the model, in order to test and classify a new instance, collaboration between parties is essential. 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 effectively, without jeopardizing data owners' privacy.

Moreover, it is also experimentally shown that even if collaborative parts have their own detection mechanisms working on their own sides based on their own data, distributed attacks, which are injected by the proposed attack generation strategy for arbitrarily distributed data, cannot be detected. More specifically, if a malicious user, who knows the collaboration of two parts, injects distributed attack profiles into the system, it is not easy for parts to identify these attack profiles by themselves, even so, parts have their own detection mechanisms based on one of the existing attack detection methods. Hence, for detecting shilling profiles on arbitrarily distributed data, collaboration of parts is required, and with experimental studies the need for collaboration in detection of distributed shilling attacks on arbitrarily distributed data is exposed.

## 7.2. Directions for Future Research

In this dissertation, robustness analyzes of binary ratings based privacy preserving distributed collaborative filtering algorithm proposed on arbitrarily distributed data is also shown to be subject to shilling attacks with experimental studies. To keep collaboration of the online vendors on arbitrarily distributed binary data, new detection methods are required. In fact, almost there is no binary shilling attack detection algorithm in literature (Batmaz, 2015). Hence, shilling attack detection methods on binary data both for central server-based systems, and distributed systems are need to be investigated.

There are numerous shilling attack detection algorithms proposed in 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 dissertation can be utilized in deriving the required attributes.

In this dissertation, it is assumed that there are no overlapping ratings. However, in real life scenarios, overlapping ratings are inevitable, therefore effects of overlapping ratings can be studied.

In order to enhance cryptographic techniques-based solutions, new secure-multi party computation protocols might be suggested.

Developing robust privacy preserving distributed collaborative filtering algorithms for arbitrarily distributed data, which are intrinsically resistant to attacks, might also be an interesting research topic.

Besides arbitrarily distribution, data might be distributed horizontally or vertically between multiple parties, and there are several privacy preserving schemes proposed for these cases. Robustness analyzes of the proposed privacy preserving collaborative filtering schemes for vertically and horizontally distributed data among multiple parties against shilling attacks are need to be studies. In order to analyze the robustness of these systems, the proposed attack strategy might be adapted, or new attack strategies can be proposed for vertically and horizontally distributed data. Furthermore, if these systems are also vulnerable to attacks, detection algorithms, which can operate on these cases of data distributions with privacy, might be investigated.

# REFERENCES

Adomavicius, G., and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17(6), 734–749.

Aggarwal, C.C. (2016a). Attack-resistant recommender systems, *Recommender Systems*. Springer, pp. 385–410.

Aggarwal, C.C. (2016b). An introduction to recommender systems, *Recommender Systems*. Springer, pp. 1–28.

Basu, A., Kikuchi, H., and Vaidya, J. (2011a). Privacy-preserving weighted Slope One predictor for item-based collaborative filtering, *Proceedings of the International Workshop on Trust and Privacy in Distributed Information Processing (workshop at the IFIPTM'11)*, Copenhagen, Denmark.

Basu, A., Vaidya, J., Dimitrakos, T., and Kikuchi, H. (2012a). Feasibility of a privacy preserving collaborative filtering scheme on the Google App Engine: A performance case study, *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC'12)*, Trento, Italy. pp. 447–452.

Basu, A., Vaidya, J., and Kikuchi, H. (2011b). Efficient privacy-preserving collaborative filtering based on the weighted Slope One predictor. *Journal of Internet Services and Information Security (JISIS)* 1(4), 26–46.

Basu, A., Vaidya, J., and Kikuchi, H. (2012b). Perturbation based privacy preserving Slope One predictors for collaborative filtering, *Proceedings of the 6th IFIP International Conference on Trust Management (IFIPTM'12)*, Surat, India. pp. 17–35.

Basu, A., Vaidya, J., Kikuchi, H., and Dimitrakos, T. (2011c). Privacy-preserving collaborative filtering for the cloud, *Proceedings of the 3rd International Conference on Cloud Computing Technology and Science (CloudCom'11)*, Divani Caravel, Athens, Greece. pp. 223–230.

Basu, A., Vaidya, J., Kikuchi, H., and Dimitrakos, T. (2013). Privacy-preserving collaborative filtering on the cloud and practical implementation experiences, *Proceedings of the 6th EEE 6th International Conference on Cloud Computing (CLOUD'13)*, Santa Clara Marriott, CA, USA. pp. 406–413.

Basu, A., Vaidya, J., Kikuchi, H., Dimitrakos, T., and Nair, S.K. (2012c). Privacy preserving collaborative filtering for SaaS enabling PaaS clouds. *Journal of Cloud Computing* 1(1), 1–14.

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation, *Proceedings of the 15th National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI'98/IAAI'98)*, Madison, Wisconsin, USA. pp. 714–720.

Batmaz, Z. (2015). Shilling attack design and detection on masked binary data. Master's thesis. Anadolu University. Graduate School of Science, Computer Engineering Program, Eskişehir, Türkiye.

Berkovsky, S., and Freyne, J. (2015). Web personalization and recommender systems, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*, Sydney, NSW, Australia. pp. 2307–2308.

Bhaumik, R., Mobasher, B., and Burke, R.D. (2011). A clustering approach to unsupervised attack detection in collaborative recommender systems, *Proceedings of the 7th IEEE International Conference on Data Mining*, Las Vegas, NV, USA. pp. 181–187.

Bhaumik, R., Williams, C., Mobasher, B., and Burke, R. (2006). Securing collaborative filtering against malicious attacks through anomaly detection, *Proceedings of the 4th Workshop on Intelligent Techniques for Web Personalization (ITWP'06), held at AAAI 2006*, Boston, Massachusetts.

Bilge, A., Güneş, I., and Polat, H. (2014a). Robustness analysis of privacy-preserving model-based recommendation schemes. *Expert Systems with Applications* 41(8), 3671–3681.

Bilge, A., Kaleli, C., Yakut, I., Güneş, I., and Polat, H. (2013). A survey of privacy-preserving collaborative filtering schemes. *International Journal of Software Engineering and Knowledge Engineering* 23(08), 1085–1108.

Bilge, A., Özdemir, Z., and Polat, H. (2014b). A novel shilling attack detection method. *Procedia Computer Science* 31, 165–174.

Bilge, A., and Polat, H. (2011). An improved profile-based CF scheme with privacy, *Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC'11)*, San Francisco, USA. pp. 133–140.

Bilge, A., and Polat, H. (2012). An improved privacy-preserving DWT-based collaborative filtering scheme. *Expert Systems with Applications* 39(3), 3841–3854.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems* 46, 109–132.

Breese, J.S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI. pp. 43–52.

Bryan, K., O'Mahony, M., and Cunningham, P. (2008). Unsupervised retrieval of attack profiles in collaborative recommender systems, *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08)*, Lausanne, Switzerland. pp. 155–162.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12(4), 331–370.

Burke, R., Mobasher, B., and Bhaumik, R. (2005a). Limited knowledge shilling attacks in collaborative filtering systems, *Proceedings of 3rd International Workshop on Intelligent Techniques for Web Personalization (ITWP'05), held at 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edinburgh, Scotland. pp. 17–24.

Burke, R., Mobasher, B., Bhaumik, R., and Williams, C. (2005b). Collaborative recommendation vulnerability to focused bias injection attacks, *Proceedings of the Workshop on Privacy and Security Aspects of Data Mining, held at ICDM'05*, Houston, Texas.

Burke, R., Mobasher, B., Bhaumik, R., and Williams, C. (2005c). Segment-based injection attacks against collaborative filtering recommender systems, *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, Washington, DC, USA. pp. 577–580.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. (2006a). Classification features for attack detection in collaborative recommender systems, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, Philadelphia, PA, USA. pp. 542–547.

Burke, R., Mobasher, B., Williams, C., and Bhaumik, R. (2006b). Detecting profile injection attacks in collaborative recommender systems, *Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, San Francisco, California. pp. 23–23.

Burke, R., Mobasher, B., Zabicki, R., and Bhaumik, R. (2005d). Identifying attack models for secure recommendation, *Proceedings of the Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, San Diego, California. pp. 19–25.

Burke, R., O'Mahony, M.P., and Hurley, N.J. (2015). Robust collaborative recommendation, *Recommender Systems Handbook*. Springer, pp. 961–995.

Canny, J. (2002). Collaborative filtering with privacy, *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Oakland, California, USA. pp. 45–57.

Cao, J., Wu, Z., Mao, B., and Zhang, Y. (2013). Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system. *World Wide Web* 16(5-6), 729–748.

Chakraborty, P., and Karforma, S. (2013). Detection of profile-injection attacks in recommender systems using outlier analysis. *Procedia Technology* 10, 963–969.

Cheng, Z., and Hurley, N. (2009). Effective diverse and obfuscated attacks on model-based recommender systems, *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, New York, NY, USA. pp. 141–148.

Cheng, Z., and Hurley, N. (2010). Robust collaborative recommendation by least trimmed squares matrix factorization, *Proceedings of the 22nd IEEE International*

*Conference on Tools with Artificial Intelligence (ICTAI'10)*, Arras, France. pp. 105–112.

Chirita, P.A., Nejdl, W., and Zamfir, C. (2005). Preventing shilling attacks in online recommender systems, *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management (WIDM'05)*, Bremen, Germany. pp. 67–74.

Chung, C.Y., Hsu, P.Y., and Huang, S.H. (2013). $\beta$P: A novel approach to filter out malicious rating profiles from recommender systems. *Decision Support Systems* 55(1), 314–325.

Dahl, M., Ning, C., and Toft, T. (2012). On secure two-party integer division, *Proceedings of the 16th International Conference on Financial Cryptography and Data Security (FC'12)*, Kralendijk, Bonaire. pp. 164–178.

Damgård, I., and Jurik, M. (2001). A generalisation, a simplification and some applications of Paillier's probabilistic public-key system, *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC'01)*, Cheju Island, Korea. pp. 119–136.

Dokoohaki, N., Kaleli, C., Polat, H., and Matskin, M. (2010). Achieving optimal privacy in trust-aware social recommender systems, *Proceedings of the 2nd International Conference on Social Informatics (SocInfo'10)*, Laxenburg, Austria. pp. 62–79.

Ekstrand, M.D., Riedl, J.T., and Konstan, J.A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction* 4(2), 81–173.

Fug-uo, Z., and Sheng-hua, X. (2007). Analysis of trust-based e-commerce recommender systems under recommendation attacks, *Proceedings of the 1st International Symposium on Data, Privacy, and E-Commerce (ISDPE'07)*, Chengdu, Sichuan, China. pp. 385–390.

Gao, M., Tian, R., Wen, J., Xiong, Q., Ling, B., and Yang, L. (2015). Item anomaly detection based on dynamic partition for time series in recommender systems. *PloS one* 10(8).

Gao, M., Yuan, Q., Ling, B., and Xiong, Q. (2014). Detection of abnormal item based on time intervals for recommender systems. *The Scientific World Journal* 2014.

Goldberg, D., Nichols, D., Oki, B.M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12), 61–70.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval* 4(2), 133–151.

Gong, S. (2011). Privacy-preserving collaborative filtering based on randomized perturbation techniques and secure multiparty computation. *International Journal of Advancements in Computing Technology* 3(4), 89–99.

Güneş, I., Bilge, A., Kaleli, C., and Polat, H. (2013a). Shilling attacks against privacy-preserving collaborative filtering. *Journal of Advanced Management Science* 1(1), 54–60.

Güneş, I., Bilge, A., and Polat, H. (2013b). Shilling attacks against memory-based privacy-preserving recommendation algorithms. *KSII Transactions on Internet and Information Systems (TIIS)* 7(5), 1272–1290.

Güneş, I., Kaleli, C., Bilge, A., and Polat, H. (2014). Shilling attacks against recommender systems: A comprehensive survey. *Artificial Intelligence Review* 42(4), 767–799.

Güneş, I., and Polat, H. (2015). Robustness analysis of privacy-preserving hybrid recommendation algorithm. *International Journal of Information Security Science* 4(1), 13–25.

He, F., Wang, X., and Liu, B. (2010). Attack detection by rough set theory in recommendation system, *Proceedings of the 2010 IEEE International Conference on Granular Computing (GrC-2010)*, San Jose, CA, USA. pp. 692–695.

Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., and Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* 1(Oct), 49–75.

Herlocker, J., Konstan, J.A., and Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval* 5(4), 287–310.

Herlocker, J.L., Konstan, J.A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, USA. pp. 230–237.

Hsieh, C.L. (2011). Toward better recommender system by collaborative computation with privacy preserved, *Proceedings of the 11th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT'11)*, Munich, Germany. pp. 246–249.

Hsieh, C.L.A., Zhan, J., Zeng, D., and Wang, F. (2008). Preserving privacy in joining recommender systems, *Proceedings of the 2nd International Conference on Information Security and Assurance (ISA'08)*, Busan, Korea. pp. 561–566.

Huang, S., Shang, M., and Cai, S. (2012). A hybrid decision approach to detect profile injection attacks in collaborative recommender systems, *Foundations of Intelligent Systems*. Springer, pp. 377–386.

Hurley, N., Cheng, Z., and Zhang, M. (2009). Statistical attack detection, *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, New York, NY, USA. pp. 149–156.

Jagannathan, G., and Wright, R.N. (2005). Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD'05)*, Chicago, IL, USA. pp. 593–599.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). Recommender systems: An introduction. Cambridge University Press.

Jeckmans, A., Tang, Q., and Hartel, P. (2012). Privacy-preserving collaborative filtering based on horizontally partitioned dataset, *Proceedings of the International Conference on Collaboration Technologies and Systems (CTS'12)*, Denver, Colorado, USA. pp. 439–446.

Ji, A.T., Yeon, C., Kim, H.N., and Jo, G.S. (2007). Distributed collaborative filtering for robust recommendations against shilling attacks, *Advances in Artificial Intelligence*. Springer, pp. 14–25.

Kaleli, C., and Polat, H. (2007a). Providing Naïve Bayesian classifier-based private recommendations on partitioned data, *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07)*, Warsaw, Poland. pp. 515–522.

Kaleli, C., and Polat, H. (2007b). Providing private recommendations using Naïve Bayesian classifier, *Advances in Intelligent Web Mastering*. Springer, pp. 168–173.

Kaleli, C., and Polat, H. (2010). P2P collaborative filtering with privacy. *Turkish Journal of Electrical Engineering & Computer Sciences* 18(1), 101–116.

Kaleli, C., and Polat, H. (2011). Privacy-preserving trust-based recommendations on vertically distributed data, *Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC'11)*, Stanford University, Palo Alto, CA, USA. pp. 376–379.

Kaleli, C., and Polat, H. (2012a). Privacy-preserving SOM-based recommendations on horizontally distributed data. *Knowledge-Based Systems* 33, 124–135.

Kaleli, C., and Polat, H. (2012b). SOM-based recommendations with privacy on multi-party vertically distributed data. *Operational Research Society* 63(4), 826–838.

Kaleli, C., and Polat, H. (2013). Robustness analysis of Naïve Bayesian classifier-based collaborative filtering, *E-Commerce and Web Technologies*. Springer, pp. 202–209.

Kaleli, C., and Polat, H. (2015). Privacy-preserving Naïve Bayesian classifier–based recommendations on distributed data. *Computational Intelligence* 31(1), 47–68.

Kim, L. (2015). 15 Mind-Blowing Statistics Reveal What Happens on the Internet in a Minute [Infographic]. `http://www.inc.com/larry-kim/15-mind-blowing-statistics-reveal-what-happens-on-the-internet-in-a-minute.html`. [Online; accessed 27-January-2016].

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., and Riedl, J. (1997). GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM* 40(3), 77–87.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, Las Vegas, NV, USA. pp. 426–434.

Lam, S.K., and Riedl, J. (2004). Shilling recommender systems for fun and profit, *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*, New York, NY, USA. pp. 393–402.

Lam, S.K., and Riedl, J. (2005). Privacy, shilling, and the value of information in recommender systems, *Proceedings of User Modeling Workshop on Privacy-Enhanced Personalization*, pp. 85–92.

Lee, J.S., and Zhu, D. (2012). Shilling attack detection - a new approach for a trustworthy recommender system. *INFORMS Journal on Computing* 24(1), 117–131.

Li, C., and Luo, Z. (2011). Detection of shilling attacks in collaborative filtering recommender systems, *Proceedings of the 3rd International Conference of Soft Computing and Pattern Recognition (SoCPaR'11)*, Dalian, China. pp. 190–193.

Lindell, Y., and Pinkas, B. (2009). Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality* 1(1), 5.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7(1), 76–80.

Long, Q., and Hu, Q. (2010). Robust evaluation of binary collaborative recommendation under profile injection attack, *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing (PIC'10)*, Shanghai, China. pp. 1246–1250.

Mehta, B. (2007). Unsupervised shilling detection for collaborative filtering, *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI'07)*, Vancouver, British Columbia. pp. 1402–1407.

Mehta, B., and Hofmann, T. (2008). A survey of attack-resistant collaborative filtering algorithms. *IEEE Data Engineering Bulletin* 31(2), 14–22.

Mehta, B., Hofmann, T., and Fankhauser, P. (2007a). Lies and propaganda: detecting spam users in collaborative filtering, *Proceedings of the 12th International Conference on Intelligent User Interfaces, (IUI'07)*, Honolulu, HI, USA. pp. 14–21.

Mehta, B., Hofmann, T., and Nejdl, W. (2007b). Robust collaborative filtering, *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07)*, Minneapolis, MN, USA. pp. 49–56.

Mehta, B., and Nejdl, W. (2008). Attack resistant collaborative filtering, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*, Singapore, Singapore. pp. 75–82.

Mehta, B., and Nejdl, W. (2009). Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User-Adapted Interaction* 19(1-2), 65–97.

Melville, P., Mooney, R.J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations, *Proceedings of the 14th Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence collocated with the 18th National Conference on Artificial Intelligence (AAAI'02/IAAI'02)*, Edmonton, Alberta, Canada. pp. 187–192.

Memiş, B., and Yakut, I. (2014). Privacy-preserving two-party collaborative filtering on overlapped ratings. *TIIS* 8(8), 2948–2966.

Mobasher, B., Burke, R., Bhaumik, R., and Sandvig, J.J. (2007a). Attacks and remedies in collaborative recommendation. *EEE Intelligent Systems* 22(3), 56–63.

Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. (2005). Effective attack models for shilling item-based collaborative filtering systems, *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD'05*, Chicago, IL, USA. pp. 13–23.

Mobasher, B., Burke, R., Bhaumik, R., and Williams, C. (2007b). Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology (TOIT)* 7(4), 23–60.

Mobasher, B., Burke, R., and Sandvig, J.J. (2006a). Model-based collaborative filtering as a defense against profile injection attacks, *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, Boston, Massachusetts. p. 1388.

Mobasher, B., Burke, R., Williams, C., and Bhaumik, R. (2006b). Analysis and detection of Segment-focused attacks against collaborative recommendation, *Proceedings of the 7th International Conference on Knowledge Discovery on the Web: Advances in Web Mining and Web Usage Analysis (WebKDD'05)*, Chicago, IL, USA. pp. 96–118.

Morid, M.A., Shajari, M., and Hashemi, A.R. (2014). Defending recommender systems by influence analysis. *Information Retrieval* 17(2), 137–152.

Ning, X., Desrosiers, C., and Karypis, G. (2015). A comprehensive survey of neighborhood-based recommendation methods, *Recommender Systems Handbook.* Springer, pp. 37–76.

O'Connor, M., and Herlocker, J. (1999). Clustering items for collaborative filtering, *Proceedings of the ACM SIGIR Workshop on Recommender Systems (SIGIR'99)*, Berkeley, California, USA.

O'Donovan, J., and Smyth, B. (2006). Is trust robust?: an analysis of trust-based recommendation, *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI'06)*, Sydney, Australia. pp. 101–108.

OECD (2000). Guidelines for Consumer Protection in the Context of Electronic Commerce. Technical Report. OECD Publishing.

OECD (2005). Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. Technical Report. OECD Publishing.

O'Mahony, M., Hurley, N., Kushmerick, N., and Silvestre, G. (2004a). Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology (TOIT)* 4(4), 344–377.

O'Mahony, M.P. (2004). Towards Robust and Efficient Automated Collaborative Filtering. PhD dissertation. University College Dublin. Department of Computer Science, Belfield, Dublin 4, Ireland.

O'Mahony, M.P., Hurley, N.J., and Silvestre, G. (2004b). Utility-based neighbourhood formation for efficient and robust collaborative filtering, *Proceedings of the 5th ACM Conference on Electronic Commerce (EC'04)*, New York, NY, USA. pp. 260–261.

O'Mahony, M.P., Hurley, N.J., and Silvestre, G.C. (2002). Promoting recommendations: An attack on collaborative filtering, *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA'02)*, Aix-en-Provence, France. pp. 494–503.

O'Mahony, M.P., Hurley, N.J., and Silvestre, G.C. (2004c). Efficient and secure collaborative filtering through intelligent neighbour selection, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, Valencia, Spain. pp. 383–387.

O'Mahony, M.P., Hurley, N.J., and Silvestre, G.C. (2004d). An evaluation of neighbourhood formation on the performance of collaborative filtering. *Artificial Intelligence Review* 21(3-4), 215–228.

O'Mahony, M.P., Hurley, N.J., and Silvestre, G.C. (2005). Recommender systems: Attack types and strategies, *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, Pennsylvania. pp. 334–339.

Oostendorp, N., and Sami, R. (2009). The copied item injection attack. *Recommender Systems & the Social Web* 1001, 1.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes, *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*, Prague, Czech Republic. pp. 223–238.

Parkes, D.C., Rabin, M.O., Shieber, S.M., and Thorpe, C. (2008). Practical secrecy-preserving, verifiably correct and trustworthy auctions. *Electronic Commerce Research and Applications* 7(3), 294–312.

Pazzani, M.J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review* 13(5-6), 393–408.

Pedersen, T.B., Saygın, Y., and Savaş, E. (2007). Secret charing vs. encryption-based techniques for privacy preserving data mining, *Proceedings of the Joint UNECE/Eurostat Work Session on Statistical Disclosure Control*, Manchester, UK. pp. WP.4 1–11.

Polat, H., and Du, W. (2005a). Privacy-preserving collaborative filtering. *International Journal of Electronic Commerce* 9(4), 9–35.

Polat, H., and Du, W. (2005b). Privacy-preserving collaborative filtering on vertically partitioned data, *Knowledge Discovery in Databases: PKDD 2005*. Springer, pp. 651–658.

Polat, H., and Du, W. (2005c). Privacy-preserving top-$n$ recommendation on horizontally partitioned data, *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, Compiègne University of Technology, France. pp. 725–731.

Polat, H., and Du, W. (2005d). SVD-based collaborative filtering with privacy, *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05)*, Santa Fe, NM, USA. pp. 791–795.

Polat, H., and Du, W. (2006). Achieving private recommendations using randomized response techniques, *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'06)*, Singapore, Singapore. pp. 637–646.

Polat, H., and Du, W. (2007). Effects of inconsistently masked data using RPT on CF with privacy, *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC'07)*, Seoul, Republic of Korea. pp. 649–653.

Polat, H., and Du, W. (2008). Privacy-preserving top-$n$ recommendation on distributed data. *Journal of the American Society for Information Science and Technology* 59(7), 1093–1108.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of Netnews, *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, Chapel Hill, North Carolina, USA. pp. 175–186.

Resnick, P., and Sami, R. (2007). The influence limiter: Provably manipulation-resistant recommender systems, *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07)*, Minneapolis, MN, USA. pp. 25–32.

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender systems: Introduction and challenges, *Recommender Systems Handbook*. Springer, pp. 1–34.

Russell, S., and Yoon, V. (2008). Applications of wavelet data reduction in a recommender system. *Expert Systems with Applications* 34(4), 2316–2325.

Sandvig, J.J., Mobasher, B., and Burke, R. (2007). Robustness of collaborative recommendation based on association rule mining, *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07)*, Minneapolis, MN, USA. pp. 105–112.

Sandvig, J.J., Mobasher, B., and Burke, R.D. (2008). A survey of collaborative recommendation and the robustness of model-based algorithms. *IEEE Data Engineering Bullettin* 31(2), 3–13.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce, *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00)*, Minneapolis, MN, USA. pp. 158–167.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms, *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, Hong Kong, Hong Kong. pp. 285–295.

Schafer, J.B., Konstan, J.A., and Riedl, J. (2001). E-commerce recommendation applications, *Applications of Data Mining to Electronic Commerce*. Springer, pp. 115–153.

Shardanand, U., and Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, Denver, CO, USA. pp. 210–217.

Su, X., and Khoshgoftaar, T.M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 2009, 4.

Su, X.F., Zeng, H.J., and Chen, Z. (2005). Finding group shilling in recommendation system, *Proceedings of the Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW'05)*, Chiba, Japan. pp. 960–961.

Tang, T., and Tang, Y. (2011). An effective recommender attack detection method based on time SFM factors, *Proceedings of the IEEE 3rd International Conference on Communication Software and Networks (ICCSN'11)*, Convention Center of Xidian University, Xi'an, China. pp. 78–81.

Ungar, L.H., and Foster, D.P. (1998). Clustering methods for collaborative filtering, *Proceedings of the 1998 AAAI Workshop on Recommendation Systems*, Menlo Park, California. pp. 114–129.

Van Roy, B., and Yan, X. (2009). Manipulation-resistant collaborative filtering systems, *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, New York, NY, USA. pp. 165–172.

Van Roy, B., and Yan, X. (2010). Manipulation robustness of collaborative filtering. *Management Science* 56(11), 1911–1929.

Williams, C., Bhaumik, R., Burke, R., and Mobasher, B. (2006a). The impact of attack profile classification on the robustness of collaborative recommendation, *Proceedings of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, Philadelphia, PA, USA.

Williams, C., and Mobasher, B. (2006). Profile injection attack detection for securing collaborative recommender systems. Master's thesis. DePaul University. Department of Computer Science, Chicago, IL, USA. 1–47.

Williams, C.A., Mobasher, B., and Burke, R. (2007). Defending recommender systems: Detection of profile injection attacks. *Service Oriented Computing and Applications* 1(3), 157–170.

Williams, C.A., Mobasher, B., Burke, R., and Bhaumik, R. (2006b). Detecting profile injection attacks in collaborative filtering: A classification-based approach, *Advances in Web Mining and Web Usage Analysis.* Springer, pp. 167–186.

Witten, I.H., and Frank, E. (2005). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Wu, Z., Cao, J., Mao, B., and Wang, Y. (2011). Semi-SAD: Applying semi-supervised learning to shilling attack detection, *Proceedings of the 5th ACM conference on Recommender Systems (RecSys'11)*, Chicago, IL, USA. pp. 289–292.

Wu, Z., Wu, J., Cao, J., and Tao, D. (2012). HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation, *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*, Beijing, China. pp. 985–993.

Xia, H., Fang, B., Gao, M., Ma, H., Tang, Y., and Wen, J. (2015). A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique. *Information Sciences* 306, 150–165.

Yakut, I., and Polat, H. (2010). Privacy-preserving SVD-based collaborative filtering on partitioned data. *International Journal of Information Technology & Decision Making* 09(03), 473–502.

Yakut, I., and Polat, H. (2012a). Arbitrarily distributed data-based recommendations with privacy. *Data & Knowledge Engineering* 72, 239–256.

Yakut, I., and Polat, H. (2012b). Estimating NBC-based recommendations on arbitrarily partitioned data with privacy. *Knowledge-Based Systems* 36, 353–362.

Yakut, I., and Polat, H. (2012c). Privacy-preserving hybrid collaborative filtering on cross distributed data. *Knowledge and Information Systems* 30(2), 405–433.

Yang, Z., Xu, L., Cai, Z., and Xu, Z. (2016). Re-scale AdaBoost for attack detection in collaborative filtering recommender systems. *Knowledge-Based Systems* 100, 74–88.

Yao, A.C. (1982). Protocols for secure computations, *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS'82)*, Chicago, IL, USA. pp. 160–164.

Yi, H., and Zhang, F. (2016). Robust recommendation method based on suspicious users measurement and multidimensional trust. *Journal of Intelligent Information Systems* 46(2), 349–367.

Yılmazel, B.Y., and Kaleli, C. (2016). Robustness analysis of arbitrarily distributed data-based recommendation methods. *Expert Systems with Applications* 44, 217–229.

Yu, K., Schwaighofer, A., Tresp, V., Xu, X., and Kriegel, H.P. (2004). Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering* 16(1), 56–69.

Zhan, J., Hsieh, C.L., Wang, I.C., Hsu, T.S., Liau, C.J., and Wang, D.W. (2010). Privacy-preserving collaborative recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 40(4), 472–476.

Zhan, J., Wang, I.C., Hsieh, C.L., Hsu, T.S., Liau, C.J., and Wang, D.W. (2008). Towards efficient privacy-preserving collaborative recommender systems, *Proceedings of the IEEE International Conference on Granular Computing (GrC'08)*, Hangzhou, China. pp. 778–783.

Zhang, F., and Chen, H. (2016). An ensemble method for detecting shilling attacks based on ordered item sequences. *Security and Communication Networks* 9, 680–696.

Zhang, F., and Sun, S. (2014). A robust collaborative recommendation algorithm based on least median squares estimator. *Journal of Computers* 9(2), 308–314.

Zhang, F., Sun, S., and Yi, H. (2015). Robust collaborative recommendation algorithm based on Kernel function and Welsch reweighted M-estimator. *IET Information Security* 9(5), 257–265.

Zhang, F., and Zhou, Q. (2012). A meta-learning-based approach for detecting profile injection attacks in collaborative recommender systems. *Journal of Computers* 7(1), 226–234.

Zhang, F., and Zhou, Q. (2014). HHT-SVM: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems* 65, 96–105.

Zhang, F., and Zhou, Q. (2015). Ensemble detection model for profile injection attacks in collaborative recommender systems based on BP neural network. *IET Information Security* 9(1), 24–31.

Zhang, S., Chakrabarti, A., Ford, J., and Makedon, F. (2006a). Attack detection in time series for recommender systems, *Proceedings of the 12th ACM SIGKDD*

*International Conference on Knowledge Discovery and Data Mining (KDD'06)*, Philadelphia, PA, USA. pp. 809–814.

Zhang, S., Ford, J., and Makedon, F. (2006b). A privacy-preserving collaborative filtering scheme with two-way communication, *Proceedings of the 7th ACM Conference on Electronic Commerce, (EC'06)*, Ann Arbor, Michigan, USA. pp. 316–323.

Zhang, S., Ouyang, Y., Ford, J., and Makedon, F. (2006c). Analysis of a low-dimensional linear model under recommendation attacks, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, Seattle, Washington, USA. pp. 517–524.

Zhang, X.L., Lee, T.M.D., and Pitsilis, G. (2013). Securing recommender systems against shilling attacks using social-based clustering. *Journal of Computer Science and Technology* 28(4), 616–624.

Zhang, Z., and Kulkarni, S.R. (2013). Graph-based detection of shilling attacks in recommender systems, *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP'13)*, Southampton, United Kingdom. pp. 1–6.

Zhang, Z., and Kulkarni, S.R. (2014). Detection of shilling attacks in recommender systems via spectral clustering, *Proceedings of the 17th International Conference on Information Fusion (FUSION'14)*, Salamanca, Spain. pp. 1–8.

Zhou, Q., and Zhang, F. (2012). A hybrid unsupervised approach for detecting profile injection attacks in collaborative recommender systems. *Journal of Information & Computational Science* 9(3), 687–694.

Zhou, W., Koh, Y.S., Wen, J., Alam, S., and Dobbie, G. (2014a). Detection of abnormal profiles on group attacks in recommender systems, *Proceedings of the 37th international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'14)*, Gold Coast, Queensland, Australia. pp. 955–958.

Zhou, W., Wen, J., Gao, M., Liu, L., Cai, H., and Wang, X. (2015a). A shilling attack detection method based on SVM and target item analysis in collaborative filtering recommender systems, *Proceedings of the 8th International Conference on Knowledge Science, Engineering and Management (KSEM'15)*, Chongqing, China. pp. 751–763.

Zhou, W., Wen, J., Gao, M., Ren, H., and Li, P. (2015b). Abnormal profiles detection based on time series and target item analysis for recommender systems. *Mathematical Problems in Engineering* 2015.

Zhou, W., Wen, J., Koh, Y.S., Alam, S., and Dobbie, G. (2014b). Attack detection in recommender systems based on target item analysis, *Proceedings of the International Joint Conference on Neural Networks (IJCNN'14)*, Beijing, China. pp. 332–339.

Zhou, W., Wen, J., Koh, Y.S., Xiong, Q., Gao, M., Dobbie, G., and Alam, S. (2015c). Shilling attacks detection in recommender systems based on target item analysis. *PloS one* 10(7).

Zhou, W., Wen, J., Xiong, Q., Gao, M., and Zeng, J. (2016). SVM-TIA: A shilling attack detection method based on SVM and target item analysis in recommender systems. *Neurocomputing* 210(C), 197–205.

Zou, J., and Fekri, F. (2013). A belief propagation approach for detecting shilling attacks in collaborative filtering, *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management (CIKM'13)*, San Francisco, California, USA. pp. 1837–1840.