UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA - VARESE

DiSTA

Dipartimento di Scienze Teoriche e Applicate

PHD THESIS

to obtain the title of

Doctor of Computer Science

Defended by

NAEIMEH LALEH

RISK ASSESSMENT IN CENTRALIZED AND DECENTRALIZED ONLINE SOCIAL NETWORKS

Advisors: Prof. Elena Ferrari

Prof. Barbara Carminati

defended on September 5, 2016

To my wonderful mother and father...........

Words cannot describe how lucky I am to have you in my life. I would especially like to thank you for your love, support, and constant encouragement I have gotten over the years. Your love, laughter and music have kept me smiling and inspired.

Acknowledgment

Firstly, I would like to express my sincere gratitude to my advisers Prof. Elena Ferrari and Prof. Barbara Carminati for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better adviser and mentor for my Ph.D study.

Besides my advisers, I would like to thank the rest of my thesis committee: Prof. Paraskevi Fragopoulou, and Prof. Anna Cinzia Squicciarini, for their insightful comments and encouragement, but also for the hard questions which incented me to widen my research from various perspectives.

My sincere thanks also goes to Prof. Sarunas Girdzijauskas, and Prof. Anders Holst, who provided me an opportunity to join their team in KTH as intern, and who gave access to the laboratory and research facilities. Without they precious support it would not be possible to conduct this research.

Abstract

One of the main concerns in centralized and decentralized OSNs is related to the fact that OSNs users establish new relationships with unknown people with the result of exposing a huge amount of personal data. This can attract the variety of attackers that try to propagate malwares and malicious items in the network to misuse the personal information of users. Therefore, there have been several research studies to detect specific kinds of attacks by focusing on the topology of the graph [159, 158, 32, 148, 157]. On the other hand, there are several solutions to detect specific kinds of attackers based on the behavior of users. But, most of these approaches either focus on just the topology of the graph [159, 158] or the detection of anomalous users by exploiting supervised learning techniques [157, 47, 86, 125]. However, we have to note that the main issue of supervised learning is that they are not able to detect new attacker's behaviors, since the classifier is trained based on the known behavioral patterns. Literature also offers approaches to detect anomalous users in OSNs that use unsupervised learning approaches [150, 153, 36, 146] or a combination of supervised and unsupervised techniques [153]. But, existing attack defenses are designed to cope with just one specific type of attack. Although several solutions to detect specific kinds of attacks have been recently proposed, there is no general solution to cope with the main privacy/security attacks in OSNs.

In such a scenario, it would be very beneficial to have a solution that can cope with the main privacy/security attacks that can be perpetrated using the social network graph. Our main contribution is proposing a unique unsupervised approach that helps OSNs providers and users to have a global understanding of risky users and detect them. We believe that the core of such a solution is a mechanism able to assign a risk score to each OSNs account. Over the last three years, we have done significant research efforts in analyzing user's behavior to detect risky users included some kinds of well known attacks in centralized and decentralized online social networks.

Our research started by proposing a risk assessment approach based on the idea that the more a user behavior diverges from normal behavior, the more it should be considered risky. In our proposed approach, we monitor and analyze the combination of interaction or activity patterns and friendship patterns of users and build the risk estimation model in order to detect and identify those risky users who follow the behavioral patterns of attackers. Since, users in OSNs follow different behavioral patterns, it is not possible to define a unique standard behavioral model that fits all OSNs users' behaviors. Towards this goal, we propose a two-phase risk assessment approach by grouping users in the first phase

to find similar users that share the same behavioral patterns and, then in the second phase, for each identified group, building some normal behavior models and compute for each user the level of divergency from these normal behaviors. Then, we extend this approach for Decentralized Online Social Networks (i.e., DOSNs). In the following of this approach, we propose a solution in defining a risk measure to help users in OSNs to judge their direct contacts so as to avoid friendship with malicious users. Finally, we monitor dynamically the friendship patterns of users in a large social graph over time for any anomalous changes reflecting attacker's behaviors. In this thesis, we will describe all the solutions that we proposed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Social networks described as internet-based applications that allow users to create profiles and share content easily with other users. These social networking websites bring people together to talk, share ideas and interests, or make new friends. Many people join a social network because they want to stay in contact with their family or current friends.

More precisely, an online social networks (OSNs) is an online platform that provides social networking services for a user to create a public profile and create a friendship link between his/her profile and other users [18]. Despite the popularity of centralized OSNs, the users privacy and control over their data is becoming a major issue for these social networking services. These issues have motivated researchers to work on and propose several solutions in order to replace centrally controlled OSNs with Decentralized OSNs (DOSNs) [30] by retaining the functionalities offered by centralized OSNs. A DOSN is a distributed social networking service with no dependency on any central infrastructure [33].

In the same way, DOSNs allow users to create a public or private profile, encourage them to share information and interact with other users and communicate with each other in the virtual environment. Once a person creates a profile and start to use a social networking website, he/she will be in contact with other people including those people he/she does not know them, called strangers. Therefore, both in traditional centralized social networks and in decentralized's one, users are used to establish new relationships with unknown people by sharing a huge amount of personal data. The interaction of users in these sites reveals a lot of personal information visible to anyone who wants to view it. Unfortunately, some users are less concerned about information privacy; therefore, they post more sensitive information on their profiles without specifying appropriate privacy settings, and this can lead to security risks.

Risk is a situation involving exposure to danger, hazard, or mischance. Therefore, exposing to risks means to participate in an activity or event that could lead to damage, injury, or loss [132]. Risk has been part of every human life and then, all activities in our life carry an element of risk. Some of these risky activities may not be completely voluntary, as involuntary risks that are negative impacts associated with an occurrence that happens to us without our prior knowledge [97]. In some disciplines, a contrast is drawn between risk and threat [129]. A threat is an event with low probability, but very large negative consequences and analyzing and accessing the probability is impossible. In the contrary, a risk is an higher probability event, where there is enough information to make assessments of both the probability and the consequences. Risk assessment is a continuing process involves monitoring, analyzing, understanding, identifying and evaluating loss exposures to reduce the overall cost of operational risk. Therefore, risk management is very important and essential for all the environments in our life. Risk management enable societies to minimize the negative consequences and the threats of the associated risks and to remove or reduce risk exposures.

Today's globalized world and fast growing technological changes increase the interconnectedness and social networking to improve social relation among people [18]. These social relations increase the vulnerabilities and create new type of risks with impacts on a longer time-span or a much larger scale [52]. Most of the users are not aware how much it will be risky to expose the personal information as well as the serious consequences this might have. Because, some kind of attackers taking advantage of the popularity of the social networks and the users' trust in their relationship to propagate malware and malicious content and sending spam messages through the network [52]. On the other hand, there are some kind of attackers that steal the user's account and misuse them to propagate malware while users themselves are not aware of the consequences of stealing their accounts. Therefore, some of the information posted on these sites can lead to security risks such as, identity theft and cyber stalking. Moreover, there are some kind of attackers that create fake accounts and propagate spam massages and malicious links. In these cases, whenever a user click on these malicious contents, the message will post to his/her friends. On the other hand, creepers are those real risky users that make trouble for both users and service providers by posting some adv. items and encourage users to share those items in OSN for advertisement purposes. Besides, in some kind of attacks like Cyberbullying attack, real risky users send repeated hurtful messages to their victims. All these kind of attackers and risky users do some risky activities in OSN that make an OSN unsafe environment for both all other users and service providers.

As a result, the successful online social network needs a complete vision of risk management, where the service provider considers the solutions to protect the network against some risks, and also which risks to exploit and how to exploit them. Then, in order to get a safer environment, risk analysis and trust management in centralized and decentralized social networks are an essential and important element for successful social networking experiences.

1.1 Objective

The Marie Curie iSocial ITN project[1] aspired to bring a transformational change in Online Social Service provision, from centralized services towards totally decentralized systems.

[1] http://isocial-itn.eu/

OSNs decentralization can address privacy considerations and improve service scalability, performance and fault-tolerance in the presence of an expanding base of users and applications. The project has provided DOSNs services, in the absence of central management and control. Based on the main goal, iSocial project has consisted of several components: distributed and scalable overlay Infrastructure for DOSNs, distributed storage infrastructures that provide support for open social networks and for innovative social network applications, security, preserving end-user privacy and information ownership and risk/trust estimation and modelling/simulation.

At this purpose, as the first stage of research activities in iSocial project associated to security, risk management we made a comprehensive and detailed review of works related to risk concepts in the context of social networks. This pointed out that mechanisms for risk estimation have to be improved and adapted in order to be adopted in decentralized social networks.

1.2 Essential Approach

Based on our preliminary study, this thesis presents efforts we carried out to describe some of the main risks in centralized and decentralized OSNs and proposing some solutions for risk estimation by analyzing and monitoring the users's behavioral patterns. The key goal of this thesis is analyzing those risks associated with the behavior of users in OSNs/DOSNs. We need take into account technical capabilities and relative tendency of users to engage in risky behavior on these social networking sites by taking in consideration both users behavior (i.e., their patterns of interactions) and their social graphs (i.e., the network structure). Then, we can anticipate bad behavior and reputation damage so that these risks can be mitigated prior to adoption.

More precisely, the underling idea is that anomalous user behaviors can be risky in the network. As such, the more the user behavior diverges from 'normal behavior', the more he/she has to be considered risky. Thus, our goal is to detect anomalous users by considering user's interaction with all other users in the network. This implies to define at first what the normal behavior is, then detecting anomalies. We achieve this goal using outlier detection techniques to identify anomalous users with respect to their emerged behaviors.

1.3 Main Contribution

In summary, this thesis provides the following research contributions:

- The definition of the model for detecting risky users in centralized OSNs that follow the behavioral patterns of well-known attackers. Based on this principle, we propose a risk assessment based on the idea that the more a user behavior diverges from what it can be considered as a 'normal behavior', the more it should be considered risky. Because, although the dramatic increase in Online Social Networks (OSNs) usage, there are still a lot of security and privacy concerns. In such a scenario, it would be very beneficial to have a mechanism able to assign a risk score to each OSNs user. In doing this, we have taken into account that OSNs population is really heterogeneous in observed behaviors. As such, it is not possible to define a unique standard behavioral model that fits all OSNs users' behaviors. However, we expect that similar people tend to follow similar rules with the results of similar behavioral models. For this reason, we propose a *two-phase risk assessment model*, where users are first grouped together to find similar users that share the same behavioral patterns. Then, for each identified behavior, we build one or more normal behavioral models. To reach this goal, the key contributions include determining various user features to model normal/anomalous behaviors and assign a risk score to each OSNs user. The carried out experiments on a real Facebook dataset show that the proposed model outperforms a simplified behavioral-based risk assessment where behavioral models are built over the whole OSNs population, without a group identification phase;

- Proposing an approach for helping users to judge their direct contacts in OSNs by assigning a local risk score to them. This brings us to investigate a risk estimation measure by considering the topology of user's social graph. In particular, the proposed risk measure comes from the observation done in [5], where it has been highlighted that a user of a given network is anomalous if his/her subgraph significantly differs from those of other users. Therefore, we proposed a graph-based outlier detection methods tailored over features meaningful for the detection of risky behaviors in OSNs. The overall purpose is to obtain a local risk measure that helps users to detect potential attackers among their contacts. This consideration brought us to design a set of features defined based on attacker activity patterns. To prove the effectiveness of the proposed risk measure, we run several experiments on a real OSNs dataset (i.e., Orkut social network) with more than 3 million vertices and 117 million edges, by injecting synthetic fake users according to different settings and showing how the proposed measures can indeed help in their detection. The results presented in this chapter has been published in [84];

- Proposing a change detection approach able to identify users with anomalous changes in the structure of their subgraphs in OSNs. Effectively and efficiently detect these changes has the potential to enable the service providers of OSNs to anticipate and respond to the attackers and risky users. In particular, we were interested in detecting user changes that can be considered anomalous compared to: (a) other similar users with the same change patterns, and (b) his/her own previous change patterns in the past. More precisely, we analyze and monitor the change patterns of users over time and compare them with their own previous change patterns and the change patterns of other similar users in the network and measure the degree of changes for each user, by thus providing a level of anomaly, which can be used to trigger the proper response. Our approach returns a list of these users by ranking them based on the value of their change deviation. We analyze the performance of our approach on a real Google+ dataset;

- Proposing a distributed approach for identifying behavioral groups of users that share the same behavioral patterns in decentralized social networks. In DOSNs, the aim is to give the users control over their data and keeping data locally to enhance privacy. In the fully distributed social graph, each user has only one feature vector and these vectors can not move to any central storage or other users in a raw form duo to privacy issues. The main contribution of this approach is adopting a distributed clustering algorithm to be applicable on top of DOSNs and apply it to identify behavioral groups of users. Our goal is to achieve an accuracy comparable to a centralized scheme by considering both social and individual patterns of users. Moreover, in this proposed approach, feature values of users are never send over the network in a raw form and the approach has low computation and communication cost. In order to evaluate our approach, we implement our algorithm and test it in a real Facebook graph;

- Proposing a solution that enables a target user in DOSNs to assign a risk score to other users which send friend request to him/her and also his/her direct contacts by considering their activities and friendship patterns in the network. The goal is to compare the behavioral patterns of users with other similar users in the network to find misbehaviors in a distributed manner. More precisely, we proposed a distributed two-phase risk assessment approach by grouping users in the first phase based on their group identification features and then, in the second phase, each user builds one or more behavioral models for his/her identified group and other groups by defining various user features to model normal/anomalous behaviors.

1.4 Thesis Organization

The thesis consists of seven chapters, whose content is briefly described in what follows:

- Chapter 2- In this chapter, we review several current attack detection/risk assessment approaches in OSNs/DOSNs and their drawbacks. Moreover, we summarize several graph based anomaly/outlier detection approaches and those related literature for distributed community detection and group identification approaches;

- Chapter 3- We start to propose a comprehensive unsupervised approach to detect risky users in OSNs that follow the behavioral patterns of well-known attackers. Towards this goal, we propose a risk assessment approach organized into two phases: similar users are first grouped together, then, for each identified group, we build one or more models for normal behavior. In this chapter, we review several attackers in OSNs and all the details related to their behavioral patterns in the network;

- Chapter 4-This chapter describes and discusses an efficient and effective measure helping users to judge their direct contacts so as to avoid friendship with malicious users that could misuse their personal information. At this purpose, in this chapter we propose a risk measure, called *local risk factor*, having as a key idea the fact the malicious users in OSNs (aka attackers) show some common features on the topology of their social graphs, which is different from those of legitimate users;

- Chapter 5- Once we have done the local risk estimation in the previous chapter, we propose a dynamic solution for highlighting of anomalous changes in a sequence of social graph snapshots to detect risky users in OSNs by monitoring and analyzing their behavior over time. This is interesting due to its numerous applications. For instance, it may be helpful for the identification of attackers or risky users in Online Social Networks (OSNs). Indeed, dynamically monitoring and learning the friendship patterns of users in a large social graph over time for any anomalous change often reflects and predicts significant events or attacker's behaviors. In this chapter, we focus on anomalous changes that happen in the neighborhood of OSNs users. Our main goal is to assign a risk score to each user by considering the anomalous changes that user has in the structure of his/her subgraph in compare with his/her own previous change patterns in the past and from those of other nearest users in the graph;

- Chapter 6- The main goal in this chapter is identifying behavioral groups of users that share the same behavioral patterns in decentralized OSNs. We propose a distributed approach to identify behavioral group of users that share the same behavioral patterns in DOSNs. We use a gossip learning approach where all users are involved with their local estimation of the clustering model and improve their estimations and finally converge to a final clustering model available for all users;

- Chapter 7- In this chapter, we extend the sixth chapter by proposing a distributed two-phase risk estimation approaches to detect risky users in DOSNs based on their behavioral patterns. In DOSNs, each user has a single feature vector including his/her interactions and personal information and these local information cannot be moved to a central server or to other users in a raw form due to privacy issues. Therefore, this can attract a variety of privacy and highly damaging attacks. These attackers forward spam and malware on online social network. Since attackers have weird behavioral pattern in the network, our goal is to analyze the behavior of users (interactions or activity patterns) in DOSNs by identifying those risky users whose follow the behavioral pattern of attackers. More precisely, when the user behavior diverges from 'normal behavior', the user will be considered as risky.

1.5 Related Publications

The research activities described in this thesis have bring to the following publications:

- Naeimeh Laleh, Barbara Carminati, Elena Ferrari, "Graph Based Local Risk Estimation in Large Scale Online Social Networks," In: IEEE conference in social computing and networking 2015, SocialCom2015;

- Naeimeh Laleh, Barbara Carminati, Elena Ferrari, ”Risk Assessment in Online Social Networks based on Anomalous Behavior Detection,” in IEEE Transactions on Dependable and Secure Computing 2016, accepted, To appear;

- Naeimeh Laleh, Barbara Carminati, Elena Ferrari, Sarunas Girdzijauskas, ”Distributed Gossip based Behaviour Group Identification in Decentralized Online Social Networks,” International conference in Machine Learning and Data Mining, MLDM 2016, Springer, New York, USA, Accepted, To appear;

- Naeimeh Laleh, Barbara Carminati, Elena Ferrari, “Dynamic Anomalous Change Detection in User’s Neighborhood in Time-evolving OSN Graphs,” in 15th IEEE IFIP annual Mediterranean Ad Hoc Networking 2016, Accepted, To appear;

- Naeimeh Laleh, Barbara Carminati, Elena Ferrari, Anders Holst, Sarunas Girdzijauskas, “Distributed Gossip Based Risk Assessment in Decentralized OSN based on Anomalous Behavior Detection,” Under preparation.

Chapter 2

Related Work

2.1 Introduction

The popularity of online social networks attracts a variety of attackers and malicious users which try to misuse the personal information of users in the network. Therefore, to address the growing problem of discovering malicious activities on social networks, researchers have started to propose different detection and mitigation approaches. In particular, the first category of approaches, focuses on detecting attackers and fake accounts in centralized social network. Therefore, first we discuss the related work for risk estimation on OSNs in Section 2.2. The second category of approaches try to detect anomalous users in social graphs (see Section 2.3. After that, we cover those approaches to detect changes in time evolving graphs in Section 2.4. Afterwards, we will give the state of art in community detection for risk estimation purposes in Section 2.5 and distributed risk estimation approaches on decentralized social networks in Section 2.6.

2.2 Risk Assessment in OSNs

A first stream of papers focuses on detecting attackers and fake accounts in centralized social networks. Some of these approaches are graph based and others are behaviour based.

Graph-based sybil detection schemes make assumptions about the OSNs graph growth and structure. Based on these assumptions, researchers use various graph analysis techniques to develop algorithms for sybils detection, such as sybilGuard [159], sybil-Limit [158], sybilinfer [32], and SumUp [148]. However, recent studies pointed out that these assumptions might not always hold. Indeed, it has been observed that sybils mix well into the rest of OSNs graphs [157], and that most of OSNs graphs are not fast-mixing [109]. These have implications in the proposed detection schemes, in that these may end up in false positive and false negative results [152, 15]. Compared to graph-based sybil defense techniques, our proposed risk assessment model is more flexible as it does not rely on the same assumptions, since we consider the activity patterns of sybils after they joined the

OSNs.

Most recent behavior-based approaches for the detection of anomalous users in OSNs exploit supervised learning techniques [157, 47, 86, 125]. As an example, in [157] to detect sybils, the proposed system trains a classifier by extracting four features, like: accepted incoming requests, accepted outgoing requests, invitation frequency and clustering coefficient. In [47], also authors proposed a supervised approach to detect compromised account attack by using a small manually labeled dataset of legitimate and anomalous users. [86] and [125] used classifiers to detect spam and malware respectively. However, we have to note that the main issue of supervised learning is that they are not able to detect new attacker behaviors, since the classifier is trained based on the known behavioral patterns. Literature also offers approaches to detect anomalous users in OSNs that use unsupervised learning approaches [150, 153, 36, 146]. As an example, [150] used Principal Component Analysis to detect anomalous users in OSNs. Their approach provides a framework for modeling user behavior in an OSNs and leverage the user behavioral features to detect misbehavior. They detect anomalous users based on the number of likes per day. Although there is an issue that attackers can distribute their likes on several days to avoid detection and attacker with low level of activity can not be detected because of intermixing between legitimate and anomalous behavior. In [153], the author proposed a combination of supervised and unsupervised techniques by analyzing the clickstream behavioral of users in OSNs in order to detect sybils.

Our analysis of attack behavior and characteristics in Chapter 3 demonstrates that most of the current unsupervised techniques are quite ad-hoc and complex. Some of these unsupervised behavior-based risk models suffer from high false negative and positive rates, due to the large variety and unpredictability of behaviors of both legitimate and malicious OSNs users. In addition, existing attack defenses are designed to cope with just one type of attack. However, given the presence of several type of risky users in OSNs, we believe a more comprehensive approach to effectively detect and defend against them is needed.

Similarly to the proposal presented in Chapter 3, a risk measure for OSNs users has been proposed also in [3]. However, in [3] authors defined the local risk measure, that is, a measure computed only considering the similarity between two target users (e.g., network similarity and profile similarity). In contrast, we propose a more general risk measure that takes into account behavioral patterns of a target user and compares them with the rest of network.

2.3 Graph Based Local Risk Estimation in OSNs

To cope with emerging security and privacy concerns, research community has started to deeply investigate and propose mechanisms for safer and trustworthy OSNs. Our goal is to investigate a risk estimation measure based on outlier detection approach by considering the topology of user's social graph to help users to judge their contacts. Relevant for our proposal are the works targeting graph-based outlier detection (see [6] for a survey). Among them, structure-based approaches make use of graph-centric features, such as node degree and subgraph centrality [63], that are sometimes used together with other features extracted from additional information sources to identify outliers.

The feature-based approaches have been used in several anomaly detection application domains, including network intrusion detection [41], web spam detection [11] and, fraud detection [83].

One of the research works, ODDBALL [5] extracts ego network features by considering one step subgraph, such as the degree, total weight, principal eigenvalue, etc. to find patterns that most of the nodes of the graph follow with respect to those features and spot anomalous nodes as those that do not follow the observed patterns. In our approach we consider different features driven by the structural behavior of attackers in real OSNs. In addition, we considered the 2 step subgraph (the network of all direct contacts of ego). Because based on the behavior of attackers in OSNs, considering only ego network features is not enough. More precisely, researchers stated that most sybils and fake accounts can not create link with normal users and most of their friends are either sybils or popular users [17]. Therefore, considering the network of direct contacts of attackers is important to reveal these kinds of structural behavior. Another research work use recursive graph based features to capture behavioral information for classification and de-anonymization tasks without the availability of class labels [64], although their goal is not anomaly detection.

One of the application of anomaly detection in OSNs is spam filtering. [56] performs online spam filtering on social networks using incremental clustering, based on network-level features such as sender's degree and the interaction history between users.

In addition to anomaly detection, there are some approaches for Sybil detection [14, 25]. These approaches uses different graph analysis algorithms to search for legitimate and Sybil users. Although, these schemes work by analyzing the structure of the social network, all of them make three common assumptions. First, the legitimate region of the graph is densely connected. Second, attackers cannot establish a high number of social connections to legitimate users. Third, the system is given the identity of at least one legitimate user. Thus, the performance of these schemes is heavily dependent on the size and characteristics of the community surrounding the legitimate users. Furthermore, another approach is [15] that is a combination of graph and content based to detect Sybils. All of the above mentioned approaches are supervised.

As the risk assessment in OSNs is concerned, [3] propose a measure for risk estimation by considering the profile similarity and number of mutual friends that a target user has with other strangers as a measure that how much is risky to become friend with a stranger. They used supervised classification to assign a risk score. However, due to the challenges in obtaining labels, supervised learning algorithms are less attractive for the task of risk assessment. In our proposal, we focus on risk assessment in online social networks based on unsupervised graph based anomaly detection.

2.4 Anomalous Change Detection in Time-evolving OSNs

Detection of anomalous changes in time-evolving graph has been widely studied in the context of mining and statistics [91, 75, 82, 9]. However, there are few approaches able to detect change in dynamic social graphs.

In general, these approaches extract a summary of each graph snapshot to be compared over time with the help of similarity functions [121]. Then, when the distance returned by the considered similarity function is higher than a threshold, the corresponding graph snapshot is flagged as anomalous. The goal of these approaches is detecting the time of the anomalous changes in the structure of the whole graph. As an example, [136] computes distance functions among a sequence of graphs, whereas[144] proposes an approach for detecting changes in community structure to identify times of these changes. [103] and [102] consider some network measures, such as closeness centrality, betweenness centrality and the density of the graph, to detect any change in the graph over time.

Another approach is [23], where, in addition to compute the distances between consecutive graph snapshots, the distances between all pair of nodes in the graph are computed as well. They detect the changes that occur in the time evolving graph based on the distances between pair of users. [77] proposes a graph feature-based similarity approach to compare the pairwise node similarity. But, computing all the pairwise similarity scores has high computational cost. A faster algorithm has been proposed to avoid computation of similarity among all pairwise nodes [6]. Recently, [28] proposes a statistical approach to detect change points.

However, these approaches are not able to detect which nodes are responsible for the detected changes. On of the interesting research work in this direction is [4], which considers individual local features (e.g., in-degree, out-degree, in-weight, etc.) for each user and calculates the correlation between these features value over time for all pair of users, to detect change times in the whole graph structure. However, they identify change points where the majority of the users in a whole graph deviate from their normal behaviors. Therefore, if the majority of the users do not deviate from their normal behavior, this approach fails in detecting the changes.

A most recent work, that is, [137], in addition of finding the changes in the whole graph, finds also which nodes or edges are responsible for these changes. However, their key goal is the detection of changes in the whole graph structure.

2.5 Behavioral Group Identification in DOSNs

Work related to this research can be divided into three major areas: community detection, distributed clustering algorithms in large distributed networks, and distributed EM algorithms.

In the context of community detection, there are plenty works that return clusters of similar users, where users in each cluster are strongly connected with each other. These community detection approaches focuses on identifying tightly-connected clusters (communities), relying on topological structures [38, 48], by considering the common neighbors between two user. These methods are based on Minimum-cut [114], maximal clique [2], modularity [115], edge betweenness [58] and etc. But, those topological based community detection approaches fail to group users with the same behavioral pattern in that they might belong to different communities based on their friendship links. For example, the popular users with high number of friendships can not be considered in the same community by community detection approaches if they don't have any friendship. Furthermore, active users with a high level of activities in the network will be in different communities if they don't have any friendship or interaction [111, 40]. In addition, there are some works that use user's actions to find popular people in online social network [94]. The researchers use some centrality measures like closeness centrality, degree centrality, eigenvector centrality, betweenness centrality and page rank in order to define users popularity. But, the limitation of these approaches is that in order to have popular users with high level of activity, high number of friendships and same feature vectors in the same community, they need to construct a graph and the major problem of these community detection methods is graph construction and scalability [40].

Moreover, researchers proposed some content based community detection methods, that are relying on the analysis of the content generated by each user (e.g., features of users are in a d-dimensional space) [111, 94]. The goal of these community detection approaches is finding similar users with no inherent graph structure. Therefore, in order to apply graph clustering algorithms for users, the researchers first construct a similarity graph, based on the distance between user's feature vector. Then, the problem of clustering the set of feature vector of users will be transformed to a graph clustering problem. For example, [111] combines graph structure and the action of users (i.e., sharing the same items) to identify similar users. The problem of these content-based approaches is that in order to measure the similarity, users need to send all their private information to their direct friends or other users that is a big issue in our application due to privacy. On the other hand, this identification can not be made when the data has many more dimensions (e.g., with the mixture of discrete and continuous features). Because, the various behavioral patterns in the existence of both discrete and continuous features may not be obvious by using similarity measures. Also, graph construction based on similarity measures will bring another level of complexity (e.g. defining the edges weight based on the number of similar features between two nodes in order to connect them in the graph, the required guarantee for the connectivity of all nodes in the graph and the cut-off threshold to have a link). Moreover, these content based approaches are not suitable for real-time applications that require online analysis such as risk assessment, recommendation systems and load balancing due to the problems of graph construction [40].

Some of the community detection methods are stream-based, suitable for real-time applications, that is, the researchers incrementally find communities in dynamic graphs (sequence of graphs). In some of them, they first assign nodes into communities and then quantify the similarity between different communities and detect the change of the community over time [90, 44, 116]. However, most of these approaches are link-based or interaction-based, relying on the level/degree of interaction among users rather than mere friendship [92, 40], and they didn't consider the personal behavioral data including profile and activity information of users that are in d-dimensional space.

In the second category of related works, there are several distributed clustering algorithms. The most naive approach is simply to build and use a single clustering model for each user individually based on his/her local dataset (i.e. called local clustering model) [65]. However, in general the performance of using local clustering models improves slowly due to the lack of data [65, 128]. Moreover, toward our goal in order to identify group behavior, the local clustering models are not efficient duo to the lack of behavioral pattern of users (i.e., every user has only his/her own single feature vector). There are other approaches that users share the data between themselves to have more data samples [155]. But, users need to share all their private information that is the first issue. Also, some of these approaches lead to intensive communication among them, which degrades the scalability [12]. However, there is a graph partitioning approach that is applicable for scalable social graph, but, each user requires to access the data of both his/her direct friends and a small subset of random users in the graph [124]. But, in our application personal information of users can not be moved to their friends duo to the privacy issues. On the other hand, their approach is link-based and they didn't consider the feature vector of users. Another approach is to organize the clustering model in a hierarchical fashion [128, 45, 65] by which local clustering models are computed first for each user individually, and sent to a logically higher-level user that aggregates local models, and then, returning results to the lower-level users for further processing [93]. In such approaches, output from the algorithm is much dependent on the processing of the highest level user. However, in our application users need to share all private information (their profile and behavioral information) that is a big issue as well as in some other applications, so it is essential to process them locally. But, it is not possible to learn from local models because of the lack of information. Besides, the communication cost needs to be kept low during our learning process that sharing all the raw data of users are costly.

Distributed Computation in large distributed systems are included approximate algorithms that computes the approximate data mining results and can be deterministic or probabilistic.

In the deterministic averaging techniques such as graph-laplacian [34] and linear dynamical systems [131], each user repeatedly select all his/her immediate neighbours to update the local parameters estimation. For example, authors is [62] proposed an EM algorithm based on this approach. In [112] the authors have proposed distributed algorithms for inferencing in wireless sensor networks. Also, converging algorithms for computing simple computation such as mean, sum or other computations have been proposed in [69]. However, these techniques are not suitable for social network that cause to overestimate the parameters estimation because of the existence of some high degree users.

Another kind of averaging techniques is probabilistic gossip based approach [19, 78] that at each iteration each user repeatedly selects a uniform random user and both users compute the average of their parameters estimation. For instance, the distributed k-Means algorithm in [10], the newscast model in [78] and, the gossip-based protocols in [19] are all based on gossip learning. Also, Newscast EM [79] is based on this approaches and some of it's application includes Multi-camera tracking [107] and distributed multimedia indexing [117]. Since for applying Newscast EM in social network, the network need to be fully connected and the user selection need to be done uniformly at random, we use gossip based peer sampling service on social network to afford this requirement.

In addition, there is another work in fully distributed gossip based linear models that there are a lot of local learning models that perform a random walk in the network and update the local models to converge to a global model [118]. However, our proposal in Chapter 6 is having one learning model that will be converge to a global model ready to use for each user.

2.6 Distributed Risk Assessment in DOSNs

The work related to our research work in Chapter 7 can be subdivided into three major areas: discovery of malicious activities on social networks based on graph or behavior of attackers, distributed EM algorithms and computation in large distributed systems a.k.a P2P systems.

For discovering malicious activities on social networks, several centralized solutions proposed for detecting fake/sybil accounts in OSNs. Some of them are graph-based, relying on topological structure of the graph, like sybilinfer [32], Canal [151], Sybil Defender [154], Cai and Jermaine [24], SybilRank [25], and SumUp [148]. The draw back of these approaches is that they made some assumptions about the OSNs graph growth and structure, that these assumptions are not hold in real Renren social graph [157] [1]. In our risk assessment approach, we consider the activity patterns of sybils in order to detect them.

On the other hand, there are approaches that consider the behavior of users to detect anomalous users in OSNs. Some of them exploit supervised learning techniques [47, 86, 125, 157] to detect compromised account attack, spam, malware and sybils respectively. The drawback of supervised techniques is that they train a classifier based on the known behavioral patterns, and they are not able to detect new behavioral patterns of attackers. Detecting anomalous users in OSNs based on unsupervised learning approaches proposed in several studies like [150, 153, 36, 146]. For instance, [150] proposed a misbehavior detection approach based on PCA (Principal Component Analysis) by considering the number of likes of users per day. Moreover, in [153] and in [15], the authors proposed a combination of supervised and unsupervised techniques by analyzing the clickstream behavioral of users and behavioral activity of users in OSNs respectively, in order to detect sybils. Most of the existing approaches are designed to cope with just one type of attack. But, in [85], the authors proposed a comprehensive approach to effectively detect and defend against several type of risky users in OSNs. They proposed several behavioral features of users according to the behavior of real attackers in OSNs such as, the number of friends, the average number of mutual friends, the ratio between the number of friends of mutual friends, the number of likes, number of comments, number of posts, the ratio between the number of posts and the number of feedback received on them, and etc.

[1] The popular social network in China: http://www.renren-inc.com/en/

In the distributed setup, some decentralized topological based approaches like sybil-Guard [159], sybil-Limit [158], GateKeeper [149], MobID [123], Whanau [88], SybilShield [135] and VoteTrust [156] are proposed. But, as we mention above, they make some assumptions about the graph structure and growth that do not hold in real social graph. Based on our best understanding, there is no decentralized research work to detect malicious users in DOSNs based on their behavioral patterns.

In distributed EM research, the naive approach is simply aggregating all the data at the central location that is not suitable for large asynchronous networks. The other approach is simply to divide the whole data streams into a set of substreams and to build and use local models on each substream. In this case, the performance of the local models improves slowly due to a lack of training samples. To improve the performance of local models, another approach is sharing the training data between users to have more training samples. However, this has a large communication cost which degrades the scalability [12]. In [79] a fully distributed EM algorithm, Newscast EM, is proposed which uses gossip-style distributed computation to compute the parameters of the Maximization-step in peer-to-peer network. The authors prove that this gossip-based algorithm converges to the correct result exponentially fast.

Several applications for distributed EM algorithms have also been proposed in the literature. Multi-camera tracking [107], distributed multimedia indexing [117] are some of the examples. Although, the existing approaches try to detect malicious behaviors in OSNs, we believe proposing a distributed risk assessment approach to effectively detect and defend against risky users in DOSNs, is needed. Because, detecting risky users when the feature vector of users are fully distributed and can not move thorough the network in a raw form is challenging. We used newscast EM algorithm in our two-phase risk assessment approach.

Chapter 3

Risk Assessment based on User Anomalous Behaviors

3.1 Introduction

Although there is a dramatic increase in OSNs usage – Facebook, for instance, has now 1.55 billion monthly active users, 1.31 billion mobile users, and 1.01 billion daily users[1] there are also a lot of security/privacy concerns. Because, today's social networks are exposed to many types of privacy and security attacks. These attacks convincing users to click on specific malicious links with the aim of propagating these links in the network [51]. These attacks can either target users personal information as well as the personal information of their friends. Another widely used attack is the generation of fake profiles, which are generated with the only purpose of spreading malicious content. In addition, there is a growing underground market on OSNs for malicious activities in that, for just few cents, you can buy Facebook likes, share, Twitter followers, and fake accounts.

In this chapter, we propose a risk estimation service that allow a user to make more conscious decisions about his/her privacy-risky activities within the network (e.g., answering to a friend request). Moreover, conducting a risk assessment in OSNs will allow the service providers to minimize risks and help users to create and maintain a healthier friendship environment. We believe that a risk score can be useful for those users who want to inspect their contacts, and also for the service providers wishing to know which users are risky.

Therefore, our goal in this chapter is to assign a risk score to each user, by taking into account both the user's activities and friendship patterns in the network. The goal is to compare the behavioral patterns of users with other users in the network to find anomalous behaviors. The key idea is that the more the user behavior diverges from what it can be considered as a normal behavior, the more it should be considered risky (i.e., with high risk score). Following this principle requires to address two main issues. The first is the definition of a *user behavioral profile* able to catch those user's activities and interactions that are considered meaningful for risk assessment. The second issue regards

[1] http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

how to model a normal behavior. In doing this, we have to consider that OSNs users are really heterogeneous in observed behaviors. However, similar to real world, we expect that similar users (e.g., similar in activity level, gender, education, country, and so on) tend to follow similar rules (e.g., moral, social) with the results of similar behavioral models [7, 140]. Based on this principle, we propose a *two-phase risk assessment*, where users are first grouped together according to some features meaningful for group identification. Then, for each identified group, we build one or more normal behavioral models. To reach this goal, the key contributions include determining various user features to model normal/anomalous behaviors, and integrating them with probabilistic-clustering approach (the Expectation Maximization algorithm). As it will be illustrated in the chapter, we carried out experiments on a real Facebook dataset to show that the proposed *two-phase risk assessment* outperforms a simplified behavioral-based risk assessment where behavioral models are built over the whole OSNs population, without a group identification phase.

The rest of this chapter is organized as follows. Section 3.2 introduces the overall idea underlying our approach, whereas Section 3.3 provides a summary of the considered attacks. Grouping and behavioral features are presented in Section 3.4. Section 3.5 illustrates the clustering approach as well as the risk definition. Finally, experiments are presented in Section 3.6.

3.2 Risk Assessment based on user behaviors

As introduced in the previous section, our main goal is to associate a risk score with a user based on how he/she behaves in the OSNs. More precisely, the key idea is that the more the user behavior diverges from what it can be considered as a normal behavior, the more the user should be considered risky. Therefore, we should first define a *user behavioral profile* able to catch those user's activities and interactions that we consider meaningful for risk assessment.

In an OSN, a variety of activities are possible, such as writing comments/posts, reading, or sharing items, as well as different types of interactions, like commenting on a users' post, viewing profile information, assigning likes, joining special groups or pages, sending invitations to others. In designing a behavioral profile we do not aim at monitoring all users' activities, but only those that might reveal risky conducts. As an example, writing a lot of comments/posts without receiving any like on them can be considered a warning that the corresponding user might be a victim of an attack. On the contrary, simply having a high number of friends, posts, comments and likes can not be considered as a risky behavior. However, having a high number of friends, posts, comments and likes in a short period of time can be considered risky. In order to identify a set of *behavioral features*, meaningful for risk assessment, we have deeply reviewed the literature looking for well-known OSNs attacks (see Section 3.3).

The second issue to be addressed regards how to model a normal behavior that, according to the proposed approach, has to be used as baseline to measure how much users diverge from it and thus to compute the corresponding risk score. In doing this, we have

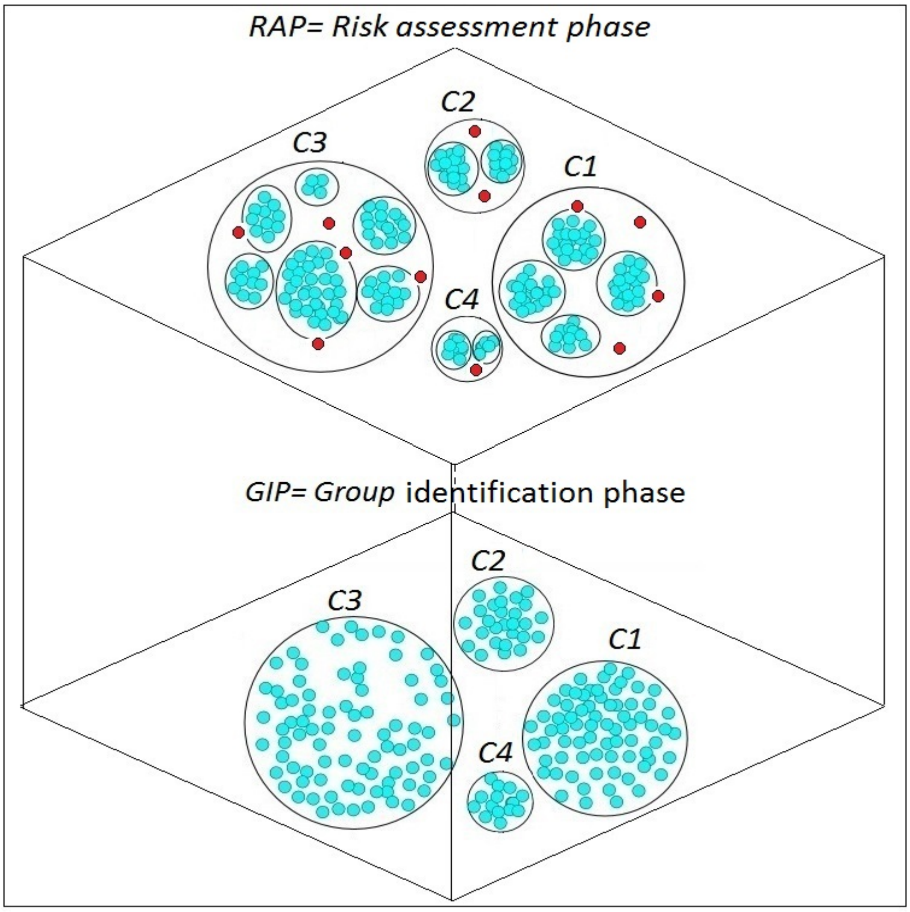

Figure 3.1: Two phase risk assesment

to consider that OSNs population is really heterogeneous in observed behaviors. As such, it is not possible to define a unique standard behavioral model that fits all OSNs users' behaviors. However, as mentioned in the introduction, we expect that similar people tend to follow similar rules with the results of similar behavioral models. Based on this principle, the proposed approach performs a first phase aiming at *identifying groups in the OSNs*. This is achieved by exploiting clustering techniques over a set of user features meaningful for group identification, called, in what follows, *Group identification features*.

As depicted in Figure 3.1, once groups have been identified it is possible to determine one or more normal behavioral models for each of them. The identification of these models is again performed by exploiting clustering techniques that, in this second phase, only exploit behavioral features. As such, the aim of the second phase is creating behavioral models for each group identified in the first phase. The clusters obtained as the result of the second phase allow us to associate a risk score to a target user by calculating the divergency of his/her behavior from each identified behavioral model (see Section 3.5 for more details).

3.3 Risky behaviors in OSNs

As social networking sites have risen in popularity, cyber-criminals or attackers started to exploit them in propagating malwares and carrying out scams [47]. The activity patterns underlying attacks are, in general, different from those of normal users. The discrepancy is defined in terms of frequency, number, as well as type of activities.

In general, attackers use the OSNs infrastructure to collect and expose personal information about a user and their friends, by even successfully compromising users into clicking on specific malicious links so as to propagate them in the network [51]. Notable types of attack are: socialbots [17, 49], sybil attacks [42], identity clone attacks [71, 73], [89], compromised account attacks [51, 139, 47], socware [125, 67], creepers [139], cyberbullying [106, 104], and clickjacking [96]. In the following, we describe the activity behavior of these types of attacks in summary.

Sybil attacks. Sybil attacks are one of the most prevalent and practical attacks in OSNs [87, 113]. As an example, more than an hundred socialbots have been detected on Facebook [16]. To launch a sybil attack, a malicious user has to create multiple fake identities [72], known as sybils, with the purpose to legitimate his/her identity [53], so as to unfairly increase his/her power and influence within a target community [157]. After that, attackers initiate their work by sending friendship requests to other users in the community. Once a request has been accepted, the socialbot can gather users' private data. Researchers have also observed sybils forwarding spam and malware on Facebook [57] and Twitter [61].

In general, we can classify sybil attacks into two types. The first is sybils with *tight-knit community (dense friendship graph).* According to these attacks, adversaries create huge number of sybils by also establishing connections among them [53]. These are beneficial since they make sybils appear more legitimate to be normal users. Therefore, many proposals, such as sybilGuard [159], sybilLimit [158], sybilInfer [32], and SumUp [148] try to detect sybils with tight-knit communities based on the links creation among them.

Authors in [157], have analyzed the distribution of sybil accounts in the OSNs Renren.[2] This analysis shows that the vast majority of sybil accounts do not form social links among them. Moreover, even in case they form, the resulting clusters are loose, rather than tightly knit. As such, the sybils shows to belong to another type, that is, those with *sparse community (sparse friendship graph).* Also, they found that attackers use snowball sampling techniques to identify and send friend requests to popular users, since these are more likely to accept requests from strangers [157, 53]. Therefore, sybils have friendship links with a lot of strangers and their friendship graph become sparse.

A final note is about the generation of fake profiles (also referred to as socialbots), which are automatic or semi-automatic profiles that mimic human behaviors [51]. Even if these can be also created by non-malicious users wishing some extra accounts, more commonly, fake accounts are generated by attackers that want to compromise and influence regions of

[2] http://www.renren-inc.com/en/

the graph [139]. In general, these can be created for several purposes, like: friendly pranks, stalking, cyberbullying, and concealing a real identity to bypass real-life constraints [25]; for profitable malicious activities, such as spamming, click-fraud, malware distribution, and identity fraud [130]. Additionally, fake profiles can be used to initiate sybil attacks [42], manipulate OSNs statistics [142, 147], or publish spam messages [57]. The market of buying fake followers and fake retweets is already a multimillion-dollar business [120]. **Identity clone attacks.** In this type of attack, a malicious user creates similar or even identical profiles to impersonate victims in an OSNs. The key goal is to obtain personal information about a victim's friends after successfully forging the victim, and to establish increased levels of trust with the victim's social circle for future deceptions. The cloned identity can also be used to propagate malicious messages to other users in the network [71]. To run this attack, the adversary first tries to obtain a victim's personal information, such as name, location, occupation, and friends list from his/her public profile on OSNs or his/her personal homepage(s) [51]. Then, the adversary forges the victim's identity. Afterwards, he/she sends friend requests to the victim's contacts. Once the friend requests have been accepted, he/she builds the victim's friend network and gains access to the profiles of the victim's friends. In general, exploiting the victim's friend network, the cloned account forms a network structure similar to the structure of sybils with tight-knit communities.

Compromised accounts attacks. Compromised accounts are accounts where legitimate users have lost complete or partial control of their login credentials. Accounts can be compromised in a number of ways, for example, by exploiting a cross-site scripting vulnerability, or by using a phishing scam to steal the user's login credentials. Also, bots have been increasingly used to harvest login information [141]. Compromising legitimate accounts is very effective, as attackers can leverage the trust relationships that the legitimate account has established in the past, to spread malicious content more effectively [13]. Detecting this kind of attacks based on link creation and profile information is almost impossible, since the victim is a legitimate user with real profile and real friendship edges. However, since attackers increasingly abuse legitimate accounts to distribute their malicious messages [57, 61], it is possible to detect these victims based on their behavioral patterns in the network.

For example, in [150], authors analyzed the browser malware Febipos that infects user's browsers and silently performs actions on Facebook and Twitter using the credentials/-cookies stored in the browser. After monitoring the behavior of this trojan, the following activities have been pointed out: (1) like a Facebook page, (2) add comments to a Facebook post, (3) share a wall post or photo album, (4) join a Facebook event or Facebook group, (5) post to the victim's wall, (6) add comments to photos, (7) send Facebook chat messages, (8) follow a Twitter user, and (9) inject third-party ads into the victim's Facebook page. In this case, all these behavioral patterns should be monitored to detect the malware.

Socware attacks. In this type of attack, an adversary creates malware items, called socware, in the form of applications, pages or events containing malicious links to be propagated in the OSNs [125]. This attack lures victims by offering false rewards to who will install/accept the socware [51]. Once users have installed the socware, it not only gets access to the user's personal information but also gains the ability to post on the victim's wall. As a consequence, users unknowingly end up sending socware messages or posts to their friends, essentially assisting the socware's viral spread [126].

A recent work studied the ecosystem enabling socware to propagate [67]. By analyzing data from the walls of approximately 3 million Facebook users over a period of five months, they discovered that socware propagation is supported by Facebook applications that are strategically collaborating with each other in large groups. This highlights that the propagation mechanism of socware items is faster than other usual items in OSNs.

Creepers attacks. Creepers are real users that are using the OSNs functionalities in inappropriate ways. As an example, they might send friend requests to many strangers or post spammy chain letters [139]. They could also temporary sell their accounts to advertisement services [108]. Sometimes, they collude with other users to send chain letters or propagate items for advertisement purposes [150]. In these types of social spam the damage is caused by real users, not automated programs. Therefore, creeper accounts are characterized by real users that make real profiles, and they might have both sparse friendship graphs, in case they send many friend requests to strangers, as well as a dense graphs.

Cyberbullying attacks. Cyberbullying has become quite common in online social networks. Attackers harass their victims (usually children and teenagers) by posting sexual remarks, threats, or repeated hurtful messages [51]. Also, attackers spread cruel rumors about the victims and share embarrassing pictures or videos of the victims in the network [119]. A recent study discovered that 12% of parents claim their children have been cyberbullied [106]. In addition, the majority of these attacks happened in OSNs sites, like Facebook. One of the catastrophic results of Cyberbullying is the case of Amanda Michelle Todd [119] and Rebecca Ann Sedwick [104], that committed suicide after being cyberbullied on Facebook.

Clickjacking attacks. In this kind of attacks, attackers trick users into clicking some items different from what they intended to click. Then, the attacker can manage the user's account by posting spam messages and performing likes on some items [35, 96]. Therefore, when users click on such items such as pictures, messages or links, the items automatically propagate in the network [105].

The analysis of all the above-mentioned attacks drives our feature selection, as detailed in the following section.

3.4 Features Description

In this section, we describe the set of features used in our two-phase risk assessment approach.

3.4.1 Group Identification Features

We recall that the aim of the first clustering is to group users for which similar behaviors are expected. At this purpose, group identification (GI) features should be those that are greatly discriminating, like age, gender, but also those that impact the possible users'

Attack name	GI	FR	MFR	FMFR	CR	SC	PR	PPS	LPS	CFR	PFR	OIR	LRLP	PRPP
Sybils (Dense friendship graph)	√				√	√	√	√	√	√	√	√	√	√
Sybils (Sparse friendship graph)	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Identity clone (Real accounts)					√	√	√	√	√	√	√	√	√	√
Compromised with real accounts								√	√				√	√
Socware (Real accounts)								√	√					
Creepers (Dense friendship graph)							√	√				√		√
Creepers (Sparse friendship graph)		√	√	√			√	√				√		√
Cyberbullying (Real accounts)					√	√	√			√	√	√		
Cyberbullying (Fake accounts)	√	√	√	√	√	√	√			√	√	√		
Clickjacking (Real accounts)								√	√					

Table 3.1: Mappings of behavioral features to well-known OSNs attacks

behaviors, like, education and nationality. In addition to these features, we have to take into account that even if in the real world, people with similar background usually behave in similar way, in an OSNs this might be impacted by the users' attitude towards online social networks that might be different even for similar users. For this reason, in addition to profile information (i.e., age, gender, education, nationality), in order to measure users' attitude in online socialization, GI features also include the following:

- Number of friends: social and popular users with a lot of friends or followers have different patterns from isolate users with few friends. For instance, popular users with high number of friends are more likely to accept requests from strangers in the OSNs, whereas isolated users do not [157, 53];
- Activity level: in general, active and popular users write a lot of messages, posts or comments and receive a lot of likes. In contrast, passive users do not send any information to others. We calculate this feature as the sum of the number of posts, likes and comments that users send to others from the first day of joining the community;
- Percentage of public profile items: the assumption is that users with all profile information public are more social. On the other hand, users without any public information either take care of their privacy too much or they are fake profiles.

3.4.2 Behavioral Features

In this section, we present the set of behavioral features (BF) on which our proposal is based. The design of this set is driven by the purpose of the system, that is, the detection of risky behaviors in OSNs. At this aim, we have taken into account the behavioral patterns identified in the study of OSNs attacks discussed in Section 3.3. This analysis brings us to the definition of 13 BFs, described in the following. As depicted in Table 3.1, this set is defined such to cover all attack patterns identified in Section 3.3. Another interesting quality of the identified set is that, as experiments in Section 3.6.4 will show, every single feature is influent and relevant in the risk assessment process.

Before presenting the proposed behavioral features, we introduce two measures, used in features computation. The first is the *user longevity*, which is measured as the number of days since the user joined the OSNs. The second one is the *item longevity*, which is measured as the number of days since an item has been uploaded in the OSNs.

Friendship Rate (FR). In general, having a high number of friends is not a sign of risky behaviors. However, literature shows that attackers are more aggressive in establishing new friendships than normal users [157]. Indeed, it has been shown that attackers try to infiltrate the target OSNs and befriend with many real accounts, by sending friend requests to popular users that are more likely to accept requests from strangers [15, 53]. Such infiltration is required because isolated attackers cannot directly interact with most users in the OSNs or promote content [15]. In order to catch this behavior, for a given target user $\overline{u}$, we are interested in tracing the number of his/her friends based on his/her longevity in the network. We assume that if a user has a lot of friends in few days after joining the community, this can be indicative of an anomalous behavior. We calculated this feature as follows:

$$FR(\overline{u}) = \frac{|Friends(\overline{u})|}{UserLongevity(\overline{u})}$$

Where $|Friends(\overline{u})|$ is the number of friends of user $\overline{u}$ and $UserLongevity(\overline{u})$ is the number of days since user $\overline{u}$ joins the community. We also have to note that in some attacks, attackers show significantly different number of friends [53], which might result in a not meaningful FR measure. However, the first phase of the proposed approach groups users also based on number of friends. As such we expect that in the same group, normal users show a similar FR value.

Mutual Friendship Rate (MFR). This feature computes the average number of mutual friends that a target user $\overline{u}$ has with all his/her friends in the network. We select this feature since in some attacks (e.g., sybil attacks), attackers send friendship invitations to a lot of strangers. This results in sparse friendship graphs, without mutual friends [157]. Whereas, in general, normal users have some mutual friends with their friends. This feature is computed as follows:

$$MFR(\overline{u}) = \frac{\sum_{\breve{u} \in Friends(\overline{u})} |MutualFriends(\overline{u}, \breve{u})|}{|Friends(\overline{u})|}$$

Where $|MutualFriends(\overline{u}, \breve{u})|$ is the number of mutual friends of $\overline{u}$ and $\breve{u}$. We also have to note that recently some attackers have became smarter as they first try to have some mutual friends with their victims before sending friendship invitations [25]. For example, as described in [16], an attacker and the victim in Facebook have more than eleven mutual friends. However, this feature is still relevant for some attacks, and together with other features can help to have a comprehensive risk estimation.

Friend Mutual Friend Ratio (FMFR). In addition to the previous features, we are interested to catch those users with high number of friends, but a lower average number of mutual friends, as this could indicate a risky conduct. This is the case, as an example, of fake accounts, where attackers have sparse communities and they send a lot of friendship invitations to strangers. This feature is computes as follows:

$$FMFR(\overline{u}) = \frac{|FR(\overline{u})|}{MFR(\overline{u})}$$

Comment Rate (CR). Another common activity in an OSNs is commenting on posts. Given the relevance of comments and posts, we identify a set of features to measure the user activities related to them. In particular, for each target user $\overline{u}$, CR measures the number of comments written by $\overline{u}$ w.r.t. his/her longevity. This comes from the assumption that normal users show regular patterns in writing comments. On the contrary, those users that flood the network with comments as soon as they join might have to be considered risky. As an example, in cyberbullying attacks, attackers harass their victims by sending repeated hurtful messages [104], [106]. CR is computed as follows:

$$CR(\overline{u}) = \frac{|CommentsBy(\overline{u})|}{UserLongevity(\overline{u})}$$

Where $|CommentsBy(\overline{u})|$ is the number of comments that user $\overline{u}$ sent to others.

Started Comments (SC). This feature complements the previous one. More precisely, it helps to catch active risky behaviors, where users maliciously start several new comments. As an example, in cyberbullying, most of the time attackers write massages or comments for his/her victims [104], [106]. This feature is computed as follows:

$$SC(\overline{u}) = \frac{|CommentsStartedBy(\overline{u})|}{UserLongevity(\overline{u})}$$

Where $|CommentsStartedBy(\overline{u})|$ is the number of comments where user $\overline{u}$ was the starter.

Post Rate (PR). This feature helps to detect users that flood the networks with a huge number of posts as soon as they join the OSNs. Indeed, since the goal of some attacks is to propagate items to gain more popularity, attackers, in general, are more active in posting items than regular users. For example, it has been shown that in social bots attackers with higher activity levels achieve significantly more popularity than less active socialbots [53]. To catch this risky behavior we measure the number of posts sent by a target user $\overline{u}$ w.r.t. his/her longevity, as follows:

$$PR(\overline{u}) = \frac{|PostsBy(\overline{u})|}{UserLongevity(\overline{u})}$$

Where $|PostsBy(\overline{u})|$ is the number of posts that user $\overline{u}$ sent to his/her friends. It is relevant to note that, if this helps in detecting some kinds of anomalous behaviors, it does not help in case of attacks where legitimate users are compromised so as to unknowingly share items. In general, these compromised accounts have long longevity that might imply normal rate of posts per longevity. This brings us to introduce further behavioral features.

Post Propagation Speed (PPS). In addition to previous features measuring the general users behavior w.r.t. comments/posts, we are also interested in tracing how single posts produced by a target user are consumed in the network. Usually, fake posts (e.g., posts created by attackers or victims) show an higher propagation speed than regular posts. In general, attackers and victims have a massive propagation of information in the network in a short period of time, which implies a propagation speed of malware items higher than the usual ones [50]. Given a target user $\overline{u}$, to measure the propagation speed of a given post, we count the number of likes, posts and comments received by it, and normalize the computed measure by the post longevity, as follows:

$$PPS(\overline{u}) = \frac{\sum_{\forall p \in PostsBy(\overline{u})} \frac{(|LikesOn(p)| + |PostsOn(p)|)}{ItemLongevity(p)}}{|PostsBy(\overline{u})|}$$

Where $|LikesOn(p)|$ and $|PostsOn(p)|$ are the number of likes and posts that item p received, whereas $ItemLongevity(p)$ is the number of days since item p has been created by user $\overline{u}$.

Like Propagation Speed (LPS). It might be considered a risky behavior the activity of assigning likes to items that show an high propagation speed. For example, in clickjacking, when legitimate users click on a malicious message/link, this automatically and virally propagates onto their accounts. Therefore, attackers can manipulate friends of the victim driving them in performing likes on malicious messages/links [50].

At this purpose, this feature aims at measuring the average propagation speed of items on which the target user $\overline{u}$ has put a like. Thus, we measure, for each like x, the number of likes and posts/comments received by x w.r.t. its longevity. After that, we normalize this measure by dividing by the number of items to which the target user has assigned a like, as follows:

$$LPS(\overline{u}) =$$

$$\frac{\sum_{\forall i \in LikesBy(\overline{u})} \frac{(|LikesOn(i)| + |PostsOn(i)|)}{ItemLongevity(i)}}{|LikesBy(\overline{u})|}$$

Where $|LikesBy(\overline{u})|$ is the number of items on which user $\overline{u}$ has performed a like.

Comment Feedback Ratio (CFR). This feature aims at detecting users that receive few likes compared to the number of comments they write. This is motivated by the fact that, in some attacks, like cyberbulling and sybil attacks, attackers are more aggressive in sending messages [106]. We measure the ratio between the number of comments that a target user $\overline{u}$ sends to others and the average number of likes that he/she receives on his/her comments. This is normalized by the user longevity, as follows:

$$CFR(\overline{u}) = \frac{\frac{|CommentsBy(\overline{u})|}{AvgFeedbackOnComment(\overline{u})}}{UserLongevity(\overline{u})}$$

where $AvgFeedbackOnComment(\overline{u})$ is the average number of likes received by comments of user $\overline{u}$. This is computed as follows:

$$AvgFeedbackOnComment(\overline{u}) =$$

$$\sum_{c \in CommentsBy(\overline{u})} \frac{|LikesOn(c)|}{|CommentsBy(\overline{u})|}$$

where $|LikesOn(c)|$ is the number of likes on comment c.

Post Feedback Ratio (PFR). If a user has a low longevity and posts a lot of items by receiving few likes and comments on them, it can be indicative of risky behaviors. By this feature, we just consider the number of likes and comments on posts, since, in most types of attacks, attackers try to post items either in a direct way or indirectly by compromising legitimate users in the network. In both cases, the attacker or victims of attacks propagate a lot of items in the network without receiving any feedback on them. PFR is defined as follows:

$$PFR(\overline{u}) = \frac{\frac{|PostsBy(\overline{u})|}{AvgFeedbackOnPost(\overline{u})}}{UserLongevity(\overline{u})}$$

where $AvgFeedbackOnPost(\overline{u})$ is the average number of likes and comments received by posts of user $\overline{u}$. This is computed as follows:

$$AvgFeedbackOnPost(\overline{u}) =$$

$$\sum_{p \in PostsBy(\overline{u})} \frac{|LikesOn(p)| + |CommentsOn(p)|}{|PostsBy(\overline{u})|}$$

where $|LikesOn(p)|$ is the number of likes and $|CommentsOn(p)|$ is the number of comments on post p.

Out In Ratio (OIR). In general, if a user propagates a lot of information in the network (e.g., comments, posts) and receive few feedback (e.g., likes, comments on posts) it can be indicative of an anomalous behavior. For instance, in cyberbulling attacks, attackers send a lot of information without receiving any feedback on them. Thus, given a target user $\overline{u}$, this feature measures the ratio between the number of posts, likes and comments generated by $\overline{u}$ and the number of likes on comments/posts plus the number of comments on posts he/she received. This ratio is normalized by the user longevity as follows:

$$OIR(\overline{u}) = \frac{\frac{|CommentsBy(\overline{u})| + |PostsBy(\overline{u})| + |LikesBy(\overline{u})|}{FeedbackOn(\overline{u})}}{UserLongevity(\overline{u})}$$

where $FeedbackOn(\overline{u})$ is the number of likes on comments plus the number of comments and likes on posts that user $\overline{u}$ received:

$$FeedbackOn(\overline{u}) = \sum_{c \in CommentsBy(\overline{u})} |LikesOn(c)| + \sum_{p \in PostsBy(\overline{u})} (|LikesOn(p)| + |CommentsOn(p)|)$$

Like Rate Like Propagation (LRLP). This feature can help in detecting creeper, clickjacking and socware, where attackers try to propagate a high number of items in the network with a high propagation speed. We recall that, in all these attacks, the victim is maliciously forced to unknowingly perform likes on posts. In order to detect these attacks, we believe that feature LPS might be not enough. Because this feature just considers the average of the propagation speed of items on which a user performs likes, without considering the number of those items. For this reason, LRLP additionally considers the number of likes that a user performs on items in the network. This helps to capture users that likes an high number of items with high propagation speed. LRLP is computed as follows:

$$LRLP(\overline{u}) = \frac{\sum_{\forall i \in ItemsLikedBy(\overline{u})} (|LikesOn(i)| + |PostsOn(i)|)}{ItemLongevity(i)} * \frac{|LikesBy(\overline{u})|}{UserLongevity(\overline{u})}$$

Post Rate Post Propagation (PRPP). By this feature, we are interested to model the number of posts generated by a target user and the propagation speed of these posts. Because in some kinds of attacks, attackers have a high number of posts and these posts have high propagation speed. For example, in identity clone attacks, the attacker could run spam attacks by requesting the fake identities to propagate malicious posts and messages to other users in the network. Therefore, we calculate the total propagation speed of posts and multiply it by the post rate per longevity, as follows:

$$PRPP(\overline{u}) =$$

$$\frac{\sum_{\forall i \in PostedBy(\overline{u})}(|LikesOn(i)| + |PostsOn(i)|)}{ItemLongevity(i)} *$$

$$\frac{|PostsBy(\overline{u})|}{UserLongevity(\overline{u})}$$

3.5 Two-phases clustering

We recall that our risk assessment approach is composed of two phases. The first aiming at organizing users according to group identification features, the second exploiting behavioral features. Regardless of the features taken into account, in both these phases we make use of the same clustering algorithm. Clustering algorithms can be classified into two main types [20]: hard and soft clustering.

Hard clustering techniques (e.g., k-means) compute the best cluster in a deterministic way, that is, each item is assigned to a unique cluster. In contrast, soft clustering (i.e., probabilistic-based clustering) computes, for each item and every available cluster, the membership probability. We recall that the proposed risk assessment approach is based on the idea of estimating the user's risk on the basis of how much his/her behavior deviates from the one of other users. Thus, given a target user we are more interested in having his/her cluster membership probability rather than just knowing the cluster to which he/she should belong to. Therefore, we adopt probabilistic-based clustering.

In general, the key goal of a probabilistic-based clustering techniques is to find the most likely cluster for each given data item. More precisely, to determine K clusters over a set of data items, we have to define K probability distributions, each one representing the likelihoods of data items to belong to a given cluster. In our setting, in the first phase, membership probabilities are computed based on the values of GI features. Then, based on these likelihoods, each user is associated with the group that better fits his/her GI features that is, the one with the highest membership probability. In the second phase, users of the same group are further clustered according to their behavioral features. As it will be discussed in Section 3.5.2, the new membership probabilities are used to compute the final user risk scores.

In order to introduce the user risk score definition, we need to more formally illustrate the probabilistic-based clustering technique, which is described in the next section.

Attack name	F-measure		Detection rate		False alarm rate	
	Two-Phase	One-Phase	Two-Phase	One-Phase	Two-Phase	One-Phase
RF	0.807	0.744	0.92	0.726	0.0216	0.0257
SR	0.804	0.729	0.937	0.676	0.0230	0.0424
MSR	0.801	0.741	0.929	0.696	0.025	0.043

Table 3.2: Comparison of two-phase vs. one-phase risk assessment

3.5.1 Probability-based clustering

Membership probability is computed based on the values contained in the user features vector, that is, GI features in the first phase and behavioral features in the second one.

To perform probabilistic clustering, we make use of the most popular algorithm, Expectation-Maximization (EM) [20], which uses probability estimation via an iterative procedure. The Expectation-Maximization algorithm iteratively learns and optimizes the parameters of the clustering model, that is, the mean and covariance matrix of each feature value for each cluster, and the fraction of users belonging to each cluster. Given the learned parameters, the algorithm assigns with each cluster a set of features values that most likely represent users belonging to that cluster. In the first iteration of the EM algorithm, the parameters are initialized by random positive values. After that, for a number of iterations, the membership probability for each user in each cluster is calculated in the Expectation-step. Then, for each cluster the parameters are optimized based on the current membership probability in the Maximization-step.

In the following, we describe probability-based clustering for a generic feature vector of a user $\vec{u}$. This vector will contain the values of the user's GI features in the first phase, while it will contain the values of the BFs in the second phase. Let N be the set of users in the OSNs, with a set of features vectors $\vec{u}$ in cluster l. The probability of membership or 'weight'of target user $u \in N$, is defined as [20]:

$$w_l(\vec{u}) = \frac{w_l.p_l(\vec{u}|\theta_l)}{\sum\limits_{i=1}^{K} w_i.p_i(\vec{u}|\theta_i)}$$

where: w_l is a weight computed as $w_l = \frac{|N_l|}{|N|}$, with N_l denoting the set of users belonging to the l^{th} cluster, where $\sum\limits_{l=1}^{K} w_l = 1$; function $p_l(\vec{u}|\theta_l)$ is the component density function modeling the users of the l^{th} cluster, where $\theta_l = \{\vec{\mu_l}, \Sigma_l\}$ represents the parameters for l^{th} distribution, that is, the mean and the covariance.

Finally, the probability function evaluated over a target user u is:

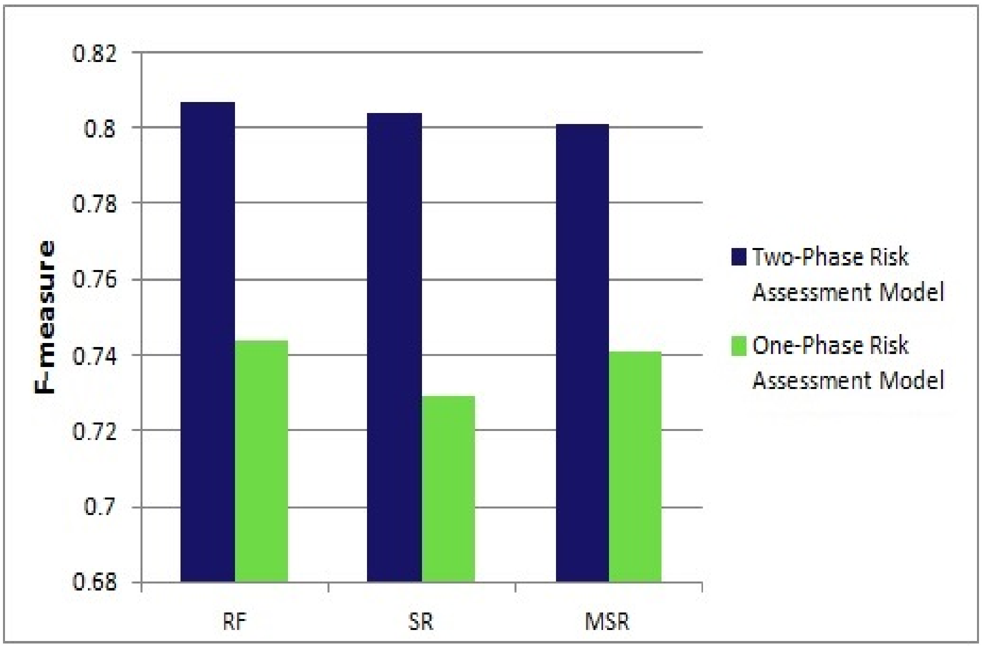

Figure 3.2: Comparison of F-measure for two-phase vs. one-phase risk assessment

$$p(\vec{u}|\Theta) = \sum_{l=1}^{K} w_l.p_l(\vec{u}|\theta_l)$$

where, we denote with Θ the set of model parameters, that is, $\Theta = \{w_l, \vec{\mu}_l, \Sigma_l\}$, $l = 1, \ldots, K$. Since user features values are independent, the probability function for all users in N can be defined as:

$$p(N|\Theta) = \prod_{i=1}^{|N|} p(\vec{u}_i|\Theta) = \prod_{i=1}^{|N|} (\sum_{l=1}^{K} w_l.p_l(\vec{u}_i|\theta_l))$$

Based on the above formula, if we know values of the model parameters Θ, then we would also know the membership probabilities for all users in N, and thus the generated clusters. For this reason, in the probabilistic-based clustering, the goal is to estimate Θ that maximizes the log-likelihood, and EM algorithm [98] generates the maximum likelihood estimation for Θ. This is quantified by the log-likelihood of all the users:

$$L(\Theta) = \sum_{\vec{u} \in N} log(\sum_{l=1}^{K} w_l.p_l(\vec{u}|\vec{\mu}_l, \Sigma_l))$$

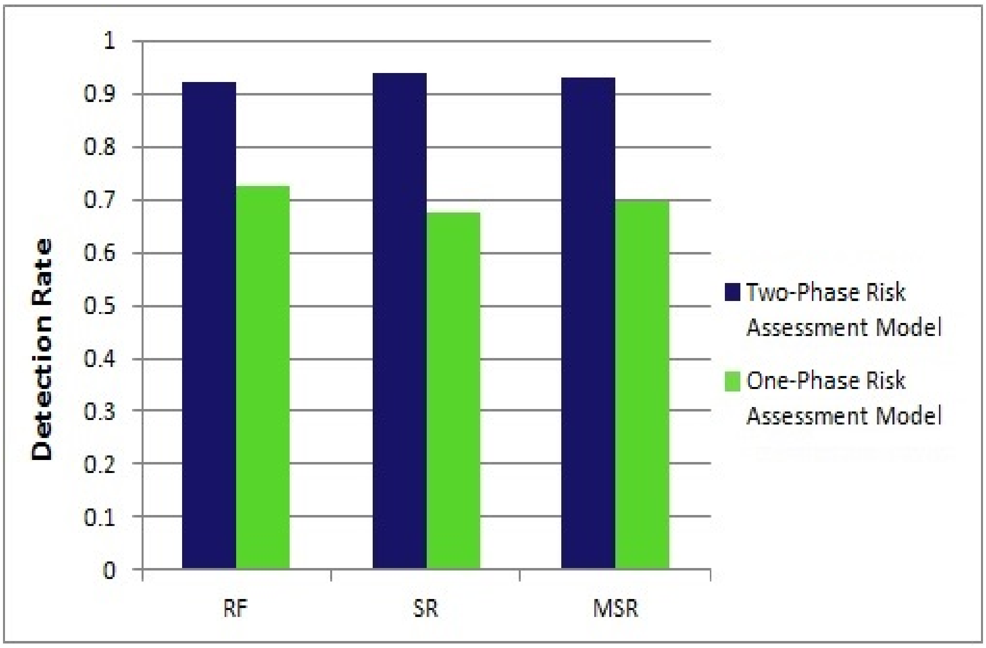

Figure 3.3: Comparison of detection rate for two-phase vs. one-phase risk assessment

According to EM, initial values are assigned to each parameters in Θ, which are then iteratively updated. We denote with Θ^j the parameter values set at iteration j. The sequence of Θ-values which is then the Likelihood $L(\Theta)$ is non-decreasing at each iteration [39, 20].

After the initialization phase, there are two steps that are iterated until the parameters converge to a local maximum [110].

Expectation-step Using the Θ^j values, this phase calculates cluster probabilities and computes the membership probability of $\vec{u}$ in each cluster.

$$w_l^j(\vec{u}) = \frac{w_l^j . p_l(\vec{u}|\theta_l^j)}{\sum\limits_{i=1}^{K} w_i^j . p_i(\vec{u}|\theta_i^j)}$$

Maximization-step In this phase, the algorithm updates parameter Θ [39] to maximize the likelihood of the data[101] as follows:

$$w_l^{j+1} = \sum_{\vec{u} \in N} w_l^j(\vec{u})$$

$$\vec{\mu}_l^{j+1} = \frac{\sum\limits_{\vec{u} \in N} w_l^j(\vec{u}).\vec{u}}{\sum\limits_{\vec{u} \in N} w_l^j(\vec{u})}$$

$$\Sigma_l^{j+1} = \frac{\sum\limits_{\vec{u} \in N} w_l^j(\vec{u})(\vec{u} - \vec{\mu}_l^{j+1})(\vec{u} - \vec{\mu}_l^{j+1})^T}{\sum\limits_{\vec{u} \in N} w_l^j(\vec{u})}$$

The expectation and maximization phases are iterated until $|L(\Theta^j) - L(\Theta^{j+1})| \leq \varepsilon$, where $\varepsilon > 0$ is the stopping tolerance.

3.5.2 User Risk Score

As discussed throughout the chapter, the idea is to consider more risky those users that diverge from normal behaviors. These deviations are actually captured by the membership probabilities computed in the second clustering phase. More precisely, an high membership probability value implies that the target user fits well one of the behaviors emerged from the group he/she belongs to. In contrast, a low membership probability value refers to a user whose behavior diverges from those that are considered normal for his/her group. Based on this, the risk score associated with a target user u is defined as the inverse of the highest among u's membership probability values resulting by the second clustering phase.

Definition 1 (***User Risk Score (RS)***) *Let N be the set of users in the OSNs, and let $N_g \subset N$ be a set of users that belong to the same cluster g, resulted from the probabilistic-based clustering computed in the first phase over the GI features' values of user in N. Let $PB(N_g)$ be the probability-based clustering algorithm that takes as input the set of users in N_g and, based on their BFs, returns, for each user $u \in N_g$, the highest membership probability, denoted as $\mathcal{PCL}(u)$. Given a target user $u \in N_g$, the associated risk score $RS(u) \in [0,1]$ is defined as:*

$$RS(u) = 1 - \mathcal{PCL}(u) \tag{3.1}$$

3.6 Experiments

In this section, we illustrate the experiments carried out to show the effectiveness of the proposed two-phase risk assessment.

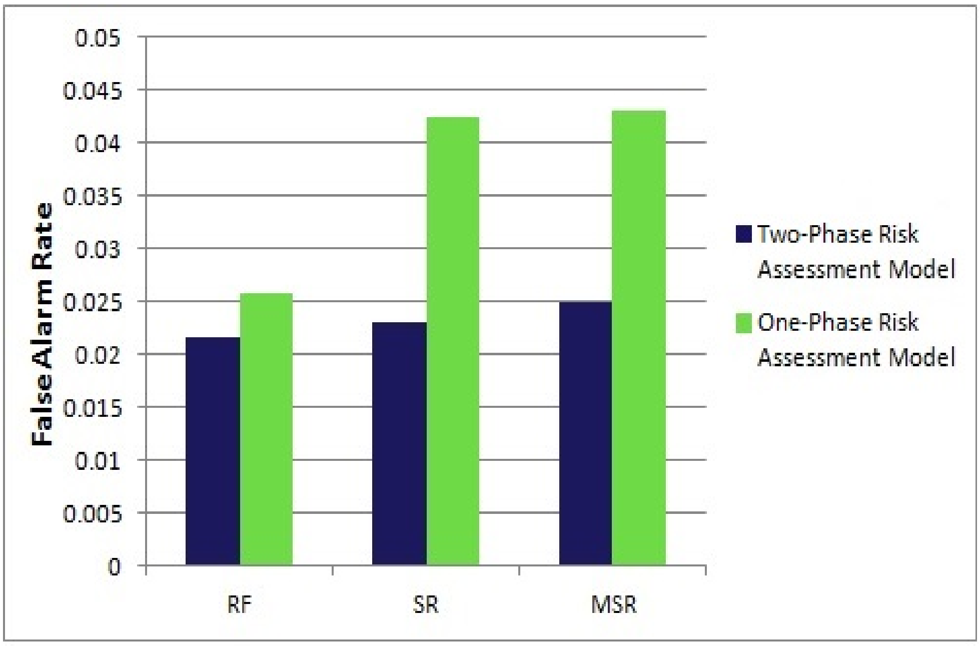

Figure 3.4: Comparison of false alarm rate for two-phase vs. one-phase risk assessment

3.6.1 Facebook Dataset

To perform the experiments on real data, we used the Facebook dataset collected and used in [3]. This dataset was collected using a Facebook application that gathered friendship links and profiles of users who launched the application. This application has been used by 75 Facebook users, which forms a first dataset called FB_{75}. This contains all posts, likes and comments written by the 75 users, that is: 33,728 friend links, 185,644 posts, 2,187 likes and 2,262 comments. Using these 75 users as seeds, the Facebook application defined in [3] crawled additional information about seeds' friendships. In particular, once a friend of a friend is found, the app queries Facebook for its mutual friends/profile information. After having deleted those profiles with too many missing features, we obtain a second dataset, called FB_{13000}, which includes 13,000 user profiles, plus the 75 seed users, with a total of about 461,700 friend links, 6,150,892 likes and 1,742,709 comments.

The Facebook app also collected the user public profiles during the crawl. Each profile consists of optional information provided by the users themselves, such as education, gender and geographic information. Since users are allowed to mark their profiles as private, we were not able to download profile information for all users. In total, around 7,000 users of FB_{13000} have more than 75% of their profile information.

3.6.2 Two-phase vs. one-phase risk assessment

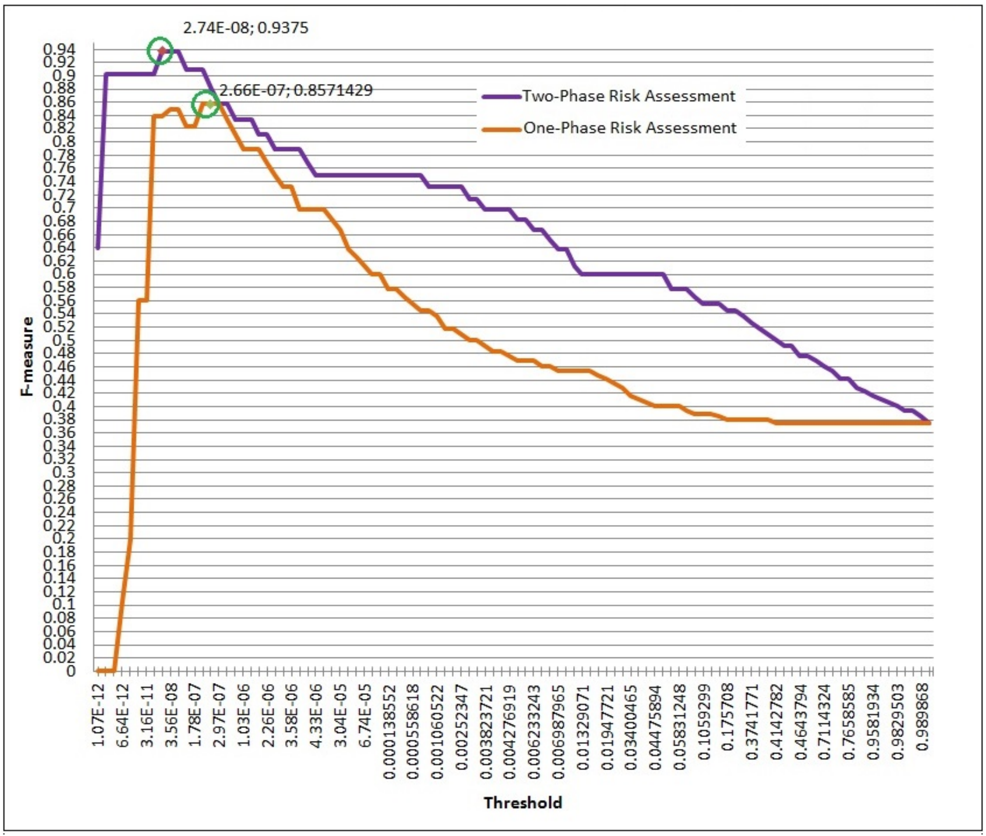

Figure 3.5: F-measure for social bots or sybils (dense friendship graph)

The first and second experiments aim at verifying the underlying idea of extracting model behaviors from user groups. At this purpose, we compare the two-phase risk assessment approach presented in this chapter against a risk assessment where we simply consider the behavioral features presented in Section 3.4 without grouping users before. We refer to that model as one-phase risk assessment.

Similarly to the two-phase approach, the risk score of a target user u returned by the one-phase risk assessment is computed as $RS(u)=$ 1- $\mathcal{PCL}(u)$, where $\mathcal{PCL}(u)$ is defined as the highest among the membership probability values associated with u and returned by the one-phase clustering.

In order to compare the two-phase risk assessment against the one-phase, we have injected into the FB dataset a set of fake users. Then, we measured the effectiveness of the two approaches in detecting them as risky based on their risk score. More precisely, we say that a user u is risky if $\mathcal{PCL}(u)$ is less than a given threshold τ. By setting the threshold value, we can determine how much the system has to be accurate in labeling users as risky.

As an example, setting a threshold value near to 1 implies to consider risky those users whose highest membership probability is near 1, that is, those users for which there exists

Attack name	Two Phase Risk Assessment		One Phase Risk Assessment	
	F-measure	Threshold	F-measure	Threshold
Socialbots or sybils (sparse friendship graph)	0.9375	5.24E-07	0.8666	2.40E-10
Socialbots or sybils (dense friendship graph)	0.937	2.74E-08	0.857	2.66E-07
Identity clone	0.88	1.24E-07	0.81	1.24E-07
Compromised accounts	0.833	3.82E-05	0.580	9.13E-06
Socware	0.731	4.70E-05	0.558	4.70E-05
Creepers (dense friendship graph)	0.812	5.57E-06	0.667	5.57E-06
Creepers (sparse friendship graph)	0.86	1.60E-08	0.774	1.60E-08
Cyberbullying (real profiles)	0.857	3.82E-05	0.619	0.0002
Cyberbullying (fake profiles)	0.882	1.94E-09	0.812	1.65E-07
Clickjacking	0.769	3.34E-05	0.560	0.0004

Table 3.3: The best F-measure and corresponding threshold value for each type of attack

a cluster that almost fits their features. In contrast, when the threshold τ is near 0, it means that the system judges as risky only those users for which the highest membership probability is close to 0, that is, those users for which it does not exist a cluster fitting their features. In general, we expect that threshold value is determined according to user preferences on having a more conservative approach or relaxed one. In this experiment, we set τ based on data distribution. In particular, we compute the membership probability of all users. Analyzing the distribution of the computed values, we noted that most of the users have a membership probability, near to 1, and just a small portion of users (less than 5%) have a membership probability near to 0. Then, the selected threshold (e.g., 2.74E-08) is the one identifying this small portion of users.

Another relevant parameter to be initialized is the number of cluster K. We have to note that several approaches to estimate the best number of clusters K have been proposed in the literature [127]. There is a general agreement that identifying the optimal K is a problem related to the nature of the dataset. Since the problem depends on data, we pre-process the available dataset to estimate the best K for each phase. In particular, we run the clustering phase with different K values and we measured the quality of the obtained clusters as the ratio of users that belong to each of them. Based on this pre-processing, we set K equal to 7 in the first phase, whereas, in the second phase, it is on average between 7-12 for each group of users.[3]

The effectiveness of our risk estimation approach is measured according to F-measure, detection rate and false alarm rate (false positive). The F-measure is the weighted harmonic mean of two measures, namely, precision and detection rate. The precision is the number of correctly detected risky users divided by the number of injected fake users and normal users that are detected as risky. Detection rate is the ratio between the number of correctly detected risky users and the total number of injected fake users. Usually, precision and detection rate scores are not discussed in isolation. Instead, both are combined into the

[3] We are aware that an approach based on data pre-processing is not suitable for big data scenarios, like OSNs. However, we expect that this pre-processing could be run over a sample of OSNs data so as to determine the right K.

F-measure as a single metric. The F-measure reaches its best value at 1 and worst at 0. False alarm (false positive) rate is a ratio between the number of normal users that are misclassified as risky and the total number of normal users.

Fake users to be injected in the real Facebook dataset have been generated following three models. The first model represents risky users with Randomized Features (RF), that is, users where values of GI features and BFs are randomly set.[4] The second one includes Smart Risky users (SR), defined with legitimate values for profile features (i.e., age, gender, education, nationality), and randomized values for BFs. In particular, the legitimate values are generated by selecting and duplicating the whole profile of some of the users in the FB dataset.

The last model considers More Smarter risky Users (MSU), where fake users are created with legitimate values for profile and friendship features, and randomized value for the remaining behavioral features. The friendship features are features related to friendship behaviors and include: FR, MFR, and FMFR selected from BFs and number of friends selected from GIs.

We run these two experiments on FB_{13000}. We repeated the experiments five times, where in each run we created 90 new fake users for each model. Moreover, we set the threshold τ as the one identifying the small portion of users based on the distribution of the $\mathcal{PCL}$ values of all users. The final result is reported in Table 3.2.

Figures 3.2, 3.3, and 3.4, report the F-measure value, the detection rate, and false alarm rate, respectively, for the three types of considered fake users. As shown in Figures 3.2, 3.3 and 3.4, two-phase risk assessment outperforms the one-phase, based on F-measure, detection rate, and false positive measure.

3.6.3 Risk assessment vs. Attacks

This experiment aims at showing how our risk scores can be used to detect very risky users. At this purpose, we have injected into the real dataset fake users created so as to simulate behavioral patterns of each type of attacks described in Section 3.3. We run this experiment using the smallest dataset, that is FB_{75}, since users in this dataset have also post information, thus it was possible to compute all features.

From FB_{75}, we generate 10 different test datasets for 10 different types of attack (e.g., sybils (dense graph), sybils (sparse graph), socware, etc). For each one, we inject 55 normal users and 15 fake users. In all the 10 datasets, we generate normal users so that each one of their feature value is randomly selected within the corresponding standard deviation in FB_{75}. In contrast, fake users are customized based on the considered attack. In particular, in the 10 datesets fake users are generated so as to have random values for those features that are related to the corresponding attack (see Table 3.1), whereas the remaining are initialized similar to normal users. For example, based on the behavioral pattern of creepers with dense friendship graphs (see Section 3.3), the value of BFs such as PR, PPS, PRPP and OIR (see Table 3.1) can be anomalous, whereas other features will

[4] Each feature value is set by randomly selecting a value from the corresponding domain, which has been computed based on values in the real dataset.

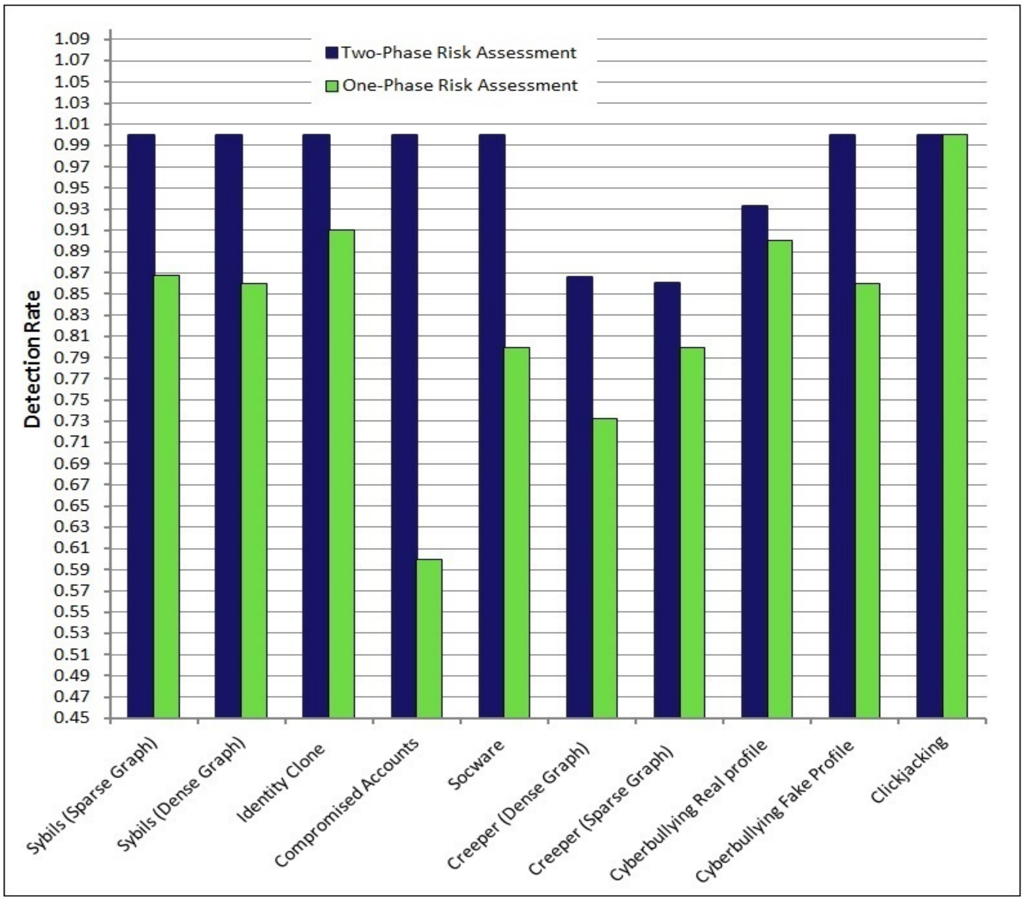

Figure 3.6: Detection rate for all types of attacks

be similar to normal users. Therefore, we set the value of these anomalous features in the test set as randomized values outside the corresponding standard deviation in FB_{75}. We then see if these anomalous users can be detected using our model.

Experiments have been carried out both for the two-phase risk assessment as well as the one-phase. Moreover, we analyzed the effectiveness by considering several values for the threshold τ. As such, for each type of attack, we obtained different F-measure, detection rate and false positive rate. As an example, Figure 3.5 shows the F-measure for social bots or sybils (with dense friendship graph) attacks, where the F-measure value is represented in the Y-axis, and the threshold value in the X-axis.

Then, from F-measure curves of each type of attack we selected the best F-measure values. These values are reported in Table 3.3. By using these selected threshold values, we run further experiments to evaluate the detection rate and false positive rate for each type of attack. In particular, Figure 3.6 shows the results for detection rate, whereas Figure 3.7 shows the false positive rate. As we can see the two-phase risk assessment outperforms the one-phase.

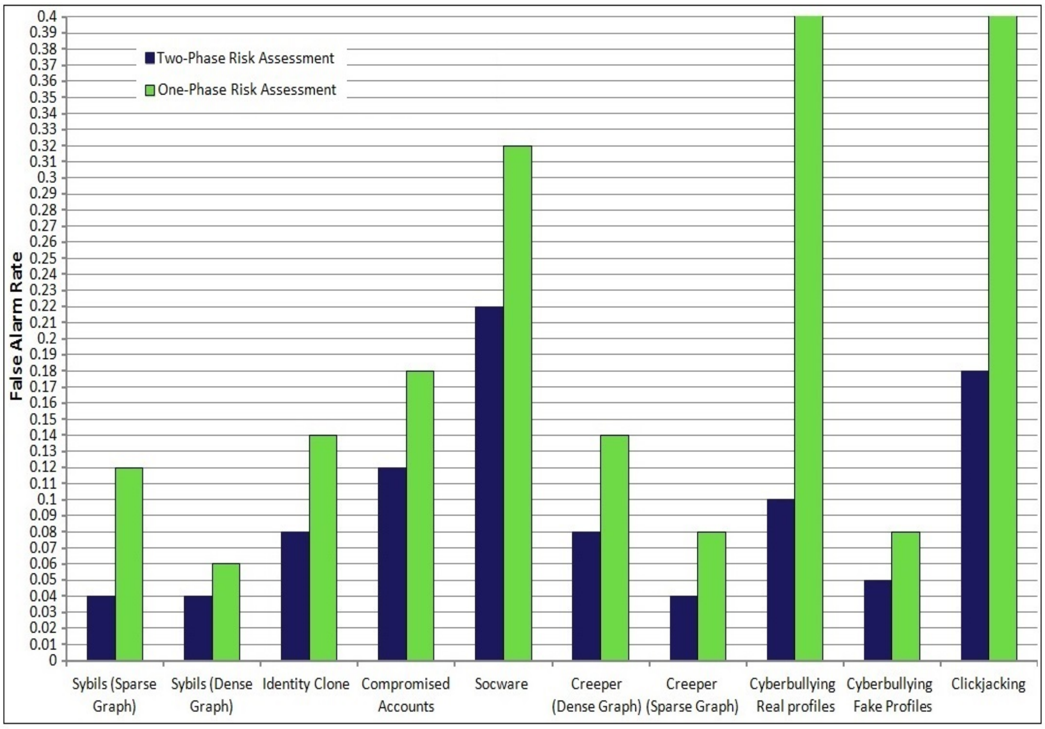

Figure 3.7: False Alarm Rate for all types of attacks

3.6.4 Testing the Influence of BFs

The aim of this experiment is to test the relevance of each of the identified BFs. Toward this goal, we generate a new dataset, called $FB_{75\,Attackers}$, by injecting into FB_{75} a set of fake users. These fake users are generated following behavioral patterns of 10 different types of attack. In particular, for each attack we inject 15 fake users whose features' values follow the corresponding attack behavior. Then, we run the proposed two-phase risk assessment model several times, by removing in every run one of the 13 behavioral features. In every run, we compute the corresponding F-measure to see the accuracy of the risk assessment model without the dropped behavioral feature. We need to mention that we set up the threshold by analyzing the distribution of membership probability $\mathcal{PCL}$ of all users. Since the $\mathcal{PCL}$ value for most of the users is near to 1 and just for a small portion of users (less than 5%) is near 0, we set the threshold as 0.00001 thus to select a small portion of users. As we can see in Figure 3.8, the obtained F-measure shows that in every execution we have a significant (i.e., around 0.2-0.6) loss of accuracy.

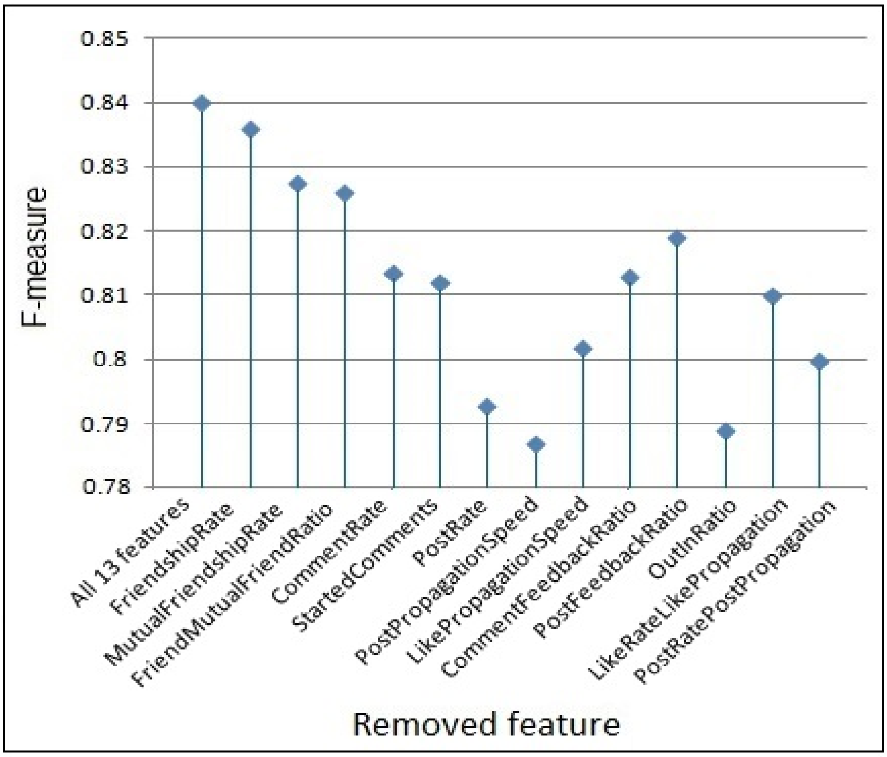

Figure 3.8: The best F-measure by removing one BF at a time

Chapter 4

Graph Based Local Risk Estimation in Large Scale OSNs

4.1 Introduction

A huge amount of personal information of users in OSN encouraged attackers to develop different techniques to exploit OSNs infrastructures for malicious purposes in order to misuse the personal information of users. The most notable types of attacks in OSNs are sybils/-socialbots, identity cloned attacks, socwares, compromised account attacks, cyberbulling attacks, and creepers [139, 51]. To cope with emerging security and privacy concerns, research community has started to deeply investigate and propose mechanisms for safer and trustworthy OSNs. Orthogonal to all these efforts, we believe there is the need of measures helping user to judge his/her contacts. This brings us to investigate a risk estimation measure by considering the topology of user's social graph. In doing this, we leverage on relevant results achieved on graph-based outlier detection. In particular, the proposed risk measure comes from the observation done in [5], where it has been highlighted that a user of a given network is anomalous if his/her subgraph significantly differs from those of other users. More precisely, in this chapter, we adapt the definitions proposed in [74] so as to have an unsupervised graph-based outlier detection methods tailored over features meaningful for the detection of risk behaviors in OSNs. The overall purpose is to obtain a local risk factor measure that helps users to detect potential attackers among their contacts. The obtained results show that these topological features can indeed be used to define the risk of direct contacts in large scale OSNs.

The remainder of this chapter is organized as follows. Section 4.2 introduces the overall idea underlying our approach, whereas Section 4.3 provides a summary of the considered graph based features. Section 4.4 illustrates our graph based risk measure. Finally, experiments are presented in Section 4.5.

4.2 Overall Approach

The proposing risk estimation measure comes from the observation done in [5], where it has been highlighted that a user of a given network is anomalous if his/her subgraph significantly differs from those of other users. In our proposal, based on the behavior of malicious users in OSNs, we consider not only the direct subgraph of a target user, but also the direct subgraph of his/her direct contacts (two step subgraph). Therefore, the subgraph of a user is a collection of all of his/her direct contacts, the direct contacts of his/her direct contacts as well, and all the connections among them (see Section 4.3.2 for a more detailed discussion).

We have then to define a measure for comparing different subgraphs. We exploit the lesson learnt from an outlier detection technique presented in [22], saying that density can represent an interesting measure to catch anomalous nodes. In particular, by comparing the local density of a node to the local densities of its nearest users, one can identify nodes that have a substantially lower density than their nearest users, considered to be outliers. The local density of a user u is computed based on a distance measure. In particular, we first compute the Euclidean distance between u and all the users in the network. Then, we rank the results and we select the distance of the user at the *k-th* position. Thus, local density of u is defined as the inverse of this distance. More precisely, given the purpose of this chapter, i.e., risk estimation, we compute the Euclidean distances based on a set of topological features that we believe are meaningful for the detection of risky behaviors. These features have been selected based on a review of current well-known OSNs attacks (see discussion in Section 4.3 for more details).

According to [74], we can exploit the local density of u, to determine its *Divergency Factor*, that is, how much u is different from the rest of the network. More precisely, how much u's density is different from the ones of users that are topologically similar to u. These users are defined as *k-nearest users.*[1]

Literature offers different ways to compute the divergency factor, see for instance [22, 74]. In this chapter, we exploit the method proposed in [74], called INFLO. The benefit of this method is that it also considers the symmetric neighborhood relationship in computing the k-nearest users. This means that a user z is into the k-nearest users of a user y if the Euclidean distance between z and y is less than the one at the k-th position in the ranking and there is symmetry in their k-nearest users, that is, y is into the k-nearest users of z as well.

Once computed the divergency factors for all users in the network, a target user u will be able to assign a *Local Risk Factor* to each node y in his/her direct contacts list, based on how much y's divergency factor is different from the divergency factor of the other u's direct contacts. Thus, u can understand how much his/her contacts are risky by ranking

[1] Given a k, the *k-nearest users* are determined by computing the Euclidean distance of u with all other users in the network. Once all the Euclidean distances of u with other users in the network have been computed, we rank the results and we select the value $dist_k(u)$ of the Euclidean distance at the k-th position in the ranking. Based on this value, we can define the *k-nearest users of a target user u* as the set of users whose Euclidean distances with u is less than $dist_k(u)$.

their local risk factors (see Section 4.4 for more details).

4.3 Topological-based Features for Risk Estimation

In this section, we introduce the features we use for computing our measures. We have driven the selection of them by what have been so far recognized as risky users in OSNs, that is, attackers. The topological patterns of attackers are, in general, different from those of normal users. In the following, we first summarize the most notable attacks and related topological information, we then introduce the considered features.

4.3.1 Risky behaviors in OSNs

Sybil attacks are one of the most prevalent and practical attack in OSNs [54]. To launch a Sybil attack, a malicious user has to create multiple fake identities, known as Sybils, with the purpose to legitimate his/her identity [54]. After that, attackers start sending friendship requests to other users in the community. Once the requests have been accepted, the socialbot can gather users' private data. Sybil attacks can be classified into fourth main types.

The first is Sybils with a *tight-knit community* (dense friendship graph), where adversaries create huge number of Sybils by also establishing connections among them [14, 25]. Due to this high number of connections, Sybils tend to form tight knit clusters in their direct subgraph.

The second category are Sybils with a *sparse community* (sparse friendship graph). Authors in [157] have analyzed the distribution of Sybil accounts in the Renren OSNs.[2] This shows that the vast majority of Sybil accounts do not form social links among them. Moreover, even in case they form, the resulting clusters are loose, rather than tightly connected. They found that attackers use snowball sampling techniques to identify and send friend requests to popular users, since these are more likely to accept requests from strangers [54]. Therefore, their friendship graph become sparse. Moreover, authors in [5] show that the majority of anomalous users in an OSNs have neighbors that are either very well connected (forming a near-cliques), similar to Sybils with tight-knit community, or not connected (stars), similar to Sybils with sparse community.

The third category are Sybils with normal friendship graph. Some works show that Sybils first establish few connections among themselves, and then they try to send friendship requests to other users [54]. In this way Sybils have a friendship graph that is not sparse or dense in the direct subgraph, since they have a small number of mutual friends with their direct contacts. Although Sybils in this category have a normal friendship graph in their direct subgraph, researchers prove that in most cases Sybils fail to create friendship links with legitimate users [14, 25] and majority of their friends are popular users or other malicious users. Therefore, the structure of the direct subgraph of their friends is different from those of legitimate users.

[2] http://www.renren-inc.com/en/

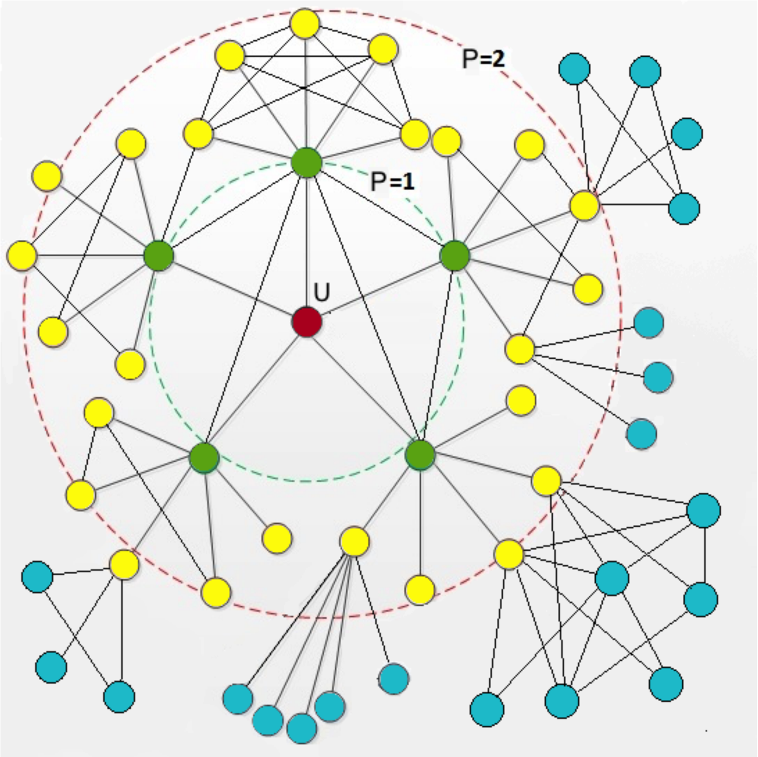

Figure 4.1: Subgraph for each target user

The fourth category of fake users are those that are created by non-malicious users, called creepers [139] wishing some extra accounts, for several purposes, like social reasons such as friendly pranks, stalking, cyberbullying, etc. [139]. A recent paper stated that the market of buying fake followers and fake retweets is already a multimillion-dollar business [142].

4.3.2 Features

Given a user u, we model the graph from which the topological features of u are extracted as the subgraph formed by the set of users, and related relationships, that can be reached by P-steps from u as exemplified by Figure 4.1. In computing the features, we consider P=2, since, as discussed in [8], real social networks show a small diameter. Moreover, the motivation of extracting the topological features of u from its 2-step subgraph is analysing the topological patterns of the direct and two-step subgraph of real attackers in OSNs based on the discussion in Section 4.3.1, that are different from those of normal users. As it will be discussed in the following, the first three features consider the structure of the first subgraph of user u:

Degree of u, (*Degree*), that is the number of direct contacts of u. Researchers show that most attackers send a lot of friendship request to increase their influence in the network

[54, 139]. This feature is helpful to extract this topological pattern of attackers.

Triangles count of u, ($TriangleCount$), where a triangle exists when a node has two adjacent nodes that are also adjacent to each other. This feature can be useful to indicate the topological pattern of both Sybils with dense friendship graph that have a lot of mutual friends with their direct contacts (high triangle count (near-cliques)) and Sybils with sparse friendship graph that do not have mutual friends with their direct contacts (low triangle count (stars)).

The ratio between degree and triangle count of u, that is, $RateDT = Degree/TriangleCount$. The motivation of extracting this feature is that we assume attackers show a different value compared to legitimate user. For example, Sybils tend to have a high degree, although their triangle count is either too high or low in comparison with those of normal users.

We also introduce these features that consider the topological patterns of the two-step subgraph of the target user u. These features can help us to detect Sybils with normal friendship graph that have a small number of mutual friends with their direct contacts and their first sub-graph is similar to the normal users. But, their two-step subgraph is different from normal users, because majority of their friends are either popular users or other malicious users that tend to have a high degree in the network due to accepting friend request from strangers [157, 14, 25]. Therefore, the topological pattern of these popular and malicious users are different from normal users.

The average degree of all direct contacts of u, that is, $AvgDegree$.

The average triangle counts of all direct contacts of u, $AvgTriangleCount$.

The average ratio between degree and triangle count of all direct contacts of u, $AvgRateDT = Avg(Degree/TriangleCount)$.

4.4 Local Risk Score

Our goal is to assign a risk score to the direct contacts of a target user u, based on the deviation of their divergency factors. As introduced in Section 4.2, we exploit the Influence Outlierness (INFLO) [74] for the computation of the divergency factor. INFLO exploits not only the k-nearest users, but also, the reverse k-nearest users (RNU) [29]. Members of RNU of a user u are users that have u as one of their k-nearest users. More formally, we introduce the definition of k-nearest users and reverse k-nearest users.

Definition 2 (*k-nearest users of* u) *Let G be the graph modeling the OSNs, and u be a node in G. Given a value k, the k-nearest-users of u are defined as:*

$$NU_k(u) = \{u' \mid u' \in G, dist(u,u') \leq dist_k(u)\} \tag{4.1}$$

where $dist(u,u')$ denotes the Euclidean distance between u and u' computed on a selection of features among $\{Degree$, $TriangleCount$, $RateDT$, $AvgDegree$, *AvgTriangleCount*,*AvgRateDT*$\}$;[3] $dist_k(u)$ is the Euclidean distance value between u and the user in G placed in the k-th position w.r.t. the Euclidean distance ranking.

Definition 3 (***Reverse k-nearest users of u)*** *Let G be the graph modeling the OSNs, and u be a node in G. Given a value k, the reverse k-nearest-users of u is defined as:*

$$RNU_k(u) = \{u' \mid u' \in G, u \in NU_k(u')\} \tag{4.2}$$

Given a user u the union of $NU_k(u)$ and $RNU_k(u)$ forms its local neighborhood space denoted as $SN_k(u)$ to estimate the density distribution around u. According to [74], INFLO is defined as the ratio of the average density of users in $SN_k(u)$ to the u's local density:

$$INFLO_k(u) = \frac{den_{avg}(SN_k(u))}{den(u)} \tag{4.3}$$

where den_{avg} is the average density of users in $SN_k(u)$ and $den(u)$ is the local density of user u. Based on INFLO, we can now provide the definition of divergency factor.

Definition 4 (***Divergency Factor)*** *Let G be the graph modeling the OSNs, and u be a node in G. Given a value k, the Divergency Factor of u, $DF_k(u)$, is given by $INFLO_k(u)$, where the Euclidean distances are computed on a selection of features among $\{$Degree, TriangleCount, RateDT, AvgDegree, AvgTriangleCount,AvgRateDT$\}$.*

The divergency factor is a measure to define how much the neighborhood of a given user u is different from his/her nearest users. We can then use the divergency factor to assign a local risk factor to each direct contact of a target user u. In more detail, given a target user u, in order to calculate the LRF of a user y in the u's direct contacts, say users set Y, we consider two measures for y. The first is the y's DF that shows how much the density of the neighborhood of y is different from its density. However, since it might be that all direct contacts of user u are considered risky (i.e., their DF values are higher then a given threshold) or all of them are normal (i.e., DF lower than the threshold), we introduce a second measure. This is divergency factor deviation (DFD), that shows how much the divergency factor of y is different from the divergency factor of other u's direct contacts, that is, users in Y. This helps us to detect those users that are more deviate than others in Y that is formally defined as follows.

Definition 5 (***Divergency factor deviation)*** *Let G be the graph modeling the OSNs, and let u be a node in G, and let y be one of its direct contacts. Given a value k, the divergency factor deviation of y for u is the defined as:*

$$DFD_k(u, y) = DF_k(y) - (STDDF(u) + ADF(u)) \tag{4.4}$$

[3] As described in Section 4.5, in order to validate the effectiveness of the proposed features, we carried out experiments by considering not only all the six features, but also a selection of them.

where $DF_k(y)$ is y's divergency factor; $STDDF(u)$ and $ADF(u)$ are the standard deviation and the mean of the divergency factor values of all direct contacts of u, respectively. Based on the above definitions, given a target node u and one of its direct contact, say y, we combine these two measures $DF_k(y)$ and $DFD_k(u, y)$ to obtain the Local Risk Factor, formally defined as follows.

Definition 6 (*Local Risk Factor*) *Let G be the graph modeling the OSNs, and let u be a node in G, and let y be one of its direct contacts. Given a value k, the Local Risk Factor of y for u is the defined as:*

$$LRF_k(u, y) = DF_k(y) + DFD_k(u, y) \tag{4.5}$$

Given a target node u, we first compute the LRF for each of its direct contacts, then we rank them based on their LRFs and we flag as risky those contacts whose LRF is higher than a threshold, denoted as $LRFT(u)$ and defined based on the distribution of LRF values of u's contacts. In particular, the threshold for target user u is computed as: $LRFT(u) = STDLRF(u) + MeanLRF(u)$, where $STDLRF(u)$ and $MeanLRF(u)$ are the standard deviation and the mean of the LRFs of all direct contacts of u, respectively.

4.5 Experiments

Experiments aim at showing how the proposed local risk factor measure can be used to detect risky users in the contact list of a target user u. At this purpose, we have used a real social graph, that is, the Orkut Online Social Network (OSNs) dataset taken from SNAP.[4] In this dataset, there are 3,072,441 nodes and 117,185,083 edges. Unfortunately, the dataset is not provided with a ground truth, in that we do not have any information about which nodes in the Orkut dataset are risky. This is a common problem in validating anomaly detection techniques, where, as discussed in [6], several different validation approaches have been used in the literature, such as anomaly injections or qualitative analysis. There are several works based on anomaly injection to evaluate the result of anomaly detection models [31, 133]. In this chapter, we follow the idea of injecting fake users into the real graph, that is, nodes and random connections, created such as to simulate some kind of attacks. In particular, we simulate four different categories of risky users in OSNs (see discussion in Section 4.3).

For each category, we inject the nodes into the Orkut dataset. Then, we compute a divergency factor for all the users in the obtained new dataset and the local risk factor for all the injected fake users. The LRF of a fake user is computed w.r.t. a target user u. In particular, the target user is selected among the real nodes in the Orkut social graph that have at least a connection with the injected fake node. Finally, we flag as risky a fake user, if its LRF deviates from those of the other direct contacts of u, based on the computed threshold value. In other words, if the LRF of fake node is higher than a local threshold of target user, we flag him/her as risky.

[4] https://snap.stanford.edu/data/

In all the experiments we set k from 5 to 10, since considering a high value for k in large social graph has a high computational cost and, according to [74], does not have a big effect on the result. Moreover, in order to test the effectiveness of the features described in Section 4.3.2, we carried out experiments by considering different combinations of the features. In the following, we first describe the feature combinations, we then introduce the three categories of injected fake users, finally we discuss the experimental results.

4.5.1 Features settings

Once the six features described in Section 4.3.2 have been computed, we consider the following different combinations for computing the Euclidean distances.

All the six features. According to this setting, we consider all the six features described in Section 4.3.2. Figure 4.2 shows the distribution of computed divergency factors for all users in the considered dataset. In this feature setting, the DF is in the range between [0, 250] and most of the users have a DF in the range [0, 2.7], whereas few of them are with DF higher than 2.7.

Ratio of degree to triangles count. In this setting, we consider only *RateDT* and *AvgRateDT* in the divergency factor computation. In the *RateDT*, we are interested to catch those users for witch there is no balance between their degree and triangle count as this could indicate a risky conduct. In addition, by bringing *AvgRateDT* into account, we consider also for all his/her direct contacts as well. Therefore, we consider both the subgraph around each user and all his/her direct contacts for our risk estimation.

In this way, we are able to catch users whose two steps subgraph are very well connected (near-cliques) or not connected (stars). As we can see in Figure 4.3, most of the users have a DF in the range [0, 2.7], whereas just few of them diverge from their nearest users with DF higher than 2.7. The range of DF is [0, 84].

4.5.2 Injected risky users

In this section, we introduce the four categories of fake users we inject in the Orkut dataset.

Sybils with sparse friendship graph. The first kind of attack that we try to simulate is Sybils with sparse friendship graph in their direct subgraph. As we discuss in Section 4.3.1, these kind of attackers have friendship links with a lot of strangers by using random sampling techniques to send friend requests to strangers. Therefore, we inject 100 users, each one having a number of edges selected randomly in the range of the mean and the sum of the mean and the standard deviation of degree of all users in the real graph (i.e., values in [100, 250]) to be similar to regular users. Then, we totally create around 10000 to 25000 friendship links among these 100 Sybils with randomly selected users from the whole graph by considering random sampling.

Sybils with dense friendship graph. The second type of attackers are Sybils with dense friendship graph or tight-nit communities. To model these attackers, after creating 100 fake nodes and inject them into the graph, we generate the edges among themselves and then, with a set of randomly selected users and also the 80% of their direct contacts

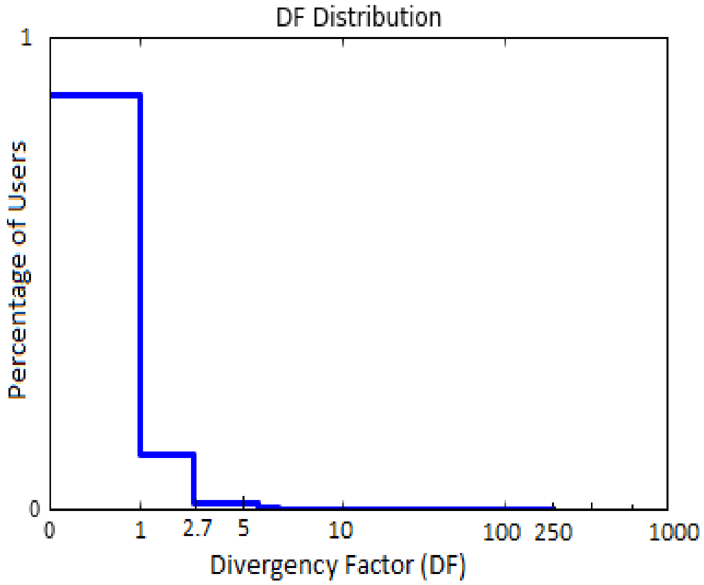

Figure 4.2: DF distribution by considering all the six features

to have more mutual friends with each friend. Moreover, we generate these edges so that each fake node has a degree in the average range of all other legitimate users, to be more similar to other regular users.

Sybils with normal friendship graph. The third type of attackers are Sybils with normal friendship graph. In this kind of attacks, attackers after creating a huge number of Sybil accounts establish few connections among themselves, and then they try to send friendship requests to popular users. To model these attackers, after creating 100 fake nodes and inject them into the graph, we generate the edges with a set of randomly selected users with high degree in the range of [1000, 33000]. Then, totally we create around 10000 to 25000 friendship links among these 100 Sybils with popular users.

Real users with additional fake accounts (creepers). These risky users are real users wishing some extra accounts. Usually these users have not a high degree in the graph, but they just create a fake account and then, randomly pick up some strangers and make friendship links with these random users. The difference of these creepers with Sybils with sparse friendship graph is that they have few friendship links since their goal is not as attackers to influence the graph by having more links. To model this type of risky users, we inject 100 users each having a number of edges in the range of the mean minus standard deviation and mean of degree of all users in real graph (i.e., values in [50, 150]). We create these friendship link with randomly selected users from whole the graph. Then, totally we create around 5000 to 150000 friendship links among all these 100 fake users and other users in the network.

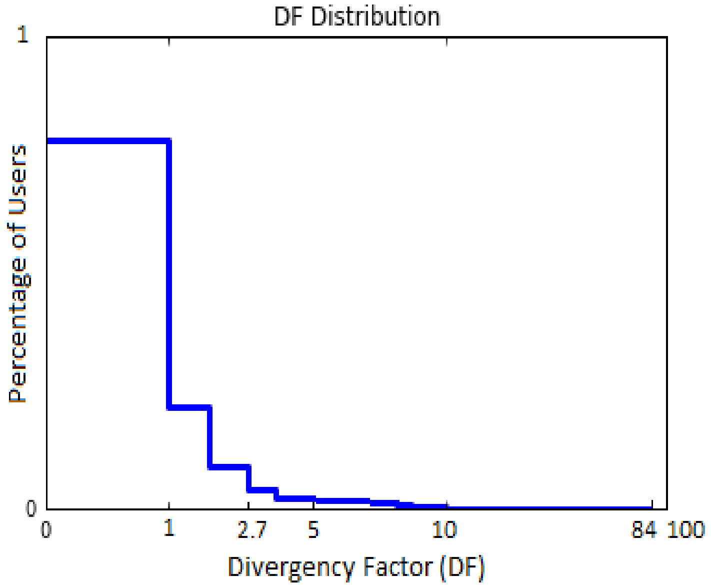

Figure 4.3: DF distribution by considering Ratio and AvgRateDT

4.5.3 Experimental results

We run our experiment with two different feature settings previously discussed on the four different graphs with injected risky users. After calculating the LRF for each user, we compute the difference of the LRF value with the local LRF threshold of target user $LRFT(u)$ that is in contact with the risky user. In this way, if the LRF of each user is higher than $LRFT(u)$ among the other contacts of target user u, the user is detected as risky. We consider the value zero for those risky users such that their LRF is lower than the $LRFT(u)$, since they are not deviate from other contacts of target user u, that is:

$$\begin{cases} LRF_k(u,y) - LRFT(u) & \text{if } LRF_k(u,y) > LRFT(u) \\ 0 & \text{if } LRF_k(u,y) \leq LRFT(u) \end{cases}$$

We flag as risky those users with the difference of their LRF and $LRFT(u)$ higher than zero. The result of the first and second feature settings for the four categories of risky users is shown in Table 4.1. Here, we can see the percentage of risky users that are detected by the majority (more than 50%) of target users that are in contact with them. Furthermore, Table 4.2 represents the percentage of fake users that are detected as risky by at least one of the target users that are in contact with them.

In more details, Figure 4.4 shows 100 different categories of fake users that are detected as risky with the percentage of target users that are in contact with each one. In particular, the x-axis shows the percentage of target users that are in contact with each fake user and able to detect him/her as risky and the y-axis show the percentage of fake users that are

Feature Setting	Sparse Sybils	Dense Sybils	Sybils with normal direct subgraph	Creepers
All Six Features	90%	41%	79%	77%
RateDT and AvgRateDT	95%	76%	90%	95%

Table 4.1: Detection rate of risky users detected by majority of the target users

Feature Setting	Sparse Sybils	Dense Sybils	Sybils with normal direct subgraph	Creepers
All Six Features	100%	52%	95%	89%
RateDT and AvgRateDT	100%	99%	99%	98%

Table 4.2: Detection rate of risky users detected by of at least one of the target users

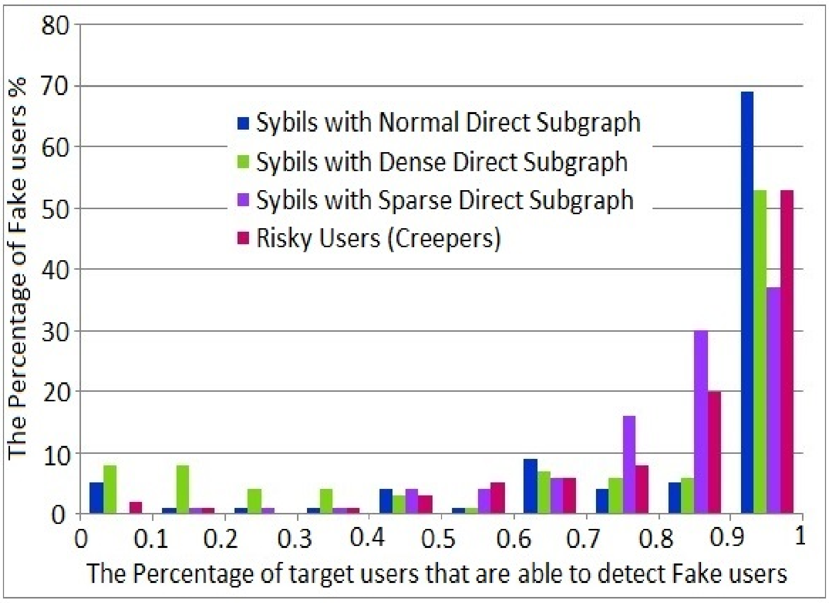

Figure 4.4: Risky users that are detected with target users in feature setting (RateDT and AvgRateDT)

detected as risky. As we can see in the Figure 4.4, most of the fake users are detected with more than 50 % (>= 0.5 in x-axis) of target users that are in contact with them.

Figure 4.5 shows all target users that are in contact with 100 Sybils with sparse friendship graph, to see how many of the target users are able to detect these 100 Sybils as risky. As we can see in Figure 4.5, among 15490 target users that are in contact with these Sybils, around 13470 are able to detect these Sybils in their contact list as risky that is around 86.95 percent of all target users that are in contact with them.

To calculate the performance of our risk models with different feature settings, we compute the F-measure. For calculating the F-measure, we need to compute precision and recall that is based on: false positive (FP), false negative (FN), true positive(TP) and true negative (TN). In order to calculate precision and recall, we need to calculate also

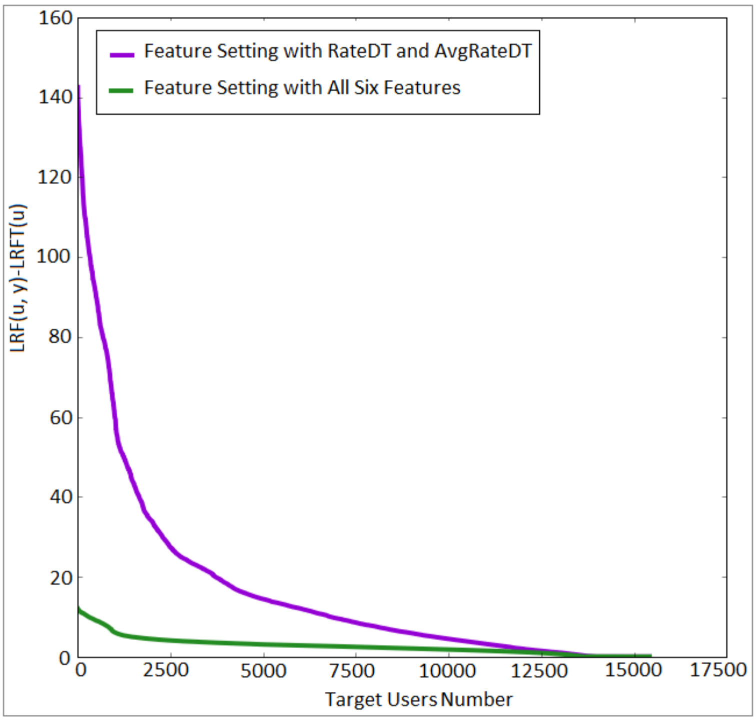

Figure 4.5: Target users that are able to detect Sybils with sparse direct subgraph

the false positive that is the number of legitimate users that are detected as risky in our risk models. Therefore, the evaluation based on precision and recall is challenging since it would be severe to call the risky users detected other than the injected ones as false positives, given that the original real graph may also contain same type of anomalies and risky users [6].

Based on anomaly detection concept, the majority of users obey a pattern and only few users that deviate, considered as outliers [5]. Therefore, to consider a set of legitimate users, we selected 1000 legitimate users randomly not from the whole graph but among those users that their graph structure is similar to majority of users inside the real graph. In other words, we didn't consider outliers for this selection.

Then, we find a measure to find legitimate users since considering all users with high degree or low degree and also users with high triangle count or low triangle count as legitimate or anomalous is not reasonable. This is motivated because, there is a large number of popular users with high degree as shown in Figure 4.6 that is around 30,105 users with number of degree higher than 700 in the range of [700, 33313]. Figure 4.6 shows the number of users in the x-axis and the number of degree in y-axis. The maximum degree of users in the graph is 33313. Furthermore, there is a high number of users with high triangle count or isolated users with low triangle count as shown in Figure 4.7. The maximum value of triangle count is 1,666,622 that we can see there are around 100,000

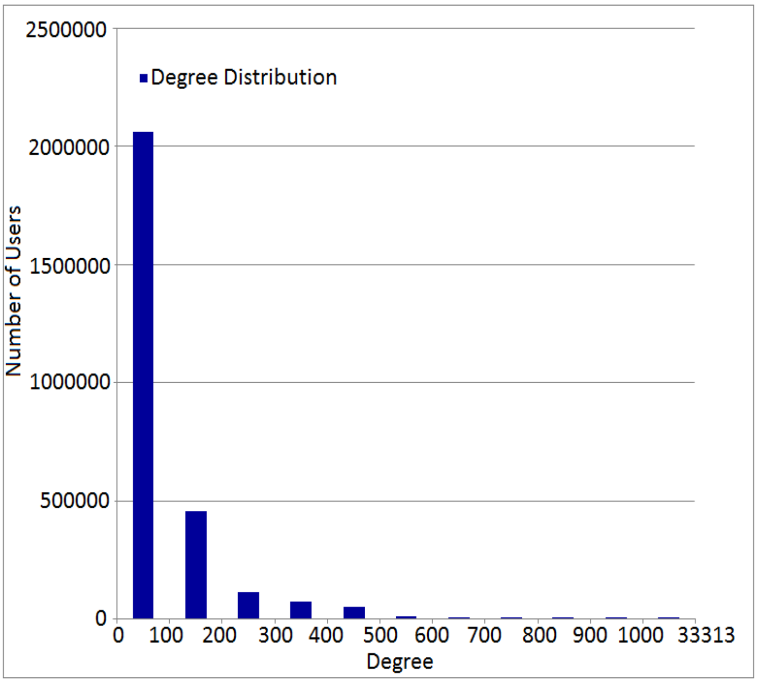

Figure 4.6: The distribution of user's degree in OSNs

users with triangle count in the range of [1000, 1666622].

But, Figure 4.8 shows the relation of increasing the degree with triangle count that is RateDT for all users inside the real social graph. As we can see the majority of users have a RateDT near the red line and just few users surrounded with red circles have these values outside the line that is considered as outlier in [5]. Therefore, we select our 1000 normal users randomly among those users with RateDT between 0.1 and 10, since 98% of users have this range of values.

Then, after computing their LRF, we consider all target users that are in contact with normal users to see the percentage of these normal users that by mistake are detected as risky. For example, between all target users (33156) that are in contact with the 1000 normal users when we consider (RateDT and AvgRateDT) as features, around 1196 of them detect these normal users as risky that is 3.60% percent of all target users.

Table 4.3 represents the F-measure for the two feature settings with all four categories of fake users when the majority of the target users, in contact with them, have detected them. Based on the result, the performance of risk model with two feature settings (RateDT and AvgRateDT) is the best, since these are the most influential features that reveal these kind of risky structural patterns in the graph. As we can see, the performance of detecting sybils with sparse direct subgraph is the best around 90% and sybils with normal direct subgraph and creepers are in the second rank still more than 90%. The performance of detecting sybils with dense direct subgraph is lower around 82% since there are some other

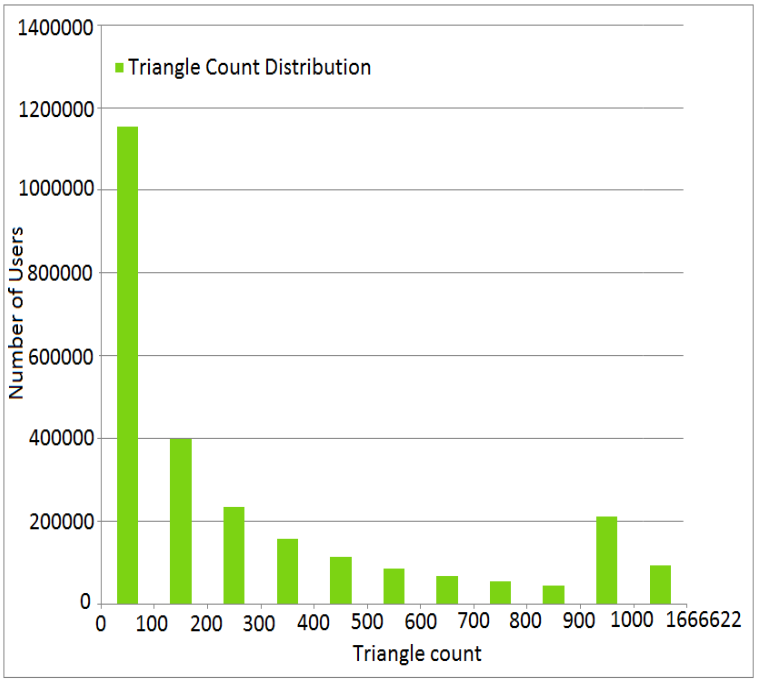

Figure 4.7: The distribution of user's triangle count in OSNs

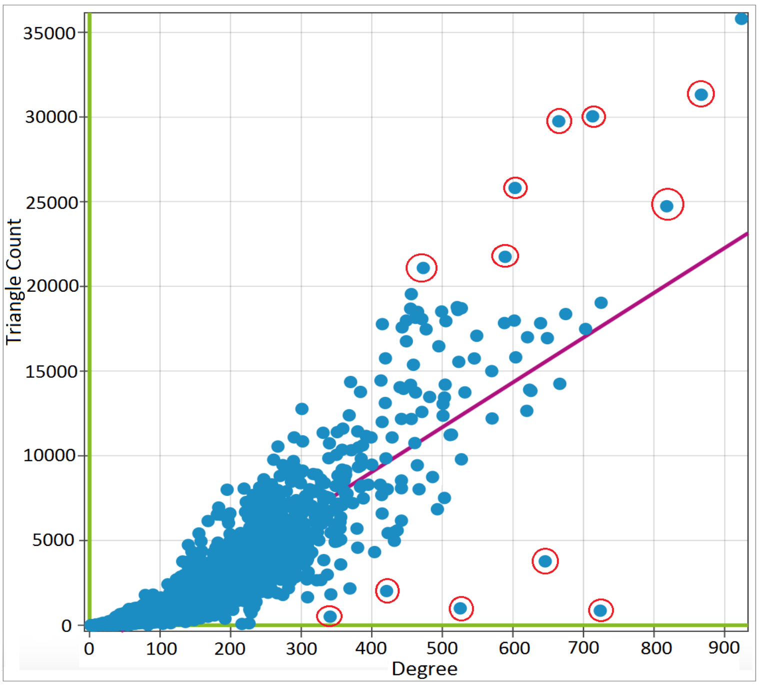

Figure 4.8: The plot of user' RateDT in OSNs

Feature Setting	Sparse Sybils	Dense Sybils	Sybils with normal direct subgraph	Creepers
All Six Features	0.90	0.54	0.839	0.826
RateDT and AvgRateDT	0.939	0.821	0.925	0.936

Table 4.3: F-measure in the two feature settings (majority of target users detect risky users)

Feature Setting	Sparse Sybils	Dense Sybils	Sybils with normal direct subgraph	Creepers
All Six Features	0.955	0.649	0.93	0.897
RateDT and AvgRateDT	0.961	0.956	0.95	0.951

Table 4.4: F-measure in the two feature settings (At least one target user detect risky users)

users inside the real graph with severe case than this category of sybils with a very high degree more than 1000 and very high triangle count more than 900,000 that make their direct subgraph denser than Sybils.

Also, Table 4.3 shows the value of F-measure when at least one target user detects fake users as risky. Based on the result, again the performance of risk model with two feature settings (RateDT and AvgRateDT) is the best. Also, the performance of all four categories of fake users are more than 95 % that is a very good result. We can see that our risk model is able to help target users to detect risky users with a high accuracy and low false alarm (FP) rate.

Chapter 5

Anomalous Change Detection in Time-evolving OSNs

5.1 Introduction

Detection of anomalous changes in time-evolving Online Social Networks (OSNs) has become interesting in recent years. This problem has been widely studied in the context of mining and statistics [91, 75, 82, 9]. However, there are few approaches able to detect change in dynamic social graphs.

In general, these approaches extract a summary of each graph snapshot to be compared over time with the help of similarity functions [121]. Then, when the distance returned by the considered similarity function is higher than a threshold, the corresponding graph snapshot is flagged as anomalous. The goal of these approaches is detecting the time of the anomalous changes in the structure of the whole graph. As an example, [136] computes distance functions among a sequence of graphs, whereas [144] proposes an approach for detecting changes in community structure to identify times of these changes. [103] and [102] consider some network measures, such as closeness centrality, betweenness centrality and the density of the graph, to detect any change in the graph over time.

Another approach is [23], where, in addition to compute the distances between consecutive graph snapshots, the distances between all pair of nodes in the graph are computed as well. They detect the changes that occur in the time evolving graph based on the distances between pair of users. [77] proposes a graph feature-based similarity approach to compare the pairwise node similarity. But, computing all the pairwise similarity scores has high computational cost. A faster algorithm has been proposed to avoid computation of similarity among all pairwise nodes [6]. Recently, [28] proposes a statistical approach to detect change points.

However, these approaches are not able to detect which nodes are responsible for the detected changes. One of the interesting research work in this direction is [4], which considers individual local features (e.g., in-degree, out-degree, in-weight, etc.) for each user and calculates the correlation between these features value over time for all pair of users, to detect change times in the whole graph structure. However, they identify change points where the majority of the users in a whole graph deviate from their normal behaviors. Therefore, if the majority of the users do not deviate from their normal behavior, this approach fails in detecting the changes.

A most recent work, that is, [137], in addition of finding the changes in the whole graph, finds also which nodes or edges are responsible for these changes. However, their key goal is the detection of changes in the whole graph structure.

Compared with the above mentioned approaches for change detection, in this chapter we want to face a different problem. Indeed, our goal is to have a change detection approach able to identify users with anomalous changes in the structure of their subgraphs. For instance, the preparation of socialbots, or other type of attackers, may be associated with changes in the structure or the patterns of friendship links between users in an OSN. Therefore, the ability to effectively and efficiently detect these changes has the potential to enable the service providers of OSNs to anticipate and respond to these attacks. In particular, given a user u we are interested in his/her changes that can be considered anomalous compared to: (a) changes of other similar users, and (b) changes performed by him/her in the past.

More precisely, we analyze and monitor the change patterns of users over time and compare them with their own previous change patterns and the change patterns of other similar users in the network. Towards this goal, we consider local structural information for each user by analyzing his/her subgraph over time. Then, we measure how much the subgraph of a user changes over time, based on some distance metrics, and compare it with his/her previous changes and with those of other similar users in the network.

This approach has many benefits. First, it does not consider global changes in the whole social graph structure, but it only considers local changes in user's subgraphs. By considering the changes in the whole graph structure, we are not able to identify some kind of attackers characterized by changes in the structure of their subgraph, nor which users change the most. Second, our approach identifies those users with anomalous changes that deviate from their own previous change patterns. This is good to detect the changes in the behavior of attackers. Third, it can identify those users with anomalous changes in their subgraph structure that deviate from other users with the same change patterns in the whole graph. This is useful to analyze the structural patterns of attackers in comparison with those of normal users. Finally, it can measure the degree of changes for each user, by thus providing a level of anomaly, which can be used to trigger the proper response. Moreover, our approach is well suited for OSNs where the number of users and edges are very large.

Furthermore, identifying the exact time that these changes occur is good to identify the preparation of socialbots or other type of attacks that are directly associated with changes in the patterns of friendship links. We analyze the performance of our approach on a real Google+ dataset over different graph snapshots.

The remainder of this chapter is organized as follows. Section 5.2 introduces the overall idea underlying our approach, whereas Section 5.3 describes the considered graph based local structural features. Section 5.4 presents the anomalous change scores. Finally, experiments are presented in Section 5.5.

5.2 Overall Approach

The goal of the proposed approach is to monitor and analyze the change patterns in the structure of user's subgraph and to identify those users with anomalous changes. Towards this goal, given a user u in an OSN, we extract local structural features in u's subgraph (e.g., degree, triangles, ratio between degree and triangles) for each graph snapshot G_t, at a given time instant t, as shown in Figure 5.1(a). In order to monitor user changes over time, we compute the Euclidean distance [121] between the structural features of u's subgraph (such features will be explained in Section 5.3), in the current graph snapshot (e.g,. G_{t+1}) and its features in the set of w previous graph snapshots, where w is a fixed size sliding window. The comparison of the structure of u's subgraphs over different snapshots is used to produce the time series of subgraph's distances (e.g., $\{d_{(G_3,G_1)}, d_{(G_3,G_2)}, \ldots, d_{(G_{t+1},G_{t-1})}, d_{(G_{t+1},G_t)}\})$, over the fixed size sliding window w.

Suppose, $w = 2$, given a sequence of four graph snapshots $\{G_1, G_2, G_3, G_4\}$ and a user u in these four graph snapshots, there is a sequence of distances between the structural features of u's subgraph at each graph snapshot, e.g., G_4, and the structural features of u's subgraph in the two previous graph snapshots, i.e., G_2 and G_3, that is, $\{d_{(G_4,G_2)}, d_{(G_4,G_3)}\}$. Therefore, there will be a set of distances for user u over all these four graph snapshots, $\{d_{(G_2,G_1)}, d_{(G_3,G_1)}, d_{(G_3,G_2)}, d_{(G_4,G_2)}, d_{(G_4,G_3)}\}$.

Given a sequence of distances indicating the changes in the structure of user's subgraph over time, as shown in Figure 5.1(b), we obtain thus the set of user distance vectors, denoted as $DV(u) = \{dv_1(u), \ldots, dv_t(u), dv_{t+1}(u), \ldots\}$, where each element corresponds to a distance vector at time instant t and contains the set of distances of u's subgraph at time t (i.e., in G_t) with its w previous snapshots (i.e., $G_{t-1}, G_{t-2}, \ldots, G_{t-w}$), where $dv_t(u) = \{d_{(G_t,G_{t-1})}, d_{(G_t,G_{t-2})}, \ldots, d_{(G_t,G_{t-w})}\})$. For example, the distance vector of user u at time 4 is $dv_4(u) = \{d_{(G_4,G_2)}, d_{(G_4,G_3)}\}$, whereas the distance vector at time 3 is $dv_3(u) = \{d_{(G_3,G_1)}, d_{(G_3,G_2)}\}$, assuming $w = 2$.

We then identify those users that have anomalous changes in the structure of their subgraphs over time by comparing their current distance vector with; a) those of other similar users in the network; and b) his/her previous distance vectors.

At this purpose, we need a measure for comparing distance vectors. For this, we exploit [74], which proposes a density based outlier detection technique. They compare the local density of a node to the local densities of its nearest users in the network and flag a node as an outlier if it has a lower density than its nearest users. In particular, based on [74], in our approach we propose two factors to check whether a node is an outlier; a) k-nearest Anomalous Change (KAC) factor, and b) Historical Anomalous Change (HAC) factor. We will explain the definition of these two factors in Section 5.4.

In the following, we first discuss how we extract local structural features for each user. We then describe how the proposed factors can be used to compute an anomaly score that reflects how the sequences of subgraph changes of a user deviate from those of other users

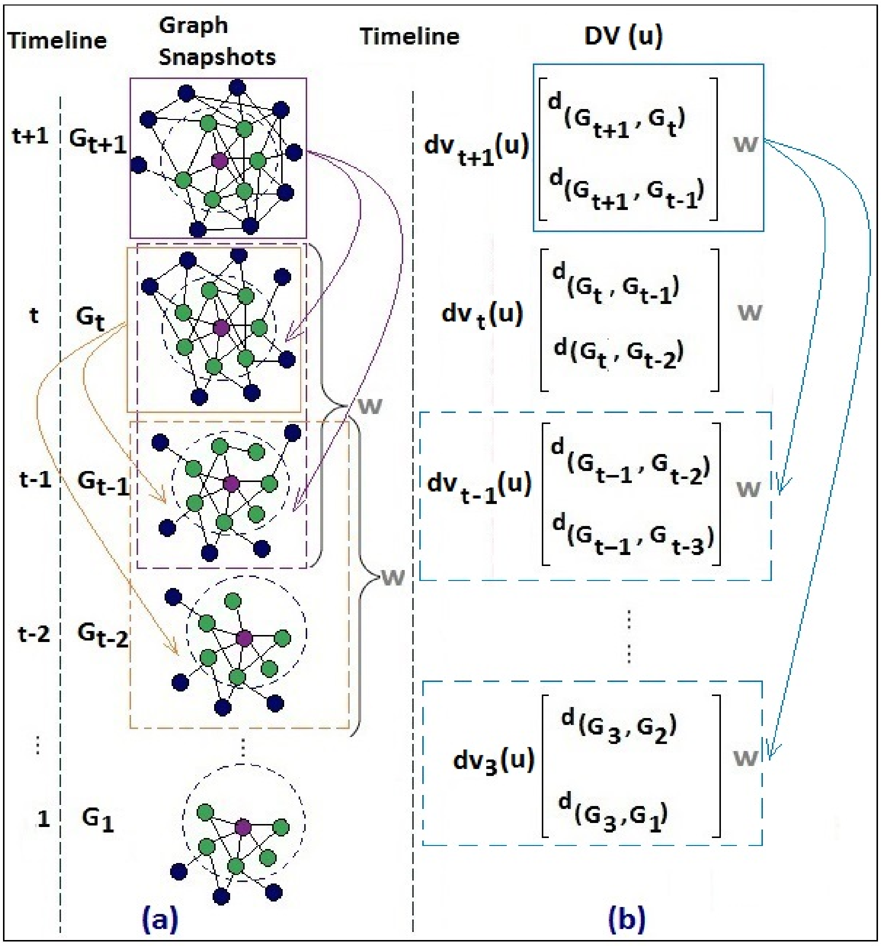

Figure 5.1: Graph snapshots and the sequence of distance vectors for a user u

and also from his/her historical change patterns.

5.3 Local Structural Features

The local structural features that we exploit are related to the type of changes that can happen in an OSN. There are several possible changes in an OSN graph such as:

- change in the number of friends (degree) of a user;
- change of the number of mutual friends of a user with his/her direct friends (triangle count);
- change in the ratio between degree and triangle count of a user. If this ratio increases it implies that the subgraph of the user becomes more sparse. In contrast, if the ratio

between degree and triangle count of a user decreases it implies that the subgraph of the user becomes more dense.

To keep into account the above types of changes, given a user u, we extract the following local structural features:

- **Degree** of u (*Degree*), that is, the number of direct friends of u. This feature is helpful to identify those users whose degree increases or decreases over time in an anomalous way.
- **Triangles count** of u ($TriangleCount$), where a triangle exists when a node has two adjacent nodes that are also adjacent to each other. An high value of this feature for a target user u shows that the high number of mutual friends among friends of u caused to have a dense (near-cliques) subgraph. On the other hand, a low value for this feature for a target user u shows a small number of mutual friends among the friend of u, that causes to have a sparse (near-stars) subgraph. For instance, sybils with a dense subgraph are those attackers that in order to legitimize their account try to have a high number of mutual friends with their friends and form a tight knit cluster in their direct subgraphs.
- **The ratio between degree and triangle count** of u, that is, $RateDT = Degree/TriangleCount$. This feature shows the relation between degree and triangle count. More precisely, the value of degree has a direct impact on the value of the ratio, whereas, the value of triangle count has inverse impact on the value of the ratio. When, user's degree increases over time while the triangle count decreases, the value of the ratio decreases. It means that, although the degree of user increases, but, the number of mutual friends with user's friends decreases the subgraph of the user becomes sparser than before. On the other hand, if the triangle count increases, it makes the subgraph of the user denser. Therefore, this feature can be useful to analyze whether the structural patterns of users over time become sparser or denser.

5.4 Anomalous Changes

Each snapshot G_t is an undirected social graph with a various number of nodes (i.e., users) $U_{G_t} = \{u_1, \ldots, u_n\}$ and an edge set $E_{G_t} = \{e_{(1,2)}, \ldots, e_{(1,n)}\}$, with $e_{(i,j)}$ denoting the edge between users u_i and u_j. For each graph snapshot G_t, and for each transition between graph snapshots G_{t+1} and G_t, our goal is to detect anomalous changes in the structure of user's subgraphs. A particular user u is deemed to be anomalous if u's distance vector at time instance $t+1$, $dv_{t+1}(u) = \{d_{(G_{t+1},G_t)}, d_{(G_{t+1},G_{t-1})}, \ldots, d_{(G_{t+1},G_{t-w})}\}$, deviates from the norm.

More precisely, our goal is to define how much the recent changes of user u in the time window w deviate from: a) those of other nearest users in the network; and b) his/her previous changes in the window w.

Toward this goal, we define two anomalous change factors, k-nearest Anomalous Change factor (KAC) and Historical Anomalous Change factor (HAC). The KAC factor for a particular user u, denoted as $KAC_k(u)$, indicates anomalous changes of user u over time window w in comparison with those of his/her k-nearest users in the network. The HAC factor, denoted as $HAC_h(u)$, indicates anomalous changes of user u in comparison with h-nearest distance vectors of u, where h is the number of nearest previous distance vectors of u. More precisely, the h-nearest distance vectors are the previous distance vectors that are more similar to the distance vector of u at time instance $t+1$, $dv_{t+1}(u)$ (we will explain this in Section 5.4.2). In the following, we explain these two anomalous factors in more details.

5.4.1 K-nearest Anomalous Change Factor

In order to define $KAC_k(u)$ for a particular user u, we borrow the idea of density based anomaly detection, called the Influence Outlierness (INFLO) [74] that is based on Local Outlier Factor (LOF) [22]. These approaches used a density measure to catch anomalous users. More precisely, authors in [74, 22] compare the local density of a node to the local densities of its k-nearest users in the network. The node is an outlier if it has a lower density than its k-nearest users. More formally, the definition of k-nearest users is as follows:

Definition 7 (***k-nearest users of*** u) *Let G_{t+1} be the graph snapshot at time $t+1$, and u be a node in G_{t+1}. Given a value k, the k-nearest users of u are defined as:*

$$NU_k(u) = \{u' \mid u' \in G_{t+1}, dist(u, u') \leq dist_k(u)\} \tag{5.1}$$

where $dist(u, u')$ denotes the Euclidean distance between u and u' computed on $\{d_{(G_{t+1},G_t)}, d_{(G_{t+1},G_{t-1})}, \ldots, d_{(G_{t+1},G_{t-w})}\}$. $dist_k(u)$ is the Euclidean distance between u and the user in G_{t+1} placed in the k-th position w.r.t. the Euclidean distance ranking. Therefore, users in $NU_k(u)$ are those users that are more similar to u, based on the distance vectors over time window w.

Definition 8 (***k-nearest local density of*** u) *The k-nearest local density of u, denoted as $den_k(u)$, is the inverse of $dist_k(u)$:*

$$den_k(u) = \frac{1}{dist_k(u)} \tag{5.2}$$

The KAC factor returns a measure of how much the changes in the structure of user u's subgraph are different from those of his/her nearest users. It is defined as the ratio of the average density of users in $NU_k(u)$ and u's k-nearest local density:

$$KAC_k(u) = \frac{den_{avg}(NU_k(u))}{den_k(u)} \tag{5.3}$$

where den_{avg} is the average density of users in $NU_k(u)$, whereas $den_k(u)$ is the local density of user u.

5.4.2 Historical Anomalous Change Factor

Historical Anomalous Change factor, $HAC_h(u)$, identifies those users whose structural changes in their subgraphs deviate from their historical changes in the past.

In order to define $HAC_h(u)$ for a user u, similar to the KAC factor, we compute the Euclidean distance but in a different way. In more details, we compute the Euclidean distance of u with his/her previous distance vectors. After ranking the results, we select the value $dist_h(u)$ at the h-th position in the ranking and define the set of distance vectors whose Euclidean distances with u is less than $dist_h(u)$. We consider this list of distance vectors as h-nearest distance vectors of u. After that, we compare the local density of a user wrt the current distance vector to the local densities of its h-nearest distance vectors in previous snapshots. The user is an outlier if it has a lower density than its h-nearest distance vectors.

In particular, the definition of h-nearest distance vectors is as follows:

Definition 9 (***h-nearest distance vectors of*** u) *Let u be a node in the graph snapshot G_{t+1}, with the current distance vector for $w = 2$, $dv_{t+1}(u) = \{d_{(G_{t+1},G_t)}, d_{(G_{t+1},G_{t-1})}\}$. The h-nearest distance vectors of u are defined as:*

$$NV_h(u) = \{v' \mid v' \in dv_t(u), dist(u, v') \leq dist_h(u)\} \tag{5.4}$$

where $dist(u, v')$ is the Euclidean distance between u and v' computed on $\{d_{(G_{t+1},G_t)}, d_{(G_{t+1},G_{t-1})}\}$. $dist_h(u)$ is the Euclidean distance between the current distance vector of u and his/her previous distance vectors placed in the h-th position w.r.t. the Euclidean distance ranking. Therefore, the vectors in $NV_h(u)$ are those historical distance vectors that are more similar to the current distance vector of u.

Definition 10 (***h-historic local density of*** u) *The h-historic local density of a user u, denoted as $den_h(u)$, is the inverse of $dist_h(u)$:*

$$den_h(u) = \frac{1}{dist_h(u)} \tag{5.5}$$

The HAC factor returns a measure of how much the change pattern of user u's subgraph at current time is different from his/her previous distance vectors. It is defined as the ratio of the average density of vectors in $NV_h(u)$ and u's h-historic local density:

$$HAC_h(u) = \frac{den_{avg}(NV_h(u))}{den_h(u)} \tag{5.6}$$

where den_{avg} is the average density of vectors in $NV_h(u)$ and $den_h(u)$ is the local density of user u.

5.4.3 Local Anomalous Change Factor

In this section, we discuss how the previously defined anomalous factors are combined to reach a final anomaly score that reflects how the current changes in the structure of user's subgraph deviate from his/her nearest friends and historic change patterns.

Given a target node u, we combine the two anomalous measures $HAC_h(u)$ and $KAC_k(u)$ to obtain the Local Anomalous Change factor (LAC), formally defined as follows:

Definition 11 (*Local Anomalous Change Factor*) *Let G_{t+1} be the graph snapshot at time $t+1$, and let u be a node in G_{t+1}. The LAC factor of u is defined as:*

$$LAC(u) = HAC_h(u) + KAC_k(u) \tag{5.7}$$

Given a target node u, we first compute the LAC factor, then we rank users based on their LACs. We flag a user as anomalous if his/her LAC is higher than a threshold, denoted as $LACT$, which is defined based on the distribution of LAC values of all users in G_{t+1}.

5.5 Experimental results

In this section, we show how our proposed measures are able to detect those users with anomalous changes in the structure of their subgraphs. At this purpose, we have used four snapshots of Google+ [60, 59]. The first snapshot, referred to as G_1, includes 4,693,129 nodes and 47,130,325 edges. In the second one (i.e., G_2) there are 17,091,929 nodes and 271,915,755 edges, whereas in the third and fourth (i.e., G_3 and G_4), there are 26,244,659 and 28,942,911 nodes and 410,445,770 and 462,994,069 edges, respectively.

Unfortunately, the dataset does not provide any information on which users in the dataset are anomalous. However, there are several validation approaches for anomaly detection techniques [6, 31, 133, 46] when the ground truth is not available, since this is a common problem in the anomaly detection area. Among the proposed approaches, we chose the one proposing the injection of anomalous nodes into the real graph in order to evaluate the performance of anomaly detection models [6]. In particular, we follow the approach of injecting fake users which are created by simulating attackers structural patterns. Then, we manually inject some connections between these fake users to simulate some kinds of attacks. After that, we monitor and analyze the changes in the structure of user's subgraph over all graph snapshots to detect those users that have anomalous change over time by calculating the three proposed anomalous change factors for all users including fake users.

More precisely, we simulate four different categories of anomalous users in OSNs. In particular, the most prevalent and practical attacks in OSNs are sybils or socialbots. In these attacks, attackers create some fake accounts, called sybils, in order to create some friendship links with other users in the network to gather private information or propagating malwares. There are different structural patterns for sybils, the most notable are: a) sybils with a *tight-knit community* (dense friendship graph) and, b) sybils with a *sparse community* (sparse friendship graph).

Sybils with a tight-knit community are those attackers that, in order to have a high number of friends, after creating some fake accounts, they first establish some friendship links among themselves and then send friend requests to other users [14, 25]. In this way, they legitimate their identity by having more mutual friends with their friends to form tight knit clusters in their direct subgraphs.

The second category of sybils is that of sybils with a sparse graph. In this category, attackers create some fake accounts and then, use snowball sampling techniques [157] to send friend requests to high degree users (i.e., popular users), since they are more interested to accept friend requests from strangers [54]. Therefore, their subgraphs become sparse.

Based on these types of sybils, we injected different types of fake users into the last Google+ snapshot, that is, G_4. In particular, we randomly select some users as sybils and inject some random edges among them with other users in the Google+ dataset. Then, we compute our three anomalous change factors for all the users in the obtained new graph snapshot, including the injected fake users. Finally, we flag as anomalous a user if its LAC deviates from those of the other nearest users and his/her historical change patterns. We set the values of w and, h to 2, and k from 5 to 10.[1]

In the following, we first describe the three categories of injected fake users and then we discuss the experimental results.

5.5.1 Injected anomalous users

In this section, we describe the three type of injected anomalous users:

- **Sybils with tight-knit community**. To model these attackers, we first randomly select 100 nodes in the last graph snapshot as sybils, then we add edges among themselves, plus edges with other random users. These random users are selected among those with large degree in the range of [1000, 35000] (greater than 10 times the average). Then, we add edges with 80% of friends of these high degree users, since attackers establish friendship with those popular users and also their friends to infiltrate OSNs. Here, we simulate the worst case of these sybils, by assuming that all these users accept sybil's friend requests. The number of injected edges for each sybil is between 10 to the average degree of all users in the network. This is done in order to have some sybils with small changes in degree and some with big changes in degree.

- **Sybils with sparse community**. This kind of attackers use snowball sampling techniques to identify and send friend requests to high degree users to have friendship links with a lot of strangers. The difference wrt the previous ones is that they do not first create friendship links among themselves. Therefore, we randomly select 100 users, and, for each one of them, we establish a number of new edges in the range

[1] since considering a high value for k in large social graphs has a high computational cost and, according to [74], it does not have a big effect on the result.

[10, 350][2] with high degree users, in order to have some sybils with big changes in degree and some with small changes in degree.

- **Random fake users**. These anomalous users are completely random in creating new edges. Therefore, we randomly select 100 nodes from the last graph snapshot as fake users and create edges among these fake nodes. For each of these fake users we establish a number of new edges in the range [10, 350] with randomly selected users from the whole graph, not only with high degree/popular users as in the previous cases.

5.5.2 Obtained results

We run our experiments to compute the proposed anomalous factors: 1) $KAC(u)$, b) $HAC(u)$, and c) $LAC(u)$ on the three different graphs with the three different types of injected anomalous users. More precisely, we have four snapshots and we injected to the last (4th) snapshot each type of sybils and fake users (sparse sybils, dense sybils, and random fake users). After that, we compute anomalous factors for all users in each snapshot separately.

First we compute the first anomalous change factor, $KAC_k(u)$, and we flag those users where KAC is higher than a threshold value, $KACT$, which means their change pattern deviates from other nearest users in the graph. We set the threshold value by computing the distribution of KAC factor values for all users in G_4.

Then, we compute the second anomalous change factor, $HAC_h(u)$, and we flag those users where HAC is higher than a threshold value, $HACT$, which means their change patterns over time deviate from their own historical changes. The setting of threshold value is based on the distribution of HAC for all users in G_4.

As shown in Figure 5.2, around 90% of users have the KAC value less than 1, less than 10% of users have the value in the range [1, 2], few users have the value in the range [2, 3], and a very small number of users have a value greater than 3. Based on this, we set the threshold $KACT$ to 2. Therefore, users with KAC higher than 2 are considered anomalous.

The distribution of HAC is shown in Figure 5.3. Here, around 80% of users have the value in the range [0,1], 15% have the value in the range [1, 2], and 5% of users have the value in the range [2,3]. Since very few users have the value higher than 3, we set the threshold $HACT$ as 3.

After selecting the thresholds, we compute the detection rate of those injected risky users based on KAC, HAC, and LAC. We can see in Figure 5.4 that the detection rate of sybils with sparse friendship graphs and random fake users are higher than the one of the sybils with dense friendship graphs. Because, there are some users with dense friendship graphs in the real snapshot that have more anomalous changes than these fake users. However, because of the lack of ground truth, we can not proof either they are fake or real users, and this can have a consequence on the obtained performance.

[2] 350 is the average degree of all users in the 4th graph snapshot.

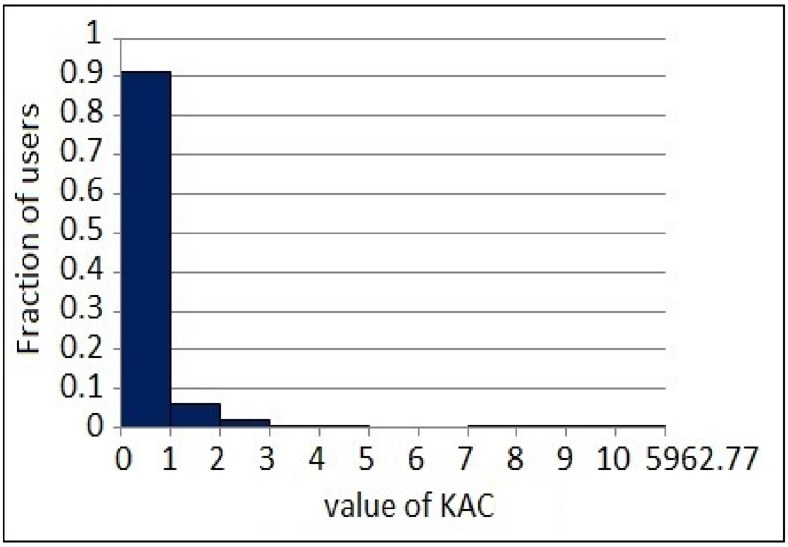

Figure 5.2: Distribution of KAC values for sybils with sparse graphs

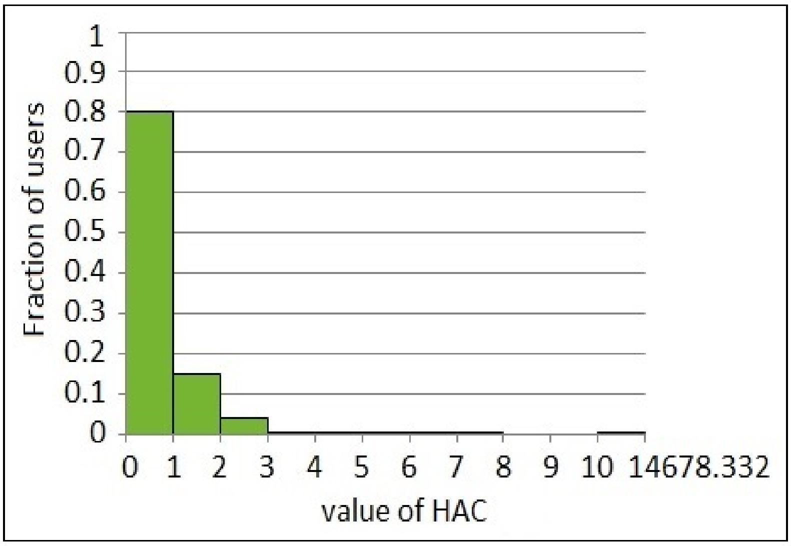

Figure 5.3: Distribution of HAC values for sybils with sparse graphs

On the other hand, the detection rate of LAC is higher than that of KAC and HAC. It shows that by combining these two anomalous change factors, we reach a better detection rate.

In order to calculate the false alarm rate, false positive (FP), we need to compute the number of legitimate users that are detected as anomalous. Calculating the false alarm rate is challenging since the real graph also may contain some anomalous users [6].

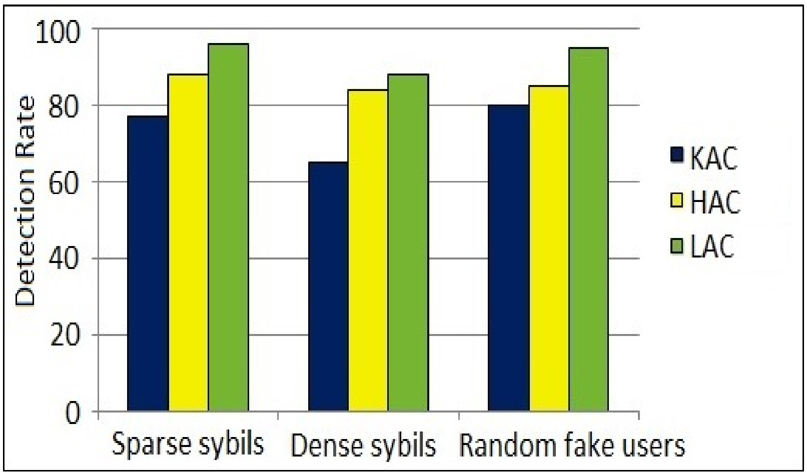

Figure 5.4: Detection rate for the three categories of injected fake users

However, according to many anomaly detection studies, the majority of users follow a common pattern and only few of them deviate from it [5]. Therefore, to select a set of users to be considered legitimate users, we randomly select users not from the whole graph but among those users whose graph structures are similar to the majority of users inside the real graph. In other words, we did not consider outliers for this selection. To perform this selection, we consider four structural features: degree, triangle count, average degree of user's friends and average triangle count of user's friends. Then, we analyzed the relation between these features. For example, Figure 5.5 shows the relation between degree and triangle count. We can see that most users concentrate in some areas and only few of them are outliers. Figure 5.6 illustrates the relation between the average degree and average triangle count and, again, it is clear that most of the users concentrated on some areas of the graph and few of them deviate as outliers. Therefore, we select our legitimate users randomly among those users with ratio between degree and triangle count and ratio between average degree and average triangle count in the range [0.1, 10], since 90% of users have this range of values. In particular, from each graph snapshot, we randomly select 10000 legitimate users and then, select 1000 common users in all graph snapshots as the final legitimate users.

Finally, we compute the false alarm rate for all these selected normal users. We can see in Figure 5.7 that false alarm rate of LAC is lower than that of KAC and HAC. On the other hand, the false alarm rate of sybils with dense friendship graphs is higher than the two other types of fake users.

In the last experiment, we calculate the F-measure by computing precision and recall based on: false positive (FP), false negative (FN), true positive (TP) and true negative (TN). Table 5.1 shows the F-measure for the three categories of fake users based on our three anomalous factors. We can see that the F-measure values for sybils with sparse graphs and random fake users are almost equal with a very good value and the performance is

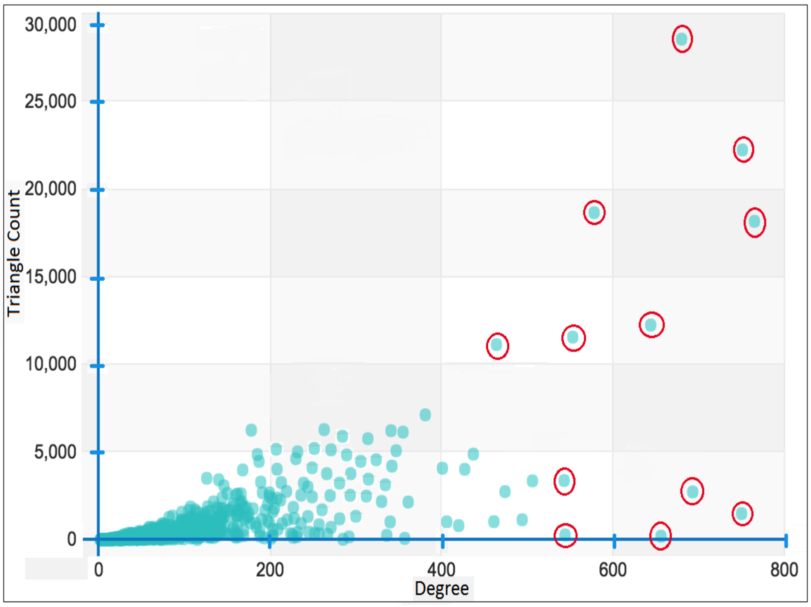

Figure 5.5: Relation between degree and triangle count

Anomalous Factors	Sparse Sybils	Dense Sybils	Random Fake Users
KAC	0.8695	0.7839	0.8883
HAC	0.93566	0.9124	0.9183
LAC	0.9790	0.9356	0.9737

Table 5.1: F-measure for three categories of fake users based on our three anomalous factors

better than the one for sybils with dense friendship graphs.

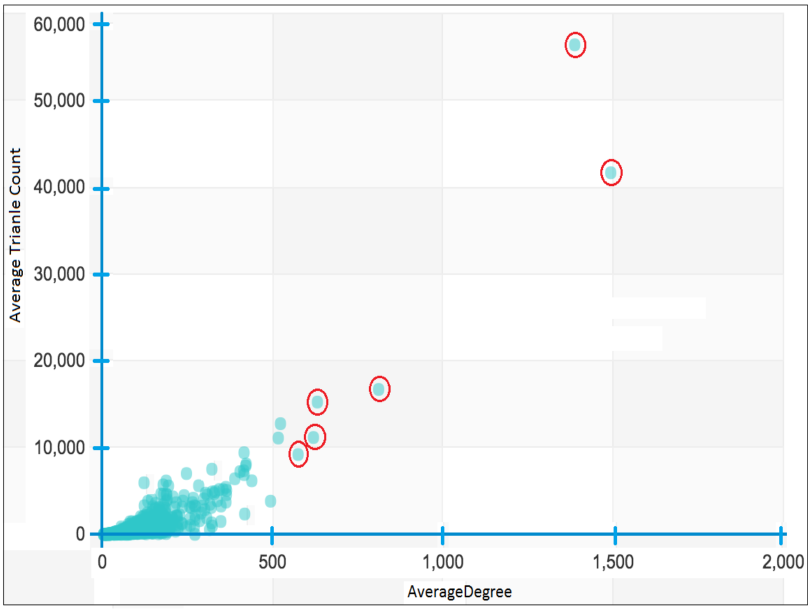

Figure 5.6: Relation between the average degree and average triangle count

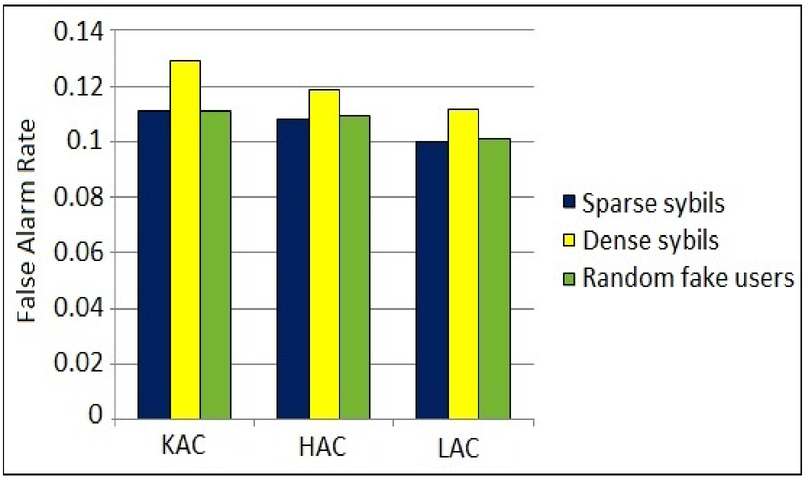

Figure 5.7: False alarm rate

Chapter 6

Gossip-based Behavioral Group Identification in DOSNs

6.1 Introduction

Recently, discovery of meaningful groups of users that share the same behavioral patterns in social networks has become an active research area. Behavioral group identification has many valuable applications. For instance, it can help in improving recommendation systems, it can be used for advertisement purposes, direct marketing, and for risk assessment in online social networks. The key idea in risk assessment is that the more the target user's behavior diverges from those of other similar users, the more the target user is risky [5]. Therefore, risk assessment approaches require to identify similar users that share the same behavioral patterns.

By considering a social network as a graph, each node is depicted as a user, and an edge connecting two users denotes the relationship between them. Here, the main problem is that all users are connected in friend to friend graph, but users that share the same behavioral patterns not necessarily have friendships in the graph. In grouping users we consider the profile and activity information such as age, gender, education, nationality, number of friends, activity level, etc. Furthermore, in investigating the discovery of behavioral groups, we have cast our attention to DOSNs [33]. In DOSNs, there is no central infrastructure and discovery of behavioral groups is more challenging than in the centralized setup. In the fully distributed social graph, each user can only communicate with his/her direct friends without sending all the private group identification feature values to his/her direct friends in a raw form. In more details, behavioral patterns can be classified into social and individual behavioral patterns. Social behavioral patterns rely on user interactions, while individual behavioral patterns are related to profile information [100]. In this chapter, we propose a methodology for identifying both social and individual patterns to group users in DOSNs.

The problem of finding similar users in social networks has been widely studied in the context of community detection. Community detection approaches that are pure link-based, relying on topological structures [38], [48], fail to group users with the same behavioral patterns in that such users might belong to different communities based on their friendship links. Moreover, some of the community detection approaches are content-based that they rely on the analysis of the content generated by each user [111], [94]. The major problem of these approaches is the overhead for building the graph, based on similarity measures, that is not suitable for real-time applications. On the other hand, when each user feature vector includes both discrete and continuous features, the various behavioral patterns may not be obvious by similarity measures and then, this identification can not be made correctly. However, there are some stream-based community detection methods suitable for real-time applications [90], [124]. But, most of these approaches are link-based [92], and they do not consider the personal feature vector of users.

To alleviate the limitations of existing approaches, we propose a fully decentralized clustering algorithm which is capable of clustering distributed information without requiring central control. The selected clustering model requires specific aspects to be considered such as: the final clustering model should maintain a reasonable performance compared to a centralized clustering model and should be robust in that it should not easily fail when some of the users leave the network or do not answer to messages. Also, all users should be able to have the final clustering model at any time after convergence to assign a group for themselves and their direct friends in a local way. Finally, we need to minimize the communication cost by decreasing the number of messages and the size of them as well. These requirements bring us to exploit Newscast EM [79], a probabilistic gossip-based randomized communication clustering approach, originally developed for clustering users in peer-to-peer networks. In Newscast EM, each user initializes a local estimation of the parameters of the clustering model (mean, standard deviation, etc.) and then, contacts a random user from all users in the network, to exchange their parameters estimation and aggregate them by weighted averaging. The choice of random selection is crucial to the wide dissemination of the gossip [79], since, the probability of a user being sampled is proportional to his/her degree [95]. Gossip-based peer-sampling service [70] provides a user with a uniform random subset of all users in the peer-to-peer network. But, the main difference between peer-to-peer and social networks is that in peer-to-peer networks each user can directly communicate with any other user in the network to exchange information. On the contrary, in social networks each user can just communicate directly with his/her direct friends. Therefore, we use the random-sampling implementation for social networks proposed in [76].

The main contribution of this approach is making Newscast EM to be applicable on top of DOSNs and apply it to identify behavioral groups of users. Our goal is to achieve an accuracy comparable to a centralized scheme. The advantages of this distributed behavioral group identification are: 1. the usage of both social and individual patterns of users, 2. feature values of users are never send over the network in a raw form and 3. it has low computation and communication cost. The remainder of this chapter is organized as follows. We first explain Newscast EM in Section 6.2. Then, we propose our gossip-based implementation for behavioral group identification in Section 6.3. In Section 6.4, we show the result of the clustering model.

6.2 Background

In selecting the clustering technique, we focused on soft clustering (i.e., probabilistic-based clustering). Hard clustering techniques, (e.g., k-means) are not proposing a solution to the problem of clustering discrete or categorical data [20] since they are based on distance metrics. We use EM (Expectation Maximization) algorithm, that is, a probabilistic based clustering method. In particular, EM defines K probability distributions to identify K clusters for all users based on their feature vectors, where each distribution represents the likelihood of those feature vectors to belong to a given cluster. In this way, EM first assigns a set of K *membership probabilities* to each user u based on his/her own feature vector. Then, EM maximizes these likelihoods by learning the parameters of the clustering model in order to assign to each user the cluster with the highest probability.

The main idea of distributed EM algorithm is that each user starts the Expectation-step with a local estimation of the parameters of the clustering model. Then, in the Maximization-step, the algorithm employs a gossip-based protocol to learn a final clustering model from these local estimations. Each user exchanges his/her own estimations with several other users by using a randomized communication protocol. By gathering these estimations from random users, the target user can update and re-estimate his/her own estimation.

In the following, we present a summary of Newscast EM. Interested readers are referred to [79] for more details. Let N be the set of users in the OSNs, the probability of membership or weight of a target user $\vec{u}$, $\vec{u} \in N$, in cluster l is defined as [20]:

$$w_l(\vec{u}) = \frac{w_l.p_l(\vec{u}|\theta_l)}{\sum_{i=1}^{K} w_i.p_i(\vec{u}|\theta_i)} \tag{6.1}$$

where, w_l is a weight computed as $w_l = \frac{|N_l|}{|N|}$, with N_l denotes the set of users belonging to the l^{th} cluster, where $\sum_{l=1}^{K} w_l = 1$; function $p_l(\vec{u}|\theta_l)$ is the component density function modeling the feature vector of the l^{th} cluster, where $\theta_l = \{\vec{\mu_l}, \Sigma_l\}$ represents the parameters for l^{th} distribution, that is, the mean and the covariance.

Newscast EM uses a fully distributed averaging process for estimating the parameters $\Theta = \{w_l, \vec{\mu_l}, \Sigma_l\}$, $l = 1, \ldots, K$. Assuming that each user has just one feature vector, then the Expectation-step implies that each user locally estimates the parameters based on his/her own feature vector. In this manner, each user u_i, $i = 1, \ldots, N$ starts with a local estimation of $\Theta_i = \{w_{li}, \vec{\mu}_{li}, \Sigma_{li}\}$ for the parameters of the cluster l. However, in the Maximization-step of the algorithm user u_i needs information from all users in the network to recompute his/her parameters estimation. Therefore, this step is implemented as a sequence of gossip-based cycles. The details of the algorithm, which each user will run in parallel is as follows:

Initialization phase: We assume that all users agree on the number of clusters K and start the exchanging protocol. Each user $\vec{u_i}$, sets the membership probability for each cluster as $w_l(\vec{u_i})$ to some random positive value and then normalizes all to sum to 1 over all l. This phase is completely local for each user.

Maximization-step: In this step, user u_i initializes the local parameters estimation for each cluster l as follows: $w_{li} = w_l(\vec{u_i})$, $\vec{\mu}_{li} = \vec{u_i}$ and $\tilde{\Sigma}_{li} = \vec{u_i}.\vec{u_i}^T$, where T is the transpose of the feature vector of user u_i. Then, user u_i for $\Re$ cycles repeatedly initiates the information exchange with random users, i.e., u_j. Then, users u_i and u_j update their local parameters estimation for each cluster l as follows:

$$w_{li}^{'} = w_{lj}^{'} = \frac{w_{li} + w_{lj}}{2} \tag{6.2}$$

$$\vec{\mu}_{li}^{'} = \vec{\mu}_{lj}^{'} = \frac{w_{li}.\vec{\mu}_{li} + w_{lj}.\vec{\mu}_{lj}}{w_{li} + w_{lj}} \tag{6.3}$$

$$\tilde{\Sigma}_{li}^{'} = \tilde{\Sigma}_{lj}^{'} = \frac{w_{li}.\tilde{\Sigma}_{li} + w_{lj}.\tilde{\Sigma}_{lj}}{w_{li} + w_{lj}} \tag{6.4}$$

Expectation-step: User u_i, after waiting for $\Re$ cycles for the convergence of his/her local parameters estimation, computes new membership probabilities for each cluster l using the Maximization-step estimations w_{li}, $\vec{\mu}_{li}$ and $\Sigma_{li} = \tilde{\Sigma}_{li} - \vec{\mu}_{li}.\vec{\mu}_{li}^T$. We denote with Θ^t the parameter values set at iteration t and then $\Theta^{t+1} = \{(\vec{\mu}_{li}^{t+1}, \Sigma_{li}^{t+1}, w_{li}^{t+1}), l = 1, \ldots, K\}$. The sequence of Θ-values which is then the likelihood of Θ, $L(\Theta)$, is non-decreasing at each iteration. Then, user u_i checks the stopping tolerance by using the estimations from the previous EM-iteration to see if it is satisfied or not, until $|L(\Theta^t) - L(\Theta^{t+1})| \leq \varepsilon$, where $\varepsilon > 0$. If it is not satisfied, the Maximization-step is repeated, unless a stopping tolerance is satisfied. In the following, we will explain how to run newscast EM in decentralized social networks.

6.3 Newscast EM in DOSNs

In social networks, each user can just communicate directly with his/her direct friends. Therefore, we propose our gossip-based clustering framework on DOSNs that contains two main components: UserSelection and ClusteringModelUpdate. The same algorithm is run by each user in the network in parallel, as shown in Algorithm 1.

6.3.1 User Selection

In randomized user selection, the problem is that if users can be selected randomly with equal probability, the estimation will be unbiased. But, it is well known that the probability of a user being sampled is proportional to his/her degree [95]. Therefore, more popular users have a higher degree and tend to have a higher probability of being sampled. This will lead to overestimate the average value during the gossip process. There are plenty of approaches to select a uniform and unbiased random sampling of users in DOSNs such as: graph traversal techniques [81] and Random Walk [80]. But, most of these approaches are biased towards high degree users [95]. To have a uniform random sample of all users, each user needs to know every other user in the network. Since, accessing each user in the network to gossip with, is unrealistic in a large-scale dynamic networks, we apply a technique for DOSNs proposed [76] for randomized communication, to define a dynamically changing random graph topology over the network. This technique includes two methods *Initialization* and *SelectUser*.

The initialization procedure initializes the service for the new user when he/she joins the social network. First, each user u_i maintains a list of its direct friends and two hops friends in a small fixed size cache, called *Random Neighbors Cache* (*RNC*), including e entries. The set e of entries in the *RNC* contains the list of random users ID, their *longevity* field, and the path to reach them. The field *longevity* is the age of the entry since the moment it was created by the user. Then, in the *SelectRandomEntries()* procedure, a user selects S subset of neighbors from *RNC* and puts them in a cache, called Exchange Cache (*ExC*). After that, the target user u_i continuously selects one of his/her neighbors with the highest *longevity* from *RNC*, i.e., u_j, in *SelectRandomUser()* procedure, to exchange entries of *ExC*. Then, u_i increases by one the *longevity* of all the other entries in u_i's *RNC*.

In social networks, as shown in Figure 6.1, if user A wants to communicate with user D, he/she needs to reach first B and then C. Therefore, each user needs to maintain both a set of random users ID and also the path to reach them in order to exchange the entries of *ExC*. For instance, let us assume that in Figure 6.1.(a) user A has six entries in the *RNC* that include C, I, K, D, N and B. User A selects user I with the highest *longevity* among all neighbors in the *RNC* and wants to exchange S (i.e., 3 in this example) number of his/her neighbors in the *ExC*, such as C, D, N, with user I. On the other hand, user A needs to pass through (M, L) to reach user I. Therefore, user I, in order to reach all of the entries in the *ExC* of user A, needs to first reach user A and then he/she will be able to reach all entries inside A 's *ExC*. The problem is that this path could be long. For instance, user I, in order to reach user D, first needs to reach user A by passing through (L, M) and then from A to user D via (B, C), i.e., (L, M, A, B, C). This path is long and it is not the shortest path from user I to D. But, user I can reach user D by passing through his/her mutual friends with D, like (E ,D) or (L, D).

More precisely, to decrease the communication cost, the protocol in [76] builds a new path for user I to reach all entries in user A's *ExC* during the exchanging process, illustrated in procedure *UpdateExC()* in Algorithm 1. In this way, the source user A asks all users within the direct path from A to I, i.e., (M, L, I) to build a new path for all the entries to be accessible for user I. This process in summary is as followings. First the source user (A) only adds his/her ID to the first part of the address of each entry in *ExC* and sends the *ExC* to the next user (M) on the path towards user I, as shown in Figure 6.1.(1) in red color. Then, every next user (for instance user M) within the path towards user I and also user I him/herself, after receiving the *ExC*, first reverses the path of each entry in *ExC*, as shown in Figure 6.1.(2).a and starts traversing all users inside the reverse path to check, if he/she has any direct friendship or mutual friends with those users. If yes, he/she removes the remaining part of the path and adds his/her friends or mutual friends ID to the path and adds the ID of his/herself to the first part of the path (except user I that does not need to add his/her ID to the first part of the path), as shows in Figure

6.1.(2).b (the first and second row of ExC). If not, he/she keeps the path and just adds his/her ID to the first part of the path as shown in Figure 6.1.(2).b (the third row of ExC).

Then, all other users within the path towards user I, i.e., user L, do the same and send the ExC to the next user towards user I. When user I receives the final ExC, checks all entries in the ExC and updates them in his/her own RNC. Then, user I replies by selecting a subset of his/her ExC entries, updates the entries path and, then, sends them to user A from the path (L, M, A).

After exchanging neighbors for a number of cycles, the service will converge to a random overlay where each user connects to a uniform random subset of all users currently in the network. But, in our framework, users in addition of exchanging neighbors, also exchange their parameters estimation of the clustering model. In this way, when the service converges to the random overlay, also converges to a final parameters estimation available for all users and then, users are able to update their local parameters estimation. After some iterations of the EM algorithm, they will converge to a final clustering model. In the next section, we will explain in more details the update of the clustering model.

6.3.2 Clutering Model Update

The second main component of our framework is the online clustering algorithm that updates the clustering model based on the local parameters estimation of each user. In our setting, in the set e entries of the user u_i's RNC, in addition to the list of random users ID, their *longevity* field, and the path to reach them, we maintain their parameters estimation of the clustering model. Therefore, the gossip-based clustering algorithm shown in Algorithm 1 performs the following steps. In the initialization phase, each user u_i, in addition to filling RNC with direct friends and two hops friends, initializes the local parameters estimation for each cluster l. After the initialization phase, users initiate exchanging neighbors and the parameters estimation of the clustering model simultaneously and periodically at a fixed period $\triangle T$. We do assume that the length of the period $\triangle T$ is the same for all users. During a period $\triangle T$, each user initiates one exchanging cycle. There are two types of communication models for exchanging the information. In the Push based model, a target user u_i sends ExC and parameters estimation (Θ_i) to the selected user. In the Push-Pull based model, both the target user u_i and selected user u_j exchange their ExC and parameters estimation. Our communication model is based on Push-Pull, since the Push approach can easily lead to partitioning the set of users in the network [70], [76].

After initialization, the initiating user u_i increases by one the *longevity* of all neighbors in his/her RNC. After that, user u_i selects neighbor u_j with the highest *longevity* among all neighbors in RNC, and set the *longevity* of u_j to zero in his/her RNC. If the information has to be pushed, user u_i replaces u_j's entry in RNC with a new entry of *longevity* 0 and with u_i's ID and path to reach user u_i. Then, user u_i selects S subset of neighbors from u_i's own RNC, and save them in the temporary ExC. Next, user u_i updates the entries path in the ExC by building a new path as mentioned in procedure *UpdateExC()* in Algorithm 1, and sends it to the next user in the path towards user u_j. After that, user u_i sends his/her local parameters estimation to the next user u_k in the path towards user

Algorithm 1 Gossip based Clustering Protocol

Input: The local Θ_i for each user u_i, $i = 1, \ldots, N$.
Output: The global Θ_i for each user u_i.

```
Initialization()
Loop:
if Push then//if $u_i$ has to push information
    Wait $\triangle T$
    ExC ← RNC.SelectRandomEntries()
    ExC ← UpdateExC($ExC, u_i$)
    $u_j$ ← RNC.SelectRandomUser()
    Sends ExC to $u_i$ 's neighbor toward $u_j$
    Sends $\Theta_i$ to $u_i$ 's neighbor toward $u_j$
    $\Theta_j \leftarrow Receive(u_j)$
    $\Theta_i \leftarrow UpdateModel(\Theta_i, \Theta_j)$
else if Pull then//If $u_j$ has to reply $u_i$
    ExC ← RNC.SelectRandomEntries()
    Sends Updated ExC to $u_j$ 's neighbor toward $u_i$
    Sends $\Theta_j$ to $u_j$ 's neighbor toward $u_i$
    Receives ExC from $u_j$ 's neighbor
    ExC ← UpdateExC($ExC, u_j$)
    Receives $\Theta_i$ from $u_j$ 's neighbor
    $\Theta_j \leftarrow UpdateModel(\Theta_i, \Theta_j)$
else//Pull user $u_k$ within the path
    ExC ← UpdateExC($ExC, u_k$)
    Sends ExC to $u_k$ 's neighbor toward $u_j$
    Sends $\Theta_i$ to $u_k$ 's neighbor toward $u_j$
End Loop
procedure INITIALIZATION()
    InitModel()
    InitRNC()
    return $\Theta$, RNC
End procedure
procedure UPDATEMODEL($\Theta_i$, $\Theta_j$)
    for each cluster $l$ do
        $w_l^{'} = (w_{li} + w_{lj})/2$
        $\vec{\mu}_l^{'} = (w_{li}.\vec{\mu}_{li} + w_{lj}.\vec{\mu}_{lj})/(w_{li} + w_{lj})$
        $\tilde{\Sigma}_l^{'} = (w_{li}.\tilde{\Sigma}_{li} + w_{lj}.\tilde{\Sigma}_{lj})/(w_{li} + w_{lj})$
    return $\Theta^{'}$
End procedure
procedure UPDATEEXC(RxC, $u$)
    UpdatedRxC= RxC
    if $u = u_i$ then
        for all entries path ∈ UpdatedRxC do
            Path.AddFirstID()
    else//($u = u_j$) or ($u = u_k$ within the path to $u_j$)
        for all entries path ∈ UpdatedRxC do
            ReversedPath=path.Reverse()
            for all users ID ∈ ReversedPath do
                if ID ∈ Direct-Friends ($u_k$) then
                    Path.AddFirstID(ID)
                    Break
                else if ID ∈ Two-Hop-Friends ($u_k$) then
                    Path.AddFirstID(ID)
                    Path.AddFirstID(GetDirectFriend(ID))
                    Break
                else
                    Path.AddFirstID(ID)
    if $u = u_k$ then
        for all entries ∈ UpdatedRxC do
            Path.AddFirstID(u)
    return UpdatedRxC
End procedure
```

u_j. Later, all users in the path towards user u_j update the entries path in the ExC and send the updated ExC and parameters estimation received from u_i to the next user in the path towards user u_j.

When user u_j receives from one of his/her direct friends the ExC coming from user u_i, user u_j replies by selecting a random subset of S neighbors of his/her own RNC and save them in his/her ExC. Next, user u_j updates the entries path in ExC by adding his/her ID to the first part of the entries path and sends ExC to the next user on the path towards user u_i. After that, user u_j sends the local parameters estimation to the next user in the path towards user u_i. Then, user u_j updates the entries path in the received ExC. Next, u_j discards entries pointing at u_j and entries already contained in u_j's RNC and updates his/her RNC to include all remaining entries, by firstly using empty cache slots, and secondly replacing entries among the ones sent to u_i. User u_j set *longevity* to zero for all entries of received ExC in the RNC and does not increase, though, any entry's *longevity* in the RNC until he/she initiates the exchanging process. After that, user u_j updates his/her own parameters estimation by calculating the weighted average with the parameters estimation coming from u_i. Finally, user u_j updates the parameters estimation of u_i and u_i's ID and the path to reach u_i inside his/her RNC.

Under this algorithm, after some cycles, the local parameters estimation of users converge exponentially fast to the global parameters estimation in each Maximization-step of the EM algorithm. Therefore, each user is able to compute new membership probabilities and check the stopping tolerance. Each user maintains the newly updated parameters estimation from the previous EM-iteration in a small cache, called *Estimation Cache* (EsC), of fixed size. When the cache is full, the parameters estimation stored for the longest time is replaced by the newly added parameters estimation. Therefore, after some iterations of clustering, all users will have a final clustering model and compute a final membership probability to belong to their most fitting cluster.

6.3.3 User Behaviour-based Group Identification in DOSNs

Our goal is to have similar users in each group based on their social and individual features. Feature vectors of each user are given as input to the Algorithm 1. Therefore, each user can assign a cluster number (behavior group) to him/herself based on the feature vector by considering the maximum membership probability among them.

User's features vector includes two types of features: individual and social features. Individual features are those, like age, gender, but also those that impact the possible users' behaviors, like, education and nationality. In addition to individual information, in order to measure users' attitude in online socialization, we consider the following social features:

Number of Friends: social users with a lot of friends have different patterns than isolated users with few friends;

Activity Level: unlike active users that write a lot of posts, passive users do not send any information to others. We calculate this feature as the sum of: a) number of posts that a user sends to others from the first day of joining the community, b) number of likes that a

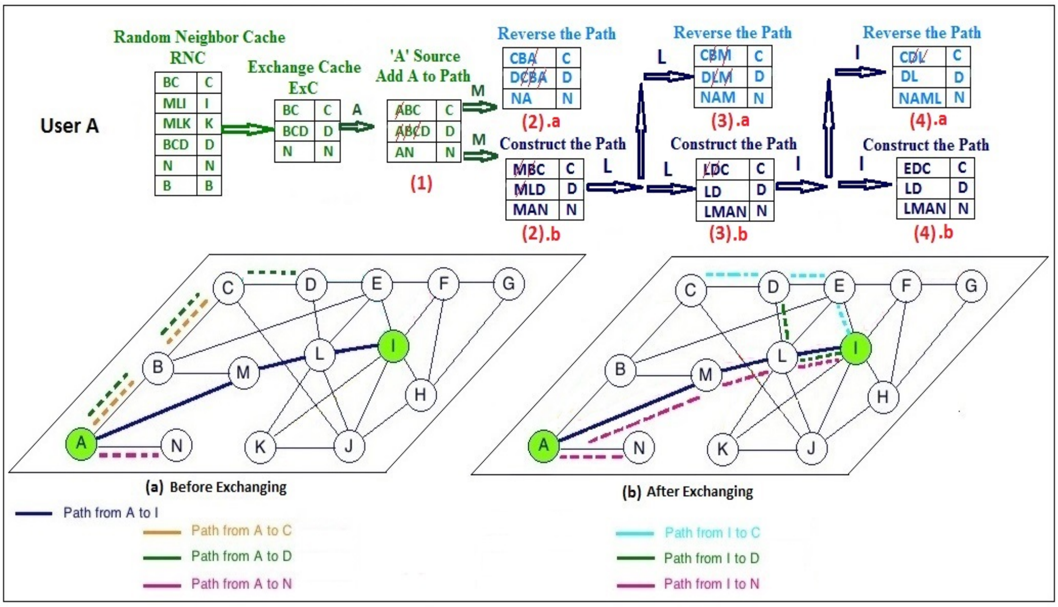

Figure 6.1: (a) before and (b) after the neighbours exchanged between A and I

user performs on posted items, and c) number of comments a user writes for posted items; *Percentage of public profile items:* the assumption is that users with all profile information (100%) public are more social.

6.4 Experiments

To perform the experiments on a real graph, we used the Facebook dataset crawled and used in [3]. The author crawled the profiles information and friendship links of 75 users that launched the application as seed. Then, the application crawled the information related to these 75 users' friendships. We removed those profiles that have many missing features and obtained a graph by considering the largest connected component which includes 13,000 user profiles, plus the 75 seed users, with a total of about 461,700 friend links, 6,150,892 likes and 1,742,709 comments. Totally, around 7,000 users have more than 75% profile information as public.

6.4.1 Results for convergence of the clustering model

The experiments are ran with $S = 50$ [70]. During the exchanging process the mean of the local parameters estimation,
$mu_{i,c}$ of each user u_i, $i = 1, \ldots, N$ for each cluster in cycle c, $c = 1, \ldots, \Re$ is always the global correct mean μ_c, but, the variance measure σ_{c^2} that expresses the deviation of the local estimations from the correct mean in the given cycle c decreases over the set of all local estimations on the average by factor γ, with $\gamma < \frac{1}{2\sqrt{e}}$ [70]. In general, when the

variance tends to zero, then all users hold the global correct mean μ_c.

$$\mu_c = \frac{1}{N}\sum_{i=1}^{N}\mu_{i,c} \qquad \text{and} \qquad \sigma_c^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\mu_{i,c} - \mu_c)^2$$

The convergence factor between cycle c and $c+1$ is given by $\sigma_{c+1}^2/\sigma_c^2$. We plot the convergence factor (values are averages for 20 independent runs) as a function of the number of cycles, as shown in Figure 6.2. It is clear to see that the speed of the convergence of the variance is fast and it decreases exponentially after few cycles. Thus, means that, after a small number of cycles, all users, including the isolated users with low friendship links, will have accurate estimations of the global correct mean μ in each M-step, when no failures occurred. From this experimental result, choosing the number of cycles $\Re$ equal to 120[1] is sufficient to reach a convergence.

6.4.2 Coping with User Failure

In a dynamic network, users continuously join and leave the network and they fail in some situations. In this section, we consider the performance of the clustering model when some percentage of users fail in each cycle of the exchanging parameters estimation.

As we mention before, each user maintains the parameters estimation he/she receives from other users in the network in his/her ENC. If a user u_i sends his/her parameters estimation to a user u_j to exchange and performs averaging and, waiting for $\triangle T$ time, he/she does not receive any answer, u_i checks the ENC to verify if he/she has the parameters estimation of user u_j from previous cycles or not. If yes, u_i performs the average with those previous values and updates his/her parameters estimation. Otherwise, he/she skips the exchanging step. We need to mention that the user selection method in [76] takes care of the failure of those users within the path between user u_i and u_j. We consider the effect of these missing exchanges on the final value of the global μ of the clustering model. Towards this goal, we remove 50% of users in each cycle of the exchanging protocol and run independently the Newscast EM for 20 times. We show the result in Figure 6.3. The y-axis shows the variance of the 120th cycle to the first cycle. The figure shows that if the failure happens in the first cycles, the result of the global μ of the clustering model is so far from the real correct global μ. But, if this failure happens in the next cycles (especially after the 100th cycle), the variance tends to zero.

6.4.3 Clustering Results

In Table 6.1, we can see the performance of randomly initialized Newscast EM and centralized EM, by setting different number of clusters. The result in this table shows the average number of EM iterations for each user to achieve convergence in Newscast EM (around 42 ± 3) and centralized EM (around 38 ± 2) that are almost near. After convergence, the value of μ_i for each cluster at each user will converge to the true global μ. For example,

[1] We assume that all users agree on the number $\Re$ of peer sampling cycles, and $\Re$ is large enough to guarantee convergence to the final parameters estimation.

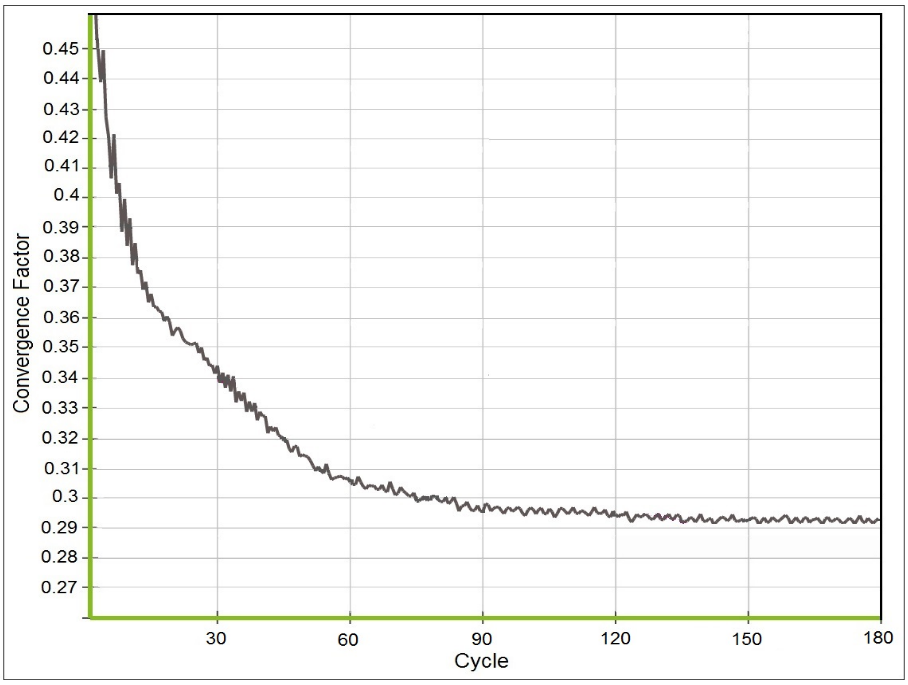

Figure 6.2: Convergence factor after users failure

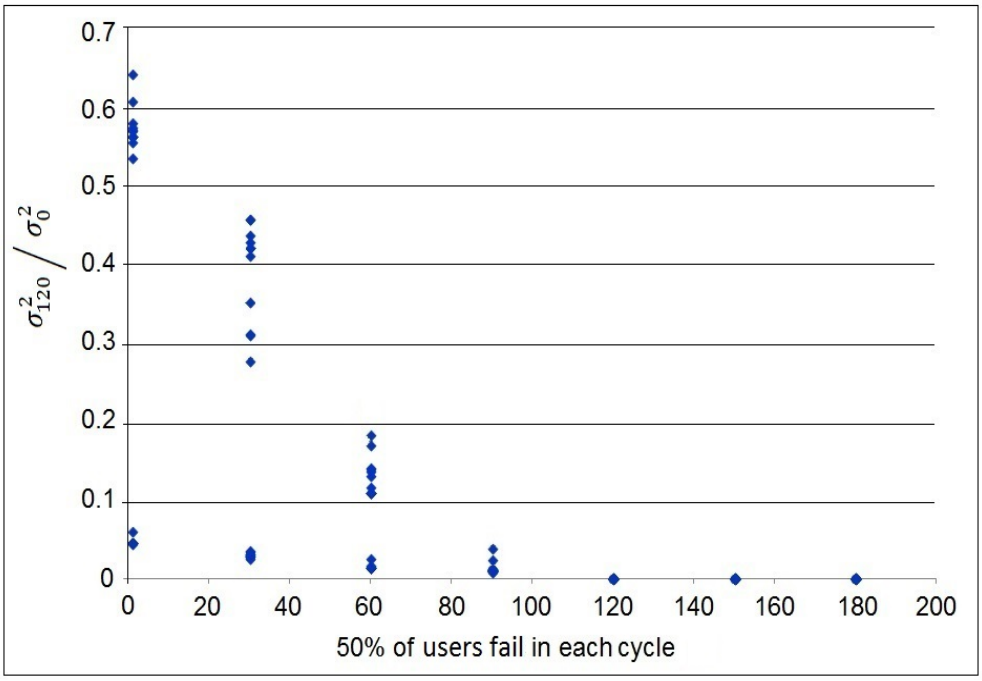

Figure 6.3: The variance of μ at cycle 120 (for 20 independent run) after the failure of 50% of users in each cycle

Table 6.2 shows this value for feature Activity level in centralized EM and Newscast EM that are almost equal. Also, Table 6.3 shows the values of μ for each cluster for the feature Number of friends.

EM Iteration		
Number of Clusters	Centralized EM	Newscast EM
3	38	41.25
5	38	42.53
7	39	41.38
10	38	43.19

Table 6.1: EM iterations for different number of clusters

Cluster No	$\mu(AL) - CentralEM$	$\mu(AL) - NewscastEM$
C1	27.5	27.35
C2	52.53	52.25
C3	48.31	48.05
C4	11301.68	11299.987
C5	6869.46	6868.299
C6	335.83	334.062
C7	18338.18	18336.647

Table 6.2: The value of μ for the feature Activity Level

Cluster No	$\mu(NF) - CentralEM$	$\mu(NF) - NewscastEM$
C1	536.39	535.56
C2	272.91	271.47
C3	194.42	193.39
C4	438.35	436.04
C5	276.87	275.41
C6	963	961.93
C7	953.6	951.58

Table 6.3: The value of μ for the feature Number of Friends

6.4.4 Dominant User Behaviors

Identifying the number of clusters or user behaviors is related to the nature of the dataset. Therefore, there is no correct or incorrect numbers of groups to find. Each increment in

Cluster No	10 clusters	7 clusters	5 clusters	3 clusters
Cluster1	20.27%	32.43%	47.29%	41.89%
Cluster2	20.27%	21.62%	22.97%	32.43%
Cluster3	14.86%	18.91%	10.81%	25.67%
Cluster4	10.81%	9.45%	12.16%	
Cluster5	9.45%	6.75%	6.75%	
Cluster6	6.75%	5.4%		
Cluster7	1.35%	5.4%		
Cluster8	5.4%			
Cluster9	5.4%			
Cluster10	5.4%			

Table 6.4: The ratio of users (RU) in each cluster for different number of clusters

the number l of clusters yields to a new group and similarly a new behavior. On the other hand, if we consider a small number of groups by aggregating some behaviors, we will have the most dominant behaviors, but, we may miss some relevant behaviors. In this chapter, we assume that we know the best number of clusters. However, we analyze the quality of the clustering for various values of l. In Table 6.4, we report the ratio of users that belong to each cluster, by considering different numbers of clusters. When we have 10 clusters, the percentage of users that belong to the 7th cluster is 1%, that is, very small in size. By setting the number of cluster to 5, we will have around 50% of users concentrated in one group (1sh cluster) that will be a huge cluster, and we are not able to precisely identify their behavioral pattern. For the number of clusters equal to 3, we will have the most dominant behaviors, but, we will miss a lot of behavioral patterns. Therefore, we consider the number of clusters equal to 7 in the rest of the experiments.

In the next experiment, we analyze the influence of each feature value on the quality of the discovered groups. Then, among all features, we remove those features that have the same distribution in all clusters. More precisely, we discard those features whose existence in the clustering will not propose a new behavioral group. In order to have a measure for the distribution of each feature value for all clusters, we use the inter-group relative feature value [100], denoted by $Related_{fl}$, which measures how a feature f of cluster l is related to the same feature of the other clusters. It is computed as follows:

$$Related_{fl} = \frac{feature_{fl}}{\sum_{l=1}^{K} feature_{fk}} \tag{6.5}$$

where $feature_{fl}$ is the value of feature f in cluster l. This allows us to see if the value of a feature is evenly distributed in all clusters or concentrated in a single cluster. For example, we can see in Figure 6.4 the distribution of all categorical features such as age, number of friends and activity level. This figure shows that the distribution of the feature **age** in all clusters is nearly the same. Then, we remove this feature since it can not help us to define

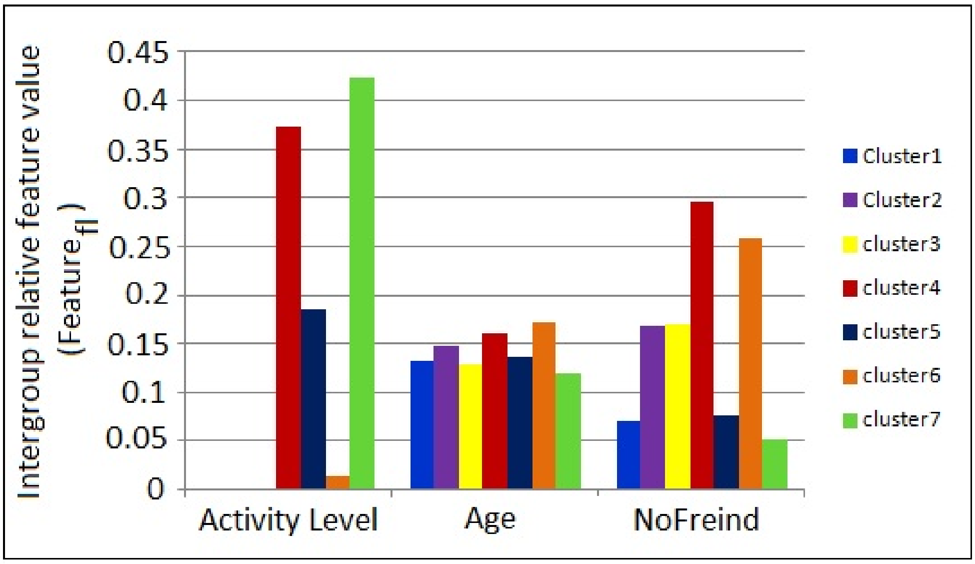

Figure 6.4: The distribution of Categorical features in each cluster

any behavioral pattern. For discrete features, such as: gender, nationality and percentage of public profile items, we consider the fraction of users that have the same value for that feature in each cluster as shown in Figure 6.5. We removed **gender** and **percentage of public profile items** and we consider all the remaining features.

Based on the experimental results, we are able to find the most dominant behaviors, that are shown in Figure 6.6. These dominant behaviors are categorized as follows: 1)*most active users with high number of friends:* users in cluster 7 have the highest activity level and the highest number of friends. These users have a bachelor or master degree and all of them are from Italy. 2)*very active users with medium number of friends:* these users are concentrated in cluster 4, have a medium activity level and a medium number of friends. Their education level is either elementary school, bachelor or PhD and they are from all countries except Italy and USA. 3)*active users with low number of friends:* these users are concentrated in cluster 5, and their education level is either diploma, bachelor or master and they are from all countries except USA. 4)*passive users with high number of friends:* these users are concentrated in cluster 6. These users have either a bachelor or PhD degree. They are from all countries except Italy. 5)*very passive users with medium number of friends:* these users are in cluster 1. These users are from all education levels. They are from all countries except Italy. 6)*most passive users with lowest number of friends:* these users are concentrated in cluster 3. Their education level is either bachelor or master and they are from all countries except USA. 7)*most passive users with low number of friends:* these users are in cluster 2 and most of them have bachelor. They are from all countries.

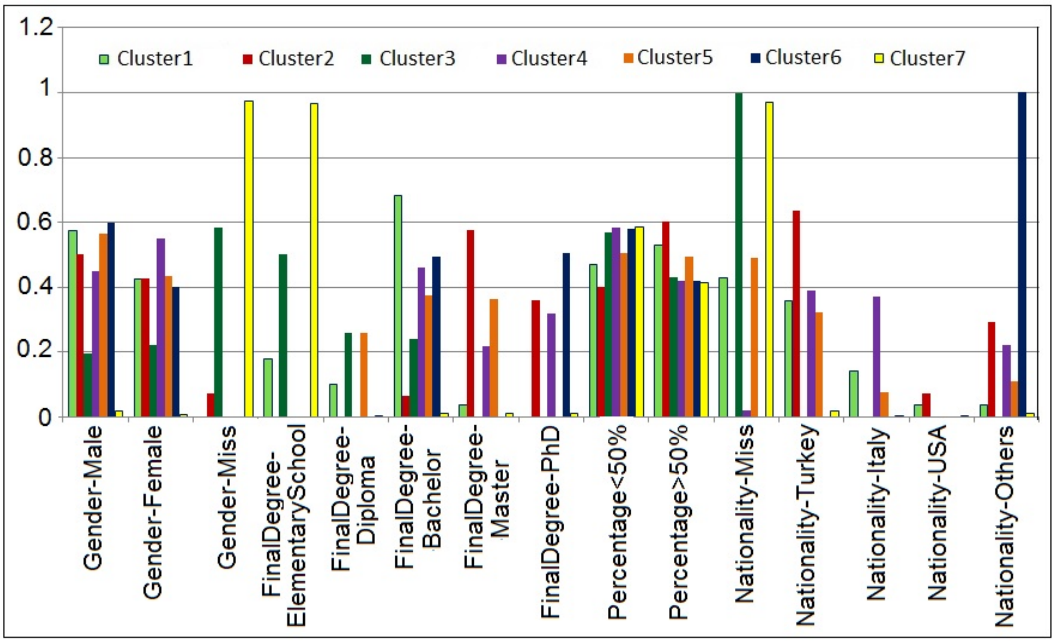

Figure 6.5: The distribution of Discrete features in each cluster

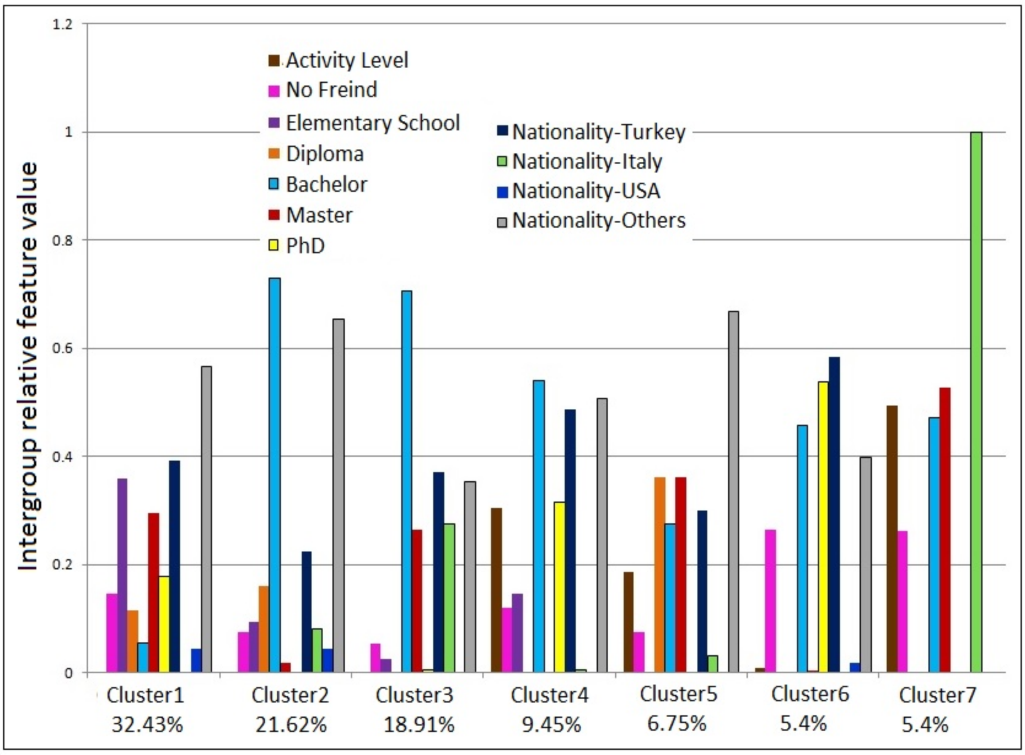

Figure 6.6: The most dominant behaviors with percentage of users in each group

Chapter 7

Distributed Two-Phase Risk Assessment in DOSNs

7.1 Introduction

In DOSNs, each user has a single feature vector including his/her interactions and personal information and these local information cannot be moved to a central server or to other users in a raw form due to privacy issues. However, this can attract a variety of privacy and highly damaging attacks. These attackers forward spam and malware on online social network. Since attackers have weird behavior pattern in the network, our goal is to analyze the behavior of users (interactions or activity patterns) in DOSNs by identifying those risky users whose follow the weird behavioral pattern of attackers. More precisely, when the user behavior diverges from 'normal behavior', the user will be considered as risky.

There are several attack detection solutions for specific kind of attacks [15, 14, 152, 157, 47]. On the other hand, although there is a unique centralized approach to cope with several kinds of attackers and risky users in OSNs [85], risk assessment over fully distributed data poses an important problem in this decentralized social network. Besides, this social network is also dynamic (users can join and leave). Therefore, we propose a unique decentralized approach that helps users to have a global understanding of risky users and able to detect several types of attackers in DOSNs. We believe that such a solution is a mechanism able to assign a risk score to the direct contact of each DOSN's user and help users to make a decision for answering friend requests from strangers. Moreover, risk assessment in DOSNs will be useful for those users desire to minimize risks by inspecting other users and to know which users are risky in the network and provide them a safe environment to create their account and do interactions.

Therefore, we propose a solution that enables a target user to assign a risk score to other users which send friend request to him/her and also his/her direct contacts by considering their activities and friendship patterns in the network. The goal is to compare the behavioral patterns of users with other similar users in the network to find misbehaviors. First, we model a 'normal behavior', based on the principle that similar users in DOSNs following the same behavioral patterns. For example, users with the same activity level, gender, education and country tend to have the same behavioral patterns in DOSNs [7, 140]. In order to model the normal behavior and measure the divergency of each user's behavior from the normal behavior and compute the risk score, we define several behavioral models, modeling similar users that share the same behavioral patterns. The goal is defining the behavioral patterns of users that are meaningful for risk assessment. For instance, users in a DOSNs doing a verity of activities such as posting or sharing items, writing comments on user's post, performing likes on posted items and sending friend requests to other users. For defining behavioral patterns, our goal is monitoring those activity of users that might reveal risky behaviors. As an example, having a high number of friends and posting a lot of items can not be considered as a risky behavior. However, if user creates a high number of friendship in a short period of time or shares a lot of items in the networks with receiving on average a high number of likes on all of them, this can be considered risky. Because, in some kind of attacks like socware, the attackers propagate malware or spam in the network in the way that when other users click on the shared item, it will be shared automatically on the user's page. Therefore, we review the activity patterns of several kind of attackers in OSNs in order to extract meaningful behavioral features for risk assessment in a distributed manner.

More precisely, we propose a distributed two-phase risk assessment approach by grouping users in the first phase based on their group identification features. This is achieved by exploiting a distributed probabilistic clustering technique available for all users over a set of user features meaningful for group identification, called *GI features*. Then, in the second phase, each user need to build one or more behavioral models for his/her identified group and other groups by defining various user features to model normal/anomalous behaviors. In both two phases we apply a distributed gossip based randomized probabilistic-clustering technique.

A key problem in our distributed risk assessment model is how to minimize the communication overhead and energy consumption in the network. We propose a gossip-based approach that involves multiple users with their local parameter estimations of the clustering model and exchange these estimations between themselves over the social network in parallel and update them. The basic idea is that all users periodically exchange these local parameters estimation of the clustering model with each other and finally converge to a global clustering model.

Thus, in the first phase all users exchange the parameters estimation of the first clustering model based on their group identification features. The result of the first phase is assigning a group number to each user in a distributed manner. After that, each user will have the group number for him/herself and also able to assign a group number to his/her direct friends based on their group identification features. As depicted in Figure 7.1, once groups have been identified, each users is able to determine one or more normal behavioral models for his/her group and other groups as well.

Thus, in the second phase users again starting to exchange the parameters estimation of the second clustering model, only exploit behavioral features and update his/her parameters estimation while he/she received the parameters estimation from users in the same group

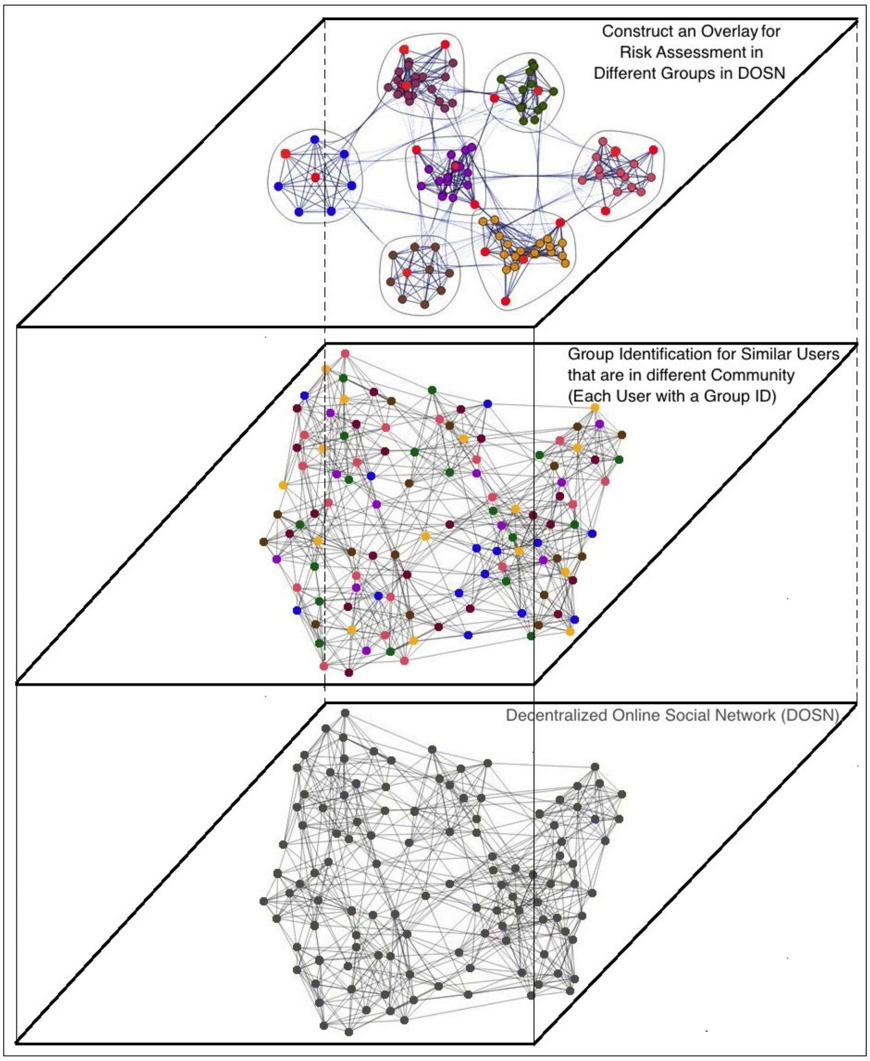
Construct an Overlay for Risk Assessment in Different Groups in DOSN
Group Identification for Similar Users that are in different Community (Each User with a Group ID)
Decentralized Online Social Network (DOSN)

Figure 7.1: Two phase risk assesment

and maintains the parameters estimation from users in different group. Finally all users will converge to a second global clustering models available for all users. The goal of the second phase is creating behavioral models for all users in each group identified in the first phase. The clusters obtained as the result of the second phase allow the target user to associate a risk score to his/her direct friends by calculating the divergency of his/her behavior from each identified behavioral model (see Section 7.5 for more details).

Then, each user is able to assign a group number based on the first clustering model by considering the GI features and then, an anomalous score based on the second-phase clustering models to his/her direct friends by considering their behavioral features. In order to evaluate our distributed scheme, we implement our algorithm in a real Facebook data set. Our goal is to achieve a comparable accuracy compared to a centralized scheme. The remainder of this chapter is organized as follows. Section 7.2 introduces the overall idea underlying our approach, whereas Section 7.3 provides a summary of Grouping and behavioral features. Then, we summarize the centralized solution for risk estimation on OSNs in 7.4. After that, we propose our gossip-based implementation for two-phase risk estimation in Section 7.5.

7.2 Overall Approach

Risk assessment over fully distributed data poses an important problem in DOSNs. All users have some features, like number of friends, posts, likes, comments, average number of mutual friends, etc where these feature values never leaves the computer of a user in a raw form due to the privacy considerations. Furthermore their feature values can also change over time. In addition, this social network is dynamic (users can join and leave any time).

Solving all these problems for risk assessment in DOSNs by monitoring (analyzing) the activity pattern of users in a distributed manner is a grand scientific challenge. The most naive approach is simply to build and use local learning models for each user [65]. But, the performance of the local learning models improves slowly due to the lack of data records [65, 128]. There are another approach that share data records between the users to have more data samples [155]. However, this leads to intensive communication among them, which degrades the scalability [12] and also, users need to share all private information with each other that is a big issue in some applications. However, there is a graph partitioning approach that is applicable for scalable social graph, but, each user requires to access the data of both his/her direct friends and a small subset of random users in the graph [124]. But, in our application personal information of users can not be moved to their friends duo to the privacy issues. On the other hand, their approach is link-based and they didn't consider the feature vector of users.

Another approach is to organize the computations in a hierarchical fashion [128, 45, 65] by which local learning models are computed first, and sent to a logically higher-level user that aggregates models, and then, returning results to the lower-level users for further processing [93]. In such approaches, output from the algorithm is much dependent on the processing of the highest level user. On the other hand, personal behavioural data records of users are so sensitive, so it is essential to process them locally. But, it is not possible to learn from local models because of the lack of information. On the contrary, there is no possibility to build local learning models and combining them. Besides, the communication cost needs to be kept low during our learning process.

To alleviate the limitations of the existing approaches, we apply a fully decentralized two-phase risk estimation based on distributed clustering algorithm without requiring central control. This distributed risk estimation should have specific characteristics include: the performance of the risk model should be comparable with the centralized risk model. Also, the risk model has to be robust and it should maintain a reasonable performance even in extreme failure when some of the users do not answer to messages. More over, all users should be able to assign a risk score to their direct contact and make prediction immediately at any time after convergence in a local way. Finally, the communication cost should be low by decreasing the size and the number of messages. The purpose of the protocol is to disseminate up-to-date information and maintain them without collecting them in the central place. The basic underlying idea is that all users are equivalent and run the same risk assessment model. Therefore, we exploit Newscast EM [79], a probabilistic gossip-based randomized communication clustering approach, originally developed for clustering users in peer-to-peer networks, for both two phases of our risk estimation approach.

In Newscast EM, each user initializes a local estimation of the parameters of the clustering model (mean, standard deviation, etc.). After that, the target user contacts a random user among all users to exchange their parameters estimation and aggregate them by weighted averaging. It is crucial to provide a uniform random subset of all users during the averaging process, to avoid any biased estimation towards high degree users [79]. Because, the probability of a user being sampled is proportional to his/her degree [95]. The degree is the number of direct friends for each user in the social graph. Therefore, we used gossip based peer-sampling service on DOSNs that provides a user with a uniform random subset of all users in the social network [76]. Based on this protocol, users exchange their parameters estimation and update them for some cycles to converge to a global correct estimation. Each user will maintain the list of last parameters estimations that he/she receives from other users in the fixed size cache.

The advantages of this distributed risk assessment model are include: 1. It uses the interactions of users that are distributed over the network, 2. the interaction of users are never send over the network in a raw form and 3. it has low computation and communication cost by gossiping parameters estimations of the clustering model instead of their raw data records.

For the following, first we will explain in summary our previous work that was a centralized two-phase risk assessment model in OSNs. Then, we summarized the decentralized group identification in DOSNs that is the first phase of our decentralized risk estimation approach. After that, we will cover all the details of the decentralized two-phase risk assessment model.

7.3 Feature Description

We used the GI features for the first phase and BF features for the second phase. We describe them in the following section.

7.3.1 GI Features

Group Identification user's features vector includes two types of features: individual and social features. Individual features are those, like age, gender, but also those that impact the possible users' behaviors, like, education and nationality. In addition to individual information, in order to measure users' attitude in online socialization, we consider the

following social features:
Number of Friends: social users with a lot of friends have different patterns than isolated users with few friends;
Activity Level: unlike active users that write a lot of posts, passive users do not send any information to others. We calculate this feature as the sum of: a) number of posts that a user sends to others from the first day of joining the community, b) number of likes that a user performs on posted items, and c) number of comments a user writes for posted items;
Percentage of public profile items: the assumption is that users with all profile information (100%) public are more social.

7.3.2 BF Features

The extraction of these features is based on the detection of risky behaviors by considering the behavioral patterns of well known attackers in DOSNs [85]. The first is the *user longevity*, which is measured as the number of days since the user joined the OSNs. The second one is the *item longevity*, which is measured as the number of days since an item has been uploaded in the OSNs.

Friendship Rate (FR). Since, attackers are more aggressive in establishing new friendship links [157], because isolated attackers cannot propagate content in OSNs [15]. Therefore, we consider the number of friends based on longevity of user, the number of days since the user joined the OSNs.

Mutual Friendship Rate (MFR). This feature computes the average number of mutual friends of a target user $\overline{u}$ with all his/her friends in the network.

Friend Mutual Friend Ratio (FMFR). In this feature, we are interested to catch those users with high number of friends, but a lower average number of mutual friends, as this could indicate a risky conduct.

Comment Rate (CR). In this feature, for each user, we measure the number of comments that the user written based on his/her longevity. For instance, in cyberbullying attacks, attackers send repeated hurtful messages to their victims [104], [106].

Started Comments (SC). In cyberbullying, most of the time attackers are starter in writing massages or comments for their victims [104], [106].

Like Propagation Speed (LPS). This feature measures the propagation speed of liked items that show an high propagation speed. For instance, in compromised account attacks, attackers, after compromising legitimate accounts, distribute malicious links in the network and manipulate a lot of users to put like on the posed items.

Comment Feedback Ratio (CFR). This feature is useful to detect those users that write a high number of comments and receive few likes on those comments. As an example, in cyberbulling and sybil attacks, attackers are more aggressive in sending messages [106].

Like Rate Like Propagation (LRLP). Since, Iin clickjacking, creeper, and socware, attackers try to propagate malicious items by forcing victims to unknowingly perform likes on these malicious items, this feature measures the number of likes on the items that have a high propagation speed.

7.4 Centralized Two Phase Risk Assessment Model

In this section, firstly we summarize our previous centralized two-phase risk assessment approach and then, we explain the distributed algorithm we need for our current decentralized two-phase risk assessment on DOSNs.

Risk assessment approach in centralized OSNs provides a general solution for both the service providers and users in OSNs to detect several types of attacks and risky users by assigning a risk score to all user's accounts in OSNs [85]. We considered both the activities and friendship patterns of users in OSNs to assign a risk score. We compared the behavioral patterns of users with other users in the network to find anomalous behaviors based on the divergency of the behavior from normal behavior. In modeling the normal behavior, the expectation is that similar users (e.g., similar in activity level, gender, education, country, and so on) tend to have similar behavioral models [7, 140]. Then, we proposed a *two-phase risk assessment*, by finding users with the same behavioral patterns in the first phase. Therefore, in the first phase, we proposed an approach to identify similar behavioral groups in the OSNs by applying clustering algorithm over a set of user's Group Identification (GI) features meaningful for people with the same behavioral patterns. After defining a set of groups including similar users, in the second phase we defined a set of normal behavioral models according to behavioral features for each group and we detect those users that deviate from other users with the same behavioral patterns. The clustering result of the second phase is allow us to assign a risk score to the target user by computing the degree of divergency of the user's behavior from other normal behavioral models.

For instance in the second phase, among a variety of activities in OSNs (i.e., sending friends request to others, sending posts or sharing items, writing comments or performing likes on users' post items, viewing profile information), we aim to monitor those activity that might reveal risky behaviors. As an example, simply having a high number of friends can not be considered risky behavior. However, having a high number of friends in a short period of time can be risky. Moreover, writing a lot of comments/posts without receiving any like on them or receiving a large number of likes in a short period of time can be considered a risky behavior and inform us that the target user might be a victim of an attack or try to propagate spam in the network. Therefore, we investigated the literature for well-known OSNs attacks to define a set of features, meaningful for risk assessment. These attacks includes: sybil, identity clone, compromised accounts, socware, creepers, cyberbullying and clickjacking attacks. Interested readers refer to [85] for more details.

Among all clustering techniques, we consider probabilistic-based clustering algorithm, that is a kind of soft clustering techniques and provide us the membership probability for each user to belong to the most fitting cluster. Because, hard clustering algorithms, (e.g., k-means) is not a perfect solution for clustering discrete or categorical data [20], since they are based on distance metrics. On the other hand, probabilistic-based clustering techniques try to find the most fitting cluster for each user by considering his/her features vector in data by defining K probability distributions for K clusters. Each probability distribution illustrates the likelihoods of those feature vectors that belong to a given cluster. Therefore, we apply EM (Expectation Maximization) algorithm that is a kind of probabilistic based clustering.

The EM algorithm iteratively learns and optimizes the parameters of the clustering model, that is, the mean and covariance matrix of each feature value for each cluster, and the fraction of users belonging to each cluster. Given the learned parameters, the algorithm assigns with each cluster a set of features values that most likely represent users belonging to that cluster. In the first iteration of the EM algorithm, EM assigns a set of K *membership probabilities* to each user u which means the parameters are initialized by random positive values. After that, EM iteratively learn and optimize the parameters of the clustering model in order to maximize these likelihoods and finally assign to each user the most fitting cluster with the highest probability.

To assign a risk score to each user, we compute the deviations of users' behavior from other normal behavioral models, by calculating the membership probabilities in the second phase of the clustering. If the target user is fitted very well to his/her normal behavioral models, the membership probability is high. In contrast, when the behavior of the user deviates from his/her behavioral models, the membership probability value is low. Based on this, the risk score associated with a target user u is defined as the inverse of the highest among u's membership probability values resulting by the second clustering phase.

Definition 12 (***User Risk Score (RS)***) *Let N be the set of users in the OSNs, and let $N_g \subset N$ be a set of users that belong to the same cluster g, resulted from the probabilistic-based clustering computed in the first phase over the GI features' values of user in N. Let $PB(N_g)$ be the probability-based clustering algorithm that takes as input the set of users in N_g and, based on their BFs, returns, for each user $u \in N_g$, the highest membership probability, denoted as $\mathcal{PCL}(u)$. Given a target user $u \in N_g$, the associated risk score $RS(u) \in [0,1]$ is defined as:*

$$RS(u) = 1 - \mathcal{PCL}(u) \tag{7.1}$$

$$AlternativeRiskScore : RS(u) = -log(\mathcal{PCL}(u)) \tag{7.2}$$

7.5 Distributed Two Phases Risk Assessment Model

In our distributed two-phase risk assessment approach, we used the same distributed clustering algorithm. In the initialization of the first phase, each user maintains the parameters estimation of the clustering model for all users in his/her RNC based on their GI features. After that, users communicate for exchanging these parameters estimation in order to converge to a global clustering model. Then, in the second phase, users communicate to exchange the parameters estimation of the clustering model according to their BF. In both the phases of our risk assessment model, we make use of the same newscast EM algorithm. Although, we have some issues in distributed risk assessment not present in the centralized one.

The first issue is that there are some kinds of attackers in the network that may be selected in the random selection phase to exchange the information with them. The goal of these attackers is to falsify their own parameters estimation of the clustering model and also try to manipulate the aggregation value during the exchanging process. For example, in Input Falsification attack, a compromised user may choose to lie about its own parameters estimation value since the raw data value of a given user is known only to that user. In Aggregation Manipulation attacks, a more powerful adversary may be able to compromise some other users that are also aggregators. Also, a malicious aggregator can report arbitrary subcomputation results and behave in arbitrary ways in order to attempt to evade detection [26].

We need to mention that there are several approaches to avoid these problems in the literature. For instance, various algorithms for outlier detection [1, 21, 134, 143, 145] can detect abnormal changes based on statistical analysis. Also, some approaches introduce One-hop Verification [66, 68] and Witness-based Verification [43, 37] to prevent the malicious user from presenting falsified aggregation results by simply ensuring the aggregation computation of each user is checked by some other users in the network as verifiers. Moreover, there are some reputation based approaches like QoI metric-based scheme [55] where users evaluate trustworthiness of other users based on their past behavior, or based on the feedback of other users [138] and determine whether to accept a received aggregate based on the sender's trustworthiness level. On the other hand, there are some secure aggregation protocols that try to produce accurate aggregation result and be resilient against general user compromised attacks. These protocols avoid intermediate users to report arbitrary values to bias the final aggregation by designing secure aggregation and prevent intermediate users from illegal modification. For instance, SIA [122], using cryptographic hash values and SecureDAV [99], is a system for encrypting and authenticating an aggregation result based on the signature mechanism and private keys where the verification of the central global result is fully distributed. This technique will be further extended by subsequent schemes, most notably SHIA [27]. In this paper, we are not taken into account these kind of attackers and we rely on these approaches and assume that the information of users is reliable and attackers are not able to manipulate the aggregation results.

The second issue is the lack of a data center that avoid our protocol to keep the result of the first grouping phase in one data center accessible for all users and then, we can not access the behavioural data of all users in each group in order to apply the processing of the second phase clustering, based on their behavioural features. On the other hand, in the centralized version, the provider of OSNs can apply these two phase clustering and after that assign a risk score to all users in the whole network. But, in a decentralized version every target user need to be able dynamically assign a risk score to other users at any time after convergence. Therefore, in our approach each user will act as a service provider and maintains the result of all clustering models associated with all groups in the network in his/her cache in order to be able to make a decision for other users and finally assign them a risk score. In the following we explain in more detail the first and second phase of our risk assessment approach in a distributed manner.

7.5.1 Modelling Behavioral Patterns: User Behaviour-based Group Identification Phase

In the first phase, our goal is to have similar users in each group based on their GI features includes: social and individual features. Feature vectors of each user are given as input to the Algorithm 1 as we explained more precisely in Section 6.3 of chapter 6. Each user with his/her features vector in the initialization phase, estimates the parameters of the first clustering model and start to exchange this information with a set of randomly selected users in his/her RNC cache. After some cycles, all users access to a set of random users in his/her RNC cache and converge to a global clustering model with the same parameters estimation and they have a membership probability to belong to their most fitting cluster. Therefore, after convergence, each user can assign a behavior group (cluster number) to him/herself or his/her friends based on the GI feature vector by considering the maximum membership probability among them.

7.5.2 Modeling Normal Behavior: Risk Assessment Phase

In the second phase, our goal is to create some behavioral models for similar users according to the behavioral features and detect risky users that deviate from these behavioral models.

In this phase all users start to exchange their parameters estimation of the second phase clustering model based on the behavioral features. The goal of this two-phase risk assessment approach is considering each user in the DOSNs as a service provider similar to a centralized OSNs. Because, in DOSNs, the goal is providing each user with the ability to assign a group number and risk score to other users. Toward this goal, each user need to have all clustering models associated with all groups in the network to be able to assign a risk score to other users based on which group they belong. The detail of the algorithm is illustrated in Algorithm 2, and summarize as the following.

After the convergence of the first phase of clustering, based on GI features of users, all users have the assigned group number and able to assign a group number to their direct friends as well. Then, in order to have all of the clustering models of all groups of users in the second phase, each user will do the following step.

First, a target user u_i initializes the local parameters estimation for each cluster $L_G, G = 1, \ldots, l$ in the second phase based on his/her BF associated to the group number he/she belong, where l is the number of groups in the first phase of clustering. Then, user u_i creates a cache called, All Groups Estimations (AGE) cache, and maintains the information includes the user ID, the user group number, the path to reach the user ID, and the parameters estimation of each cluster L_G associated to user's group number. After that, user u_i fills only the parameters estimation of his/her group in the cache and set the other entrie's parameters estimation to zero and their ID, their group number and the path empty.

Next, each user in every cycle, i.e., user u_i, contacts a user from his/her RNC with the largest longevity, i.e., user u_j to send the updated ExC, his/her group number and his/her AGE to the next user in the path towards user u_j, i.e., u_k.

When user u_j receives from one of his/her direct friends the ExC and AGE coming from user u_i, user u_j replies as follows.

First, user u_j sends back his/her own AGE to the next user toward user u_i. Next, user u_j selects a random subset of S neighbors of his/her own RNC and save them in his/her ExC and update the entries path inside. After that, user u_j will send back his/her own ExC to the next user toward user u_i. Then, user u_j updates the entries path in the received ExC and updates his/her RNC *longevity* attribute of all entries as mentioned in algorithm 1.

Finally, if user u_j belongs to the same group with user u_i, user u_j updates his/her own parameters estimation for each cluster L_G by calculating the weighted averaging and update them in his/her AGE. More over, user u_j updates the remain entries of the received AGE from user u_i includes the users ID, parameters estimation of all other groups, the entries's group number and the path to reach those entries in his/her AGE cache.

But, if the user u_j belongs to another group, user u_j keeps his/her own parameters estimation associated to his/her group in AGE cache. After that, user u_j updates the remain entries of the received AGE from user u_i includes the users ID, parameters estimation of all other groups, the entries's group number and the path to reach those entries in his/her AGE cache.

Therefore, after some exchanging cycles, all users have the same AGE including all the parameters estimation for each cluster associated to each group. In this stage all users are able to check the stopping tolerance of their group and after some EM iteration, each user have a list of last updated entries in AGE cache, and able to compute the membership probability for him/herself to belong to the most fitting cluster according to the BF features. Then, each user able to assign a group number to his/her direct friends or other users based on the value of the GI features and then, assign a risk score by computing the membership probability of that user to belong to the most fitting cluster according to their BF.

7.6 Experiments

In the following we explain the data set in Section 7.6.1. Then, we show the result for the convergence of the clustering model in the second phase and the performance of the clustering model in Section 7.6.2.

7.6.1 Facebook Dataset

We evaluate the algorithms on real Facebook data crawled and used in [3]. We considered the largest connected component by removing those user profiles with so many missing values in their BF and profile features which totally includes 5,500 user profiles, with a total of about 443,011 friend links, 50,515 likes and 124,920 comments.

Algorithm 2 Gossip based Two-Phase Risk Assessment

```
Initialization()
Loop:
if $u = u_i$ then //if $u_i$ has to push information
    $ExC \leftarrow RNC.SelectRandomEntries()$
    $u_j \leftarrow RNC.SelectRandomUser()$
    Sends $ExC \leftarrow UpdateExC(ExC, u_i)$ to $u_i$ 's neighbor toward $u_j$
    Sends $G(u_i)$ to $u_i$ 's neighbor toward $u_j$
    Sends $AGE_i$ to $u_i$ 's neighbor toward $u_j$
    $AGE_j \leftarrow Receive(u_j)$
    for all entries in $AGE_i$ do
        Update $\Theta_e$
else if $u = u_j$ then //If $u_j$ has to reply $u_i$
    Sends $G(u_j)$ to $u_j$ 's neighbor toward $u_i$
    Sends $AGE_j$ to $u_j$ 's neighbor toward $u_i$
    $ExC \leftarrow RNC.SelectRandomEntries()$
    Sends $ExC \leftarrow UpdateExC(ExC, u_j)$ to $u_j$ 's neighbor toward $u_i$
    Receives $AGE_i$ from $u_j$ 's neighbor
    Receives $ExC$ from $u_j$ 's neighbor
    $ExC \leftarrow UpdateExC(ExC, u_j)$
    if G($u_j$)= G($u_i$) then
        $\Theta_j \leftarrow UpdateModel(\Theta_i, \Theta_j)$
        Update $\Theta_j$ in $AGE_j$
        for all entries $e$ in $AGE_j$ do
            Update $\Theta_e$
    else//G($u_j$) $\neq$ G($u_i$)
        for all entries $e$ in $AGE_j$ do
            Update $\Theta_e$
else//($u = u_k$ within the path to $u_j$)
    Sends received $AGE_i$ and $ExC$ to the next user toward $u_j$
End Loop
procedure Initialization()
    InitModel()
    InitAGE()
    return $AGE$, $RNC$
End procedure
```

7.6.2 Modeling Normal Behavior: Risk Assessment Phase

The goal of the second phase is detecting risky users that deviate from the behavioral models of similar users according to their BF. In this phase users exchange the parameters estimation of the second clustering model based on their BF and update their own parameters estimation according to which group they belong. Then, each user in order to have the parameters estimation of all users associated to all groups, exchange his/her AGE with other users. The Figure 7.2, shows how these parameters estimation in AGE cache converge to a global AGE cache available for all users after convergence. In the first cycles

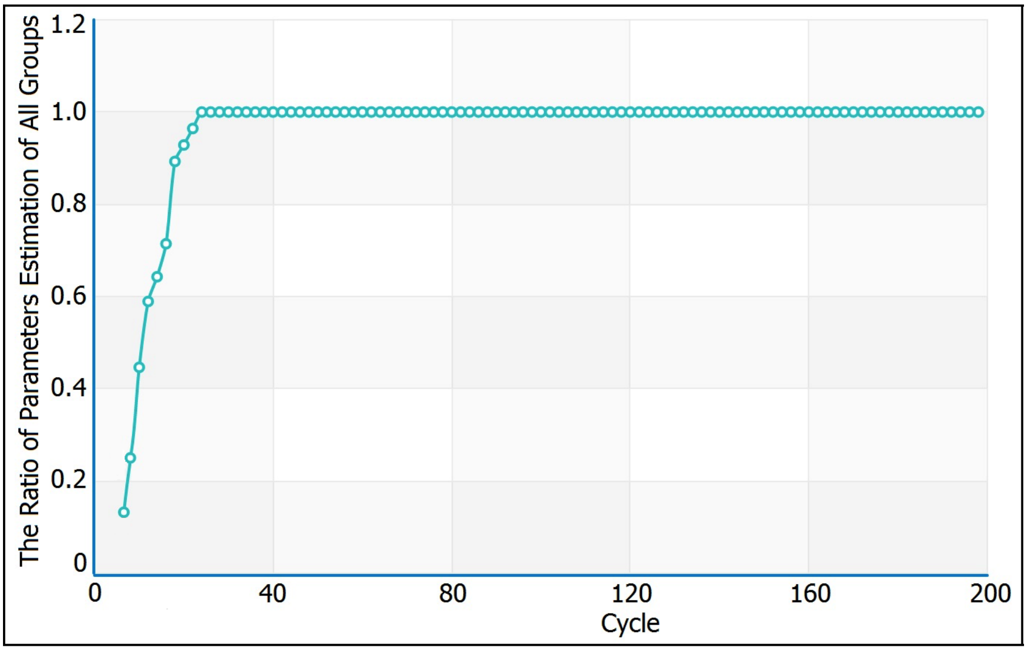

Figure 7.2: The Ratio of Parameters Estimation of All Groups

each users has only the parameters estimation of his/her group. But, after few cycles all users have the parameters estimation of all groups in their AGE cache (the values are averages for 20 independent runs). Although, we can see that after a few cycles (20 Cycles), all users have all parameters estimation associated to all groups in their AGE cache, but the value of their parameters estimation according to each feature not converge to a global value and they should exchange their parameters estimation for more cycles. In the next experiments, we show how their parameters estimation converge to a global value.

Therefore, we randomly select 10 users from each group, e.i, totally 70 users, and we monitor the value of parameters estimation of one of the groups associated to one BF feature. The Figure 7.3 illustrates how all users from different groups in the network will converge to the global value for their parameters estimation of group three (the ratio of users belonging to this group is around 32 Percent), associated to the feature ”Number of Friends” (after 120 cycles). We need to mention that users do averaging in their parameters estimation only with those users that are belong to the same group as we explain in more detail in Algorithm 2. In this Figure 7.3, we can see that even those users from other groups rather than group three that do not have the parameters estimation of group three in the first cycles (the value of the parameters estimation of the feature in group three is zero for them in the first cycle) will converge to the global value that is available for every user, no matter they belong to which group.

Moreover, Figure 7.3 demonstrates the parameters estimation of users in group 7. The ratio of users belonging to this cluster is around 5 percent. As we can see in the Figure 7.4, the value of parameters estimation of users in this group, converge to a global value

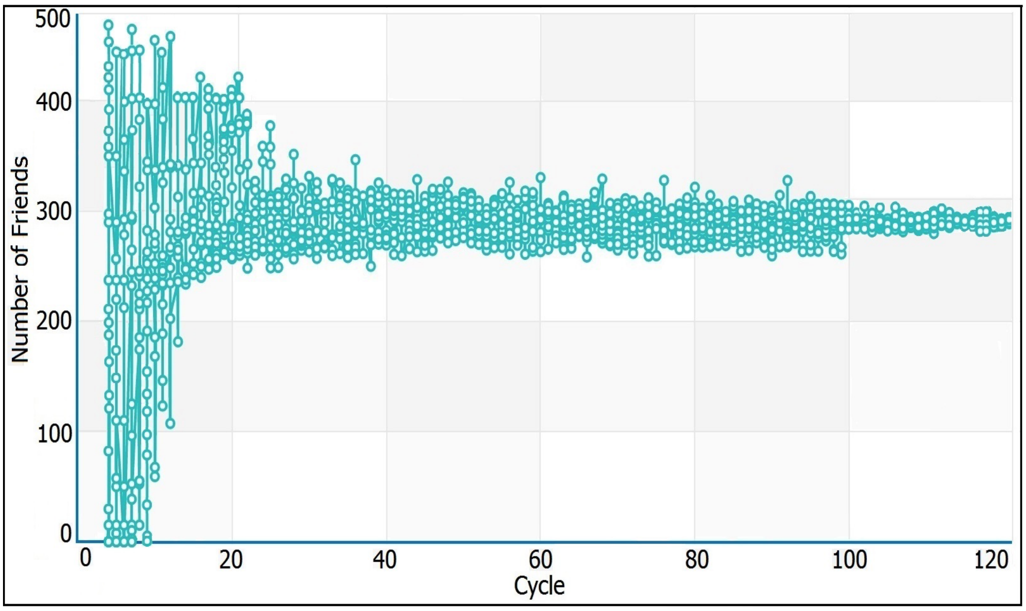

Figure 7.3: Parameters Estimation of Group Three for The Feature ”Number of Friends”

that is not exactly equal to the correct mean. Because, the number of users belonging to this group is low. But, this does not have a big consequence on the result since their parameters estimation is far from those users from other groups.

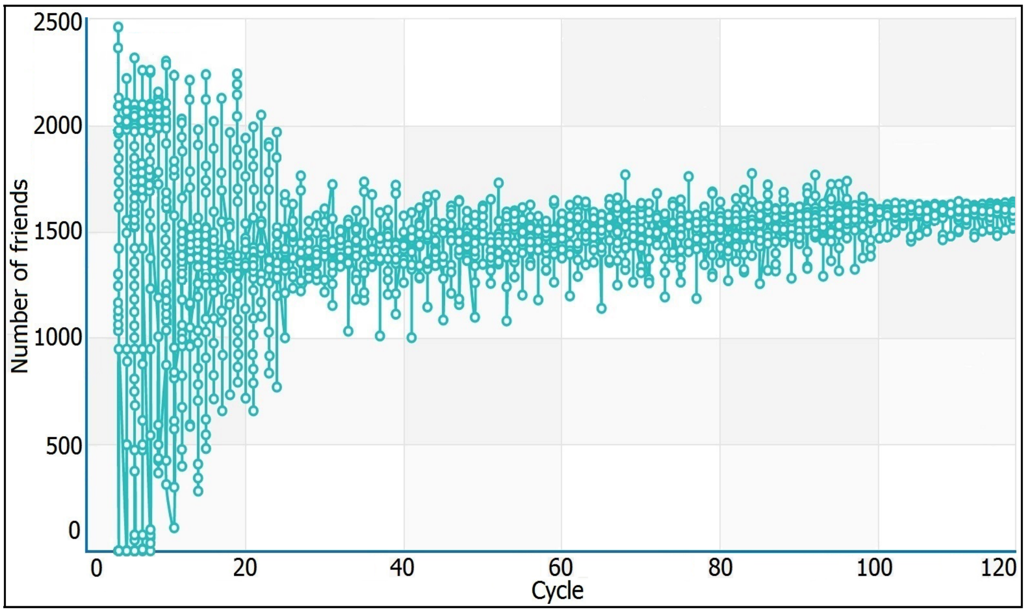

Figure 7.4: Parameters Estimation of Group Seven for The Feature ”Number of Friends”

Conclusion

Identifying misbehaviors is an important challenge for monitoring and analyzing user behaviors in OSNs/DOSNs to detect risky users including malicious attackers. Therefore, in this thesis, we analyze the user's activities in the centralized and decentralized online social network. The basic idea is the more the user behavior diverges from 'normal behavior', more the user is risky. We proposed several approaches by considering the structural patterns of users, activity patterns of users and also by monitoring the behavioral patterns of users over time-evolving graphs to detect anomalous changes. In Chapter 3, we proposed a two-phase risk assessment approach to assign a risk score to each OSNs user based on the activity patterns of users by computing the divergency of user behavior from 'normal behavior', using a clustering algorithm. Experiments carried out on a real Facebook dataset showed the effectiveness of our proposal.

After that, in Chapter 4, we proposed a local risk estimation measure (Local Risk Factor) for direct contacts of a target user. Our risk estimation is based on anomaly detection algorithm having as key idea the fact the malicious users in OSNs show some common features on the structure of their social graphs that is different from those of legitimate users. We demonstrate that LRF can be used to define the risk of direct contacts efficiently in large scale OSNs. We also show that some of the features are more influential in risk assessment in OSNs than others. Since, graph based approaches for risk assessment in dynamic graph is challenging task, In the following of this chapter, we propose a solution to have a high performance risk model that is more robust in online social network by changing the behavior of attackers. Therefore, in Chapter 5, we detected users with anomalous changes in the structure of their subgraphs over time. We analyzed and monitored the change patterns of users over time and compare them with change patterns of other similar users and previous change patterns of the same user. After that, in Chapter 6, we identify behavioral groups of users that share the same behavioral patterns in DOSNs for risk assessment purposes. We applied distributed clustering algorithm, Newscast EM, to identify group of behavioral patterns on top of DOSNs. We combine the ability to access random users on social networks with distributed clustering model to identify group of behavioral patterns. Our main goal is using this identified behavioral patterns of users for identifying misbehaviors in DOSNs. Identifying misbehaviors is an important challenge for monitoring, analyzing the social network for anomaly detection purposes, when we don't have data in a central site for analysis and the information of users are distributed over the network and this information can not be shared or moved to any central server or other users due to privacy issues. Therefore, in the last chapter, Chapter 7, we propose a distributed two-phase risk assessment approach in DOSNs to detect the divergency of users behavior from normal behavior. Therefore, we used a fully distributed clustering algorithm to compute the probability of behavior of each user with respect to other users in the same community in each cluster to predict anomalous behaviors in the network. Our existing anomaly detection algorithm does not need any global knowledge of the system. In all proposed solutions, we used real social network datasets, like Facebook, Orkut and Google+ to show the performance of our solutions.

We plan to extend our previous works along a set of directions. First, we plan to extend the work in Chapter 3 so as to make it able to perform a continuous monitoring and estimation of risk scores. Then, we plan to extend the Chapter 4 by considering more subsets of structural features and add other user features to define risk score and develop more accurate models for risk assessment in OSNs. After that, we plan to extend the work in Chapter 5 by investigating the local neighborhood structure of users with anomalous changes in more details. Because, these changes can also be because of benign reasons and then, test our measures on different datasets. Next, we are planning to extend the work in Chapter 6 by proposing a solution for defining the best number of clusters in a distributed manner and also for exchanging information in a secure way for avoiding malicious users manipulating the local parameters estimation of the clustering model during the exchanging process. Finally, our future plan in Chapter 7, is proposing a solution to have a more accurate risk assessment model in OSNs/DOSNs by proposing new features. Also, we plan to have a dynamic high performance risk assessment model that will be robust in changing the behavior of attackers over time. After that, we plan to apply the risk model for other decentralized environment.

Bibliography

[1] Daniel J Abadi, Samuel Madden, and Wolfgang Lindner. Reed: Robust, efficient filtering and event detection in sensor networks. In *Proceedings of the 31st international conference on Very large data bases*, pages 769–780. VLDB Endowment, 2005.

[2] James Abello, Mauricio GC Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *LATIN 2002: Theoretical Informatics*, pages 598–612. Springer, 2002.

[3] Cuneyt Gurcan Akcora, Barbara Carminati, and Elena Ferrari. Privacy in social networks how risky is your social graph? In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 9–19. IEEE, 2012.

[4] Leman Akoglu and Christos Faloutsos. Event detection in time series of mobile communication graphs. In *Army Science Conference*, pages 77–79, 2010.

[5] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball- spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.

[6] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[7] Christa SC Asterhan and Tammy Eisenmann. Online and face-to-face discussions in the classroom: A study on the experiences of'active'and'silent'students. In *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*, pages 132–136. International Society of the Learning Sciences, 2009.

[8] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42. ACM, 2012.

[9] Badi H Baltagi, Chihwa Kao, Long Liu, et al. Estimation and identification of change points in panel models with nonstationary or stationary regressors and error terms. *Preprint*, 2012.

[10] Sanghamitra Bandyopadhyay, Chris Giannella, Ujjwal Maulik, Hillol Kargupta, Kun Liu, and Souptik Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952–1985, 2006.

[11] Luca Becchetti, Carlos Castillo, Debora Donato, Stefano Leonardi, and Ricardo A Baeza-Yates. Link-based characterization and detection of web spam. In *AIRWeb*, pages 1–8, 2006.

[12] Kanishka Bhaduri and Ashok N Srivastava. A local scalable distributed expectation maximization algorithm for large peer-to-peer networks. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 31–40. IEEE, 2009.

[13] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM, 2009.

[14] Yazan Boshmaf, Konstantin Beznosov, and Matei Ripeanu. Graph-based sybil detection in social and information systems. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 466–473. IEEE, 2013.

[15] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, and Konstantin Beznosov. Integro: Leveraging victim prediction for robust fake account detection in osns. In *NDSS*, volume 15, pages 8–11. Citeseer, 2015.

[16] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network- when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011.

[17] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. Design and analysis of a social botnet. *Computer Networks*, 57(2):556–578, 2013.

[18] Danah Boyd and Nicole Ellison. Social network sites: definition, history, and scholarship. *IEEE Engineering Management Review*, 3(38):16–31, 2010.

[19] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653–1664. IEEE, 2005.

[20] Paul S Bradley, Usama Fayyad, and Cory Reina. Scaling em (expectation-maximization) clustering to large databases. Technical report, Technical Report MSR-TR-98-35, Microsoft Research Redmond, 1998.

[21] Joel W Branch, Chris Giannella, Boleslaw Szymanski, Ran Wolff, and Hillol Kargupta. In-network outlier detection in wireless sensor networks. *Knowledge and information systems*, 34(1):23–54, 2013.

[22] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[23] Horst Bunke, Peter Dickinson, Andreas Humm, Ch Irniger, and Miro Kraetzl. Computer network monitoring and abnormal event detection using graph matching and multidimensional scaling. In *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 576–590. Springer, 2006.

[24] Zhuhua Cai and Christopher Jermaine. The latent community model for detecting sybil attacks in social networks. NDSS, 2012.

[25] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, pages 197–210, 2012.

[26] Haowen Chan, Hsu-Chun Hsiao, Adrian Perrig, and Dawn Song. *Secure distributed data aggregation*. Now Publishers Inc, 2011.

[27] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 278–287. ACM, 2006.

[28] Hao Chen, Nancy Zhang, et al. Graph-based change-point detection. *The Annals of Statistics*, 43(1):139–176, 2015.

[29] Zhixiang Chen, Ada Wai-Chee Fu, and Jian Tang. On complementarity of cluster and outlier detection schemes. In *Data Warehousing and Knowledge Discovery*, pages 234–243. Springer, 2003.

[30] Shihabur Rahman Chowdhury, Arup Raton Roy, Maheen Shaikh, and Khuzaima Daudjee. A taxonomy of decentralized online social networks. *Peer-to-Peer Networking and Applications*, 8(3):367–383, 2015.

[31] Hanbo Dai, Feida Zhu, Ee-Peng Lim, and HweeHwa Pang. Detecting anomalies in bipartite graphs with mutual dependency principles. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 171–180. IEEE, 2012.

[32] George Danezis and Prateek Mittal. Sybilinfer- detecting sybil nodes using social networks. In *NDSS*, 2009.

[33] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.

[34] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *Internet Computing, IEEE*, 10(4):18–26, 2006.

[35] Vacha Dave, Saikat Guha, and Yin Zhang. Measuring and fingerprinting click-spam in ad networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 175–186. ACM, 2012.

[36] Vacha Dave, Saikat Guha, and Yin Zhang. Viceroi catching click-spam in search ad networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, pages 765–776. ACM, 2013.

[37] Emiliano De Cristofaro, Jens-Matthias Bohli, and Dirk Westhoff. Fair: fuzzy-based aggregation providing in-network resilience for real-time wireless sensor networks. In *Proceedings of the second ACM conference on Wireless network security*, pages 253–260. ACM, 2009.

[38] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Generalized louvain method for community detection in large networks. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 88–93. IEEE, 2011.

[39] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[40] Himel Dev, Mohammed Eunus Ali, and Tanzima Hashem. User interaction based community detection in online social networks. In *Database Systems for Advanced Applications*, pages 296–310. Springer, 2014.

[41] Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894. ACM, 2012.

[42] John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[43] Wenliang Du, Jing Deng, Yunghsiang S Han, and Pramod K Varshney. A witness-based approach for data fusion assurance in wireless sensor networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 3, pages 1435–1439. IEEE, 2003.

[44] Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18. ACM, 2009.

[45] Maurizio Dusi, Christian Vitale, Saverio Niccolini, and Christian Callegari. Distributed pca-based anomaly detection in telephone networks through legitimate-user profiling. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1107–1112. IEEE, 2012.

[46] William Eberle and Lawrence Holder. Compression versus frequency for mining patterns and anomalies in graphs. In *Ninth workshop on mining and learning with graphs (MLG 2011), SIGKDD, at the 17th ACM SIGKDD conference on knowledge discovery and data mining (KDD 2011)*, 2011.

[47] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Compa- detecting compromised accounts on social networks. In *NDSS*, 2013.

[48] Tina Eliassi-Rad, Keith Henderson, Sprios Papadimitriou, and Christos Faloutsos. A hybrid community discovery framework for complex networks. In *SIAM Conference on Data Mining*, 2010.

[49] Aviad Elishar, Michael Fire, Dima Kagan, and Yuval Elovici. Organizational intrusion- organization mining using socialbots. In *Social Informatics (SocialInformatics), 2012 International Conference on*, pages 7–12. IEEE, 2012.

[50] Mohammad Reza Faghani and Hossein Saidi. Malware propagation in online social networks. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 8–14. IEEE, 2009.

[51] Michael Fire, Roy Goldschmidt, and Yuval Elovici. Online social networks- threats and solutions. 2013.

[52] Joshua Fogel and Elham Nehmad. Internet social network communities: Risk taking, trust, and privacy concerns. *Computers in human behavior*, 25(1):153–160, 2009.

[53] Carlos Freitas, Fabricio Benevenuto, Saptarshi Ghosh, and Adriano Veloso. Reverse engineering socialbot infiltration strategies in twitter. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 25–32. ACM, 2015.

[54] Carlos A Freitas, Fabrcio Benevenuto, Saptarshi Ghosh, and Adriano Veloso. Reverse engineering socialbot infiltration strategies in twitter. *arXiv preprint arXiv:1405.4927*, 2014.

[55] Saurabh Ganeriwal, Laura K Balzano, and Mani B Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3):15, 2008.

[56] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. Towards online spam filtering in social networks. In *NDSS*, 2012.

[57] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 35–47. ACM, 2010.

[58] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[59] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Dawn Song, et al. Jointly predicting links and inferring attributes using a social-attribute network (san). *arXiv preprint arXiv:1112.3265*, 2011.

[60] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song. Evolution of social-attribute networks: measurements, modeling, and implications using google+. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 131–144. ACM, 2012.

[61] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. Spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37. ACM, 2010.

[62] Dongbing Gu. Distributed em algorithm for gaussian mixtures in sensor networks. *Neural Networks, IEEE Transactions on*, 19(7):1154–1166, 2008.

[63] Keith Henderson, Tina Eliassi-Rad, Christos Faloutsos, Leman Akoglu, Lei Li, Koji Maruhashi, B Aditya Prakash, and Hanghang Tong. Metric forensics: a multi-level approach for mining volatile graphs. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172. ACM, 2010.

[64] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 663–671. ACM, 2011.

[65] Shohei Hido, Seiya Tokui, and Satoshi Oda. Jubatus: An open source platform for distributed online machine learning. In *NIPS 2013 Workshop on Big Learning, Lake Tahoe.*

[66] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pages 384–391. IEEE, 2003.

[67] Ting-Kai Huang, Md Sazzadur Rahman, Harsha V Madhyastha, Michalis Faloutsos, and Bruno Ribeiro. An analysis of socware cascades in online social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 619–630. International World Wide Web Conferences Steering Committee, 2013.

[68] Pawan Jadia and Anish Mathuria. Efficient secure aggregation in sensor networks. In *High Performance Computing-HiPC 2004*, pages 40–49. Springer, 2004.

[69] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[70] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8, 2007.

[71] Lei Jin, Xuelian Long, Hassan Takabi, and James BD Joshi. Sybil attacks vs identity clone attacks in online social networks. *ISA*, 2012.

[72] Lei Jin, Hassan Takabi, and James BD Joshi. Analysing security and privacy issues of using e-mail address as identity. *International Journal of Information Privacy, Security and Integrity*, 1(1):34–58, 2011.

[73] Lei Jin, Hassan Takabi, and James BD Joshi. Towards active detection of identity clone attacks on online social networks. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 27–38. ACM, 2011.

[74] Wen Jin, Anthony KH Tung, Jiawei Han, and Wei Wang. Ranking outliers using symmetric neighborhood relationship. In *Advances in Knowledge Discovery and Data Mining*, pages 577–593. Springer, 2006.

[75] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. Change-point detection in time-series data based on subspace identification. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 559–564. IEEE, 2007.

[76] Mansour Khelghatdoust and Sarunas Girdzijauskas. Short: Gossip-based sampling in social overlays. In *Networked Systems*, pages 335–340. Springer, 2014.

[77] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. D elta c on: A principled massive-graph similarity function. SIAM, 2013.

[78] Wojtek Kowalczyk, Márk Jelasity, and A Eiben. Towards data mining in large and fully distributed peer-to-peer overlay networks. In *Proceedings of BNAIC'03*, pages 203–210, 2003.

[79] Wojtek Kowalczyk and Nikos A Vlassis. Newscast em. In *Advances in neural information processing systems*, pages 713–720, 2004.

[80] Maciej Kurant, Minas Gjoka, Carter T Butts, and Athina Markopoulou. Walking on a graph with a magnifying glass: stratified sampling via weighted random walks. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 281–292. ACM, 2011.

[81] Maciej Kurant, Athina Markopoulou, and Patrick Thiran. Towards unbiased bfs sampling. *Selected Areas in Communications, IEEE Journal on*, 29(9):1799–1809, 2011.

[82] Lifeng Lai. Quickest change point identification across a sensor array. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–5. IEEE, 2012.

[83] Naeimeh Laleh and Mohammad Abdollahi Azgomi. A hybrid fraud scoring and spike detection technique in streaming data. *Intelligent Data Analysis*, 14(6):773–800, 2010.

[84] Naeimeh Laleh, Barbara Carminati, and Elena Ferrari. Graph based local risk estimation in large scale online social networks. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 528–535. IEEE, 2015.

[85] Naeimeh Laleh, Barbara Carminati, and Elena Ferrari. Risk assessment in online social networks based on anomalous behavior detection. *Dependable and Secure Computing, IEEE Transactions on*, PP(99), 2016.

[86] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers- social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM, 2010.

[87] Amanda Lenhart, Kristen Purcell, Aaron Smith, and Kathryn Zickuhr. Social media & mobile internet use among teens and young adults. millennials. *Pew Internet & American Life Project*, 2010.

[88] Chris Lesniewski-Lass and M Frans Kaashoek. Whanau: A sybil-proof distributed hash table. NSDI, 2010.

[89] Jason Lewis. How spies used facebook to steal nato chief's details. *The Telegraph*, 10, 2012.

[90] Zhong Li, Cheng Wang, Siqian Yang, Changjun Jiang, and Xiangyang Li. Lass: Local-activity and social-similarity based data forwarding in mobile social networks. *Parallel and Distributed Systems, IEEE Transactions on*, 26(1):174–184, 2015.

[91] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[92] Yao Liu, Qiao Liu, and Zhiguang Qin. Community detecting and feature analysis in real directed weighted social networks. *Journal of Networks*, 8(6):1432–1439, 2013.

[93] Stefano Lodi, Gianluca Moro, and Claudio Sartori. Distributed data clustering in multi-dimensional peer-to-peer networks. In *Proceedings of the Twenty-First Australasian Conference on Database Technologies-Volume 104*, pages 171–178. Australian Computer Society, Inc., 2010.

[94] Dongyuan Lu, Qiudan Li, and Stephen Shaoyi Liao. A graph-based action network framework to identify prestigious members through member's prestige evolution. *Decision Support Systems*, 53(1):44–54, 2012.

[95] Jianguo Lu and Dingding Li. Sampling online social networks by random walk. In *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 33–40. ACM, 2012.

[96] Rich Lundeen, Jesse Ou, and Travis Rhodes. New ways i'm going to hack your web app. *Blackhat AD*, 2011.

[97] Stephen Lyng. Sociology at the edge: Social theory and voluntary risk taking. *Edgework: The sociology of risk-taking*, pages 17–49, 2005.

[98] Jamie MacLennan, ZhaoHui Tang, and Bogdan Crivat. *Data mining with Microsoft SQL server 2008*. Wiley. com, 2011.

[99] Ajay Mahimkar and Theodore S Rappaport. Securedav: A secure data aggregation and verification protocol for sensor networks. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 4, pages 2175–2179. IEEE, 2004.

[100] Marcelo Maia, Jussara Almeida, and Virgílio Almeida. Identifying user behavior in online social networks. In *Proceedings of the 1st workshop on Social network systems*, pages 1–6. ACM, 2008.

[101] Marcus A Maloof. *Machine learning and data mining for computer security*. Springer New York, 2006.

[102] Ian McCulloh. *Detecting changes in a dynamic social network*. ProQuest, 2009.

[103] Ian McCulloh, Matthew Webb, John Graham, Kathleen Carley, and Daniel B Horn. Change detection in social networks. Technical report, DTIC Document, 2008.

[104] Phil McKenna. The rise of cyberbullying. *New Scientist*, 195(2613):26–27, 2007.

[105] R McMillan. Researchers make wormy twitter attack. *PCWorld, March*, 2009.

[106] Ersilia Menesini and Annalaura Nocentini. Cyberbullying definition and measurement. *Zeitschrift fr Psychologie/Journal of Psychology*, 217(4):230–232, 2009.

[107] Thomas Mensink, Wojciech Zajdel, and Ben Krose. Distributed em learning for appearance based multi-camera tracking. In *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on*, pages 178–185. IEEE, 2007.

[108] Silvia Mitter, Claudia Wagner, and Markus Strohmaier. A categorization scheme for socialbot attacks in online social networks. *arXiv preprint arXiv:1402.6288*, 2014.

[109] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 383–389, 2010.

[110] Todd K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[111] Seyed Ahmad Moosavi and Mehrdad Jalali. Community detection in online social networks using actions of users. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pages 1–7. IEEE, 2014.

[112] Sourav Mukherjee and Hillol Kargupta. Distributed probabilistic inferencing in sensor networks using variational approximation. *Journal of Parallel and Distributed Computing*, 68(1):78–92, 2008.

[113] Samantha Murphy. Teens ditch e-mail for texting and facebook. *MSNBC. com, Aug*, 2010.

[114] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[115] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[116] Nam P Nguyen, Thang N Dinh, Sindhura Tokala, and My T Thai. Overlapping communities in dynamic networks: their detection and mobile applications. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2011.

[117] Afshin Nikseresht and Marc Gelgon. Gossip-based computation of a gaussian mixture model for distributed multimedia indexing. *Multimedia, IEEE Transactions on*, 10(3):385–392, 2008.

[118] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[119] William V Pelfrey Jr and Nicole Weber. Talking smack and the telephone game: conceptualizing cyberbullying with middle and high school youth. *Journal of Youth Studies*, 17(3):397–414, 2014.

[120] N Perlroth. Fake twitter followers become multimillion-dollar business. *The New York Times*, 2013.

[121] Brandon Pincombe. Anomaly detection in time series of graphs using arma processes. *ASOR BULLETIN*, 24(4):2, 2005.

[122] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM, 2003.

[123] Daniele Quercia and Stephen Hailes. Sybil attacks against mobile users: friends and foes to the rescue. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[124] Fatemeh Rahimian, Amir H Payberah, Sarunas Girdzijauskas, Mark Jelasity, and Seif Haridi. Ja-be-ja: A distributed algorithm for balanced graph partitioning. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, pages 51–60. IEEE, 2013.

[125] Md Sazzadur Rahman, Ting-Kai Huang, Harsha V Madhyastha, and Michalis Faloutsos. Efficient and scalable socware detection in online social networks. In *USENIX Security Symposium*, pages 663–678, 2012.

[126] Md Sazzadur Rahman, Ting-Kai Huang, Harsha V. Madhyastha, and Michalis Faloutsos. Frappe: Detecting malicious facebook applications. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 313–324, New York, NY, USA, 2012. ACM.

[127] Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge, 2012.

[128] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. Hyperspherical cluster based distributed anomaly detection in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1833–1847, 2014.

[129] Marisa Reddy, Randy Borum, John Berglund, Bryan Vossekuil, Robert Fein, and William Modzeleski. Evaluating risk for targeted violence in schools: Comparing risk assessment, threat assessment, and other approaches. *Psychology in the Schools*, 38(2):157–172, 2001.

[130] RIVA Richmond. Stolen facebook accounts for sale. *The New York Times*, 2, 2010.

[131] Dzulkifli S Scherber and Haralabos C Papadopoulos. Distributed computation of averages over ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):776–787, 2005.

[132] Clifford W Scherer and Hichang Cho. A social network contagion theory of risk perception. *Risk analysis*, 23(2):261–267, 2003.

[133] Abhishek B Sharma, Haifeng Chen, Min Ding, Kenji Yoshihira, and Guofei Jiang. Fault detection and localization in distributed systems using invariant relationships. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–8. IEEE, 2013.

[134] Bo Sheng, Qun Li, Weizhen Mao, and Wen Jin. Outlier detection in sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 219–228. ACM, 2007.

[135] Lu Shi, Shucheng Yu, Wenjing Lou, and Y Thomas Hou. Sybilshield: An agent-aided social network-based sybil defense among multiple communities. In *INFOCOM, 2013 Proceedings IEEE*, pages 1034–1042. IEEE, 2013.

[136] Peter Shoubridge, Miro Kraetzl, WAL WALLIS, and Horst Bunke. Detection of abnormal change in a time series of graphs. *Journal of Interconnection Networks*, 3(01n02):85–101, 2002.

[137] Kumar Sricharan and Kamalika Das. Localizing anomalous changes in time-evolving graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1347–1358. ACM, 2014.

[138] Mudhakar Srivatsa, Li Xiong, and Ling Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on World Wide Web*, pages 422–431. ACM, 2005.

[139] Tao Stein, Erdong Chen, and Karan Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, page 8. ACM, 2011.

[140] Abigail J Stewart and Christa McDermott. Gender in psychology. *Annu. Rev. Psychol.*, 55:519–544, 2004.

[141] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

[142] Gianluca Stringhini, Gang Wang, Manuel Egele, Christopher Kruegel, Giovanni Vigna, Haitao Zheng, and Ben Y Zhao. Follow the green: growth and dynamics in twitter follower markets. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 163–176. ACM, 2013.

[143] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using nonparametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006.

[144] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.

[145] Gelareh Taban and Virgil D Gligor. Efficient handling of adversary attacks in aggregation applications. In *Computer Security-ESORICS 2008*, pages 66–81. Springer, 2008.

[146] Enhua Tan, Lei Guo, Songqing Chen, Xiaodong Zhang, and Yihong Zhao. Unik-unsupervised social network spam detection. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 479–488. ACM, 2013.

[147] D Taylor and C TechCrunch. Startup claims 80% of its facebook ad clicks are coming from bots, 2012.

[148] Dinh Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *NSDI*, volume 9, pages 15–28, 2009.

[149] Nguyen Tran, Jinyang Li, Lakshminarayanan Subramanian, and Sherman SM Chow. Optimal sybil-resilient node admission control. In *INFOCOM, 2011 Proceedings IEEE*, pages 3218–3226. IEEE, 2011.

[150] Bimal Viswanath, M Ahmad Bashir, Mark Crovella, Saikat Guha, MSR India, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Towards detecting anomalous user behavior in online social networks. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*, 2014.

[151] Bimal Viswanath, Mainack Mondal, Krishna P Gummadi, Alan Mislove, and Ansley Post. Canal: Scaling social network-based sybil tolerance schemes. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 309–322. ACM, 2012.

[152] Bimal Viswanath, Ansley Post, Krishna P Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 41(4):363–374, 2011.

[153] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y Zhao. You are how you click- clickstream analysis for sybil detection. In *USENIX Security*, pages 241–256, 2013.

[154] Wei Wei, Fengyuan Xu, Chiu C Tan, and Qun Li. Sybildefender: Defend against sybil attacks in large social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1951–1959. IEEE, 2012.

[155] Yu Xia, Zhifeng Zhao, and Honggang Zhang. Distributed anomaly event detection in wireless networks using compressed sensing. In *Communications and Information Technologies (ISCIT), 2011 11th International Symposium on*, pages 250–255. IEEE, 2011.

[156] Jilong Xue, Zhi Yang, Xiaoyong Yang, Xiao Wang, Lijiang Chen, and Yafei Dai. Votetrust: Leveraging friend invitation graph to defend against social network sybils. In *INFOCOM, 2013 Proceedings IEEE*, pages 2400–2408. IEEE, 2013.

[157] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):2, 2014.

[158] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit a near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE, 2008.

[159] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard defending against sybil attacks via social networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 267–278. ACM, 2006.