Íntegro: Leveraging Victim Prediction for Robust Fake Account Detection in Large Scale OSNs

Yazan Boshmaf[a,*], Dionysios Logothetis[b], Georgos Siganos[a], Jorge Lería[d], Jose Lorenzo[d], Matei Ripeanu[c], Konstantin Beznosov[c], Hassan Halawa[c]

[a] *Qatar Computing Research Institute, HBKU, Doha, Qatar*
[b] *Telefonica Research, Barcelona, Spain*
[c] *University of British Columbia, Vancouver, Canada*
[d] *Tuenti, Telefonica Digital, Madrid, Spain*

Abstract

Detecting fake accounts in online social networks (OSNs) protects both OSN operators and their users from various malicious activities. Most detection mechanisms attempt to classify user accounts as real (i.e., benign, honest) or fake (i.e., malicious, Sybil) by analyzing either user-level activities or graph-level structures. These mechanisms, however, are not robust against adversarial attacks in which fake accounts cloak their operation with patterns resembling real user behavior.

In this article, we show that victims—real accounts whose users have accepted friend requests sent by fakes—form a distinct classification category that is useful for designing robust detection mechanisms. In particular, we present Íntegro—a robust and scalable defense system that leverages victim classification to rank most real accounts higher than fakes, so that OSN operators can take actions against low-ranking fake accounts. Íntegro starts by identifying potential victims from user-level activities using supervised machine learning. After that, it annotates the graph by assigning lower weights to edges incident to potential victims. Finally, Íntegro ranks user accounts based on the landing probability of a short random walk that starts from a known real account. As this walk is unlikely to traverse low-weight edges in a few steps and land on fakes, Íntegro achieves the desired ranking.

*Corresponding author. Tel.: +1 (604) 822-2872.
Email address: boshmaf@ece.ubc.ca (Yazan Boshmaf).
Work done while at the University of British Columbia.

We implemented Íntegro using widely-used, open-source distributed computing platforms, where it scaled nearly linearly. We evaluated Íntegro against SybilRank, which is the state-of-the-art in fake account detection, using real-world datasets and a large-scale deployment at Tuenti—the largest OSN in Spain with more than 15 million active users. We show that Íntegro significantly outperforms SybilRank in user ranking quality, with the only requirement that the employed victim classifier is better than random. Moreover, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher precision in fake account detection, as compared to SybilRank.

Keywords: Online social networks, fake account detection, victim account prediction, social infiltration, socialbots

---

1. Introduction

The rapid growth of online social networks (OSNs), such as Facebook, Tuenti, RenRen, and LinkedIn, has been followed by an increased interest in abusing them. Due to their open nature, OSNs are particularly vulnerable to the *Sybil attack* [1], where an attacker creates multiple fake accounts, each called a Sybil, and joins a target OSN for various adversarial objectives.

1.1. Motivation

In its 2014 earnings report, Facebook estimated that 15 millions (1.2%) of its monthly active users are in fact "undesirable," representing fake accounts that are used in violation of the site's terms of service [2]. For such OSNs, the existence of fakes leads advertisers, developers, and investors to distrust their reported user metrics, which negatively impacts their revenues [3]. Attackers create and automate fake accounts for various malicious activities, including social spamming [4], malware distribution [5], private data collection [6], and even political astroturfing [7]. It is thus important for OSNs to detect fake accounts as quickly and accurately as possible.

1.2. Research Problem

Most OSNs employ defense systems that automatically flag fake accounts by analyzing user-level activities or graph-level structures. As automated account suspension is inapplicable in practice [8], these accounts are pooled for manual verification by experienced analysts, who maintain a ground-truth for fake and real accounts [9].

Traditionally, there are two main approaches for detecting fake accounts. In the first approach, unique *features* are extracted from user activities (e.g., frequency of friend requests, fraction of accepted requests), after which they are applied to a binary fake account classifier that has been trained offline using machine learning techniques [9]. In the second approach, an OSN is modeled as a *graph*, with nodes representing user accounts and edges representing social relationships (e.g., friendships). Given the assumption that *fakes can befriend only few real accounts*, the graph is partitioned into two regions separating real accounts from fakes, with a narrow passage between them [10]. While these techniques are effective against naïve attacks, various studies showed they are inaccurate in practice and can be easily evaded [11, 6, 12]. For example, an attacker can cheaply create fakes that resemble real users, circumventing feature-based detection, or use simple social engineering tactics to befriend a large number of real users, invalidating the assumption behind graph-based detection.

To accommodate these shortcomings, we consider attackers who can create and automate fake accounts on a large scale (i.e., operate of network of malicious socialbots [6, 15]). Each automated fake account can perform social activities similar to those of real users, including befriending other users. As such, we tackle the following question under this threat model: "How can we design an *effective* and *efficient* defense mechanism that aids OSNs in detecting automated fake accounts?"

1.3. Importance and Implications

If an OSN can detect fakes efficiently and effectively, it can improve the experience of its users by thwarting spam messages and other abusive content. In addition, the OSN can increase the credibility of its user metrics and enable third parties to consider its user accounts as authentic digital identities [13]. Moreover, the OSN can better utilize the time of its analysts who manually inspect and validate accounts. For example, Tuenti—the largest OSN in Spain with 15M active users—estimates that only 5% of the accounts inspected based on user reports are in fact fake, which signifies the inefficiency of this manual process [8]. The OSN can also selectively enforce abuse mitigation techniques, such as CAPTCHA challenges [9] and photo-based social authentication [14], to suspicious accounts while running at a lower risk of annoying benign, real users.

1.4. Proposed Solution

We present Íntegro—a robust and scalable defense system that helps OSN operators identify fake accounts, *which can befriend many real accounts*, via a user ranking scheme.[1] We designed Íntegro for OSNs whose users declare bidirectional social relationships (e.g., Tuenti, Facebook, RenRen, LinkedIn), with the ranking process being completely transparent to users. While the ranking scheme is graph-based, the graph is preprocessed first and annotated with information derived from feature-based detection techniques. This new approach of integrating user-level activities into graph-level structures positions Íntegro as the first feature-and-graph-based detection mechanism.

Our design is based on the observation that *victims*—real accounts whose users have accepted friend requests sent by fakes—are useful for designing robust fake account detection mechanisms. In particular, Íntegro uses basic account features, which are cheap to extract from user-level activities (e.g., gender, number of friends, time since last update), to train a victim classifier in order to identify *potential victims* in the OSNs. As attackers do not control victim accounts nor their activities, a victim classifier is inherently more resilient to adversarial attacks than a similarly-trained fake account classifier. Moreover, as victims are directly connected to fakes in the graph, they represent a natural "borderline" that separates real accounts from fakes.

Íntegro makes use of this observation by assigning lower weights to edges incident to potential victims, after which it ranks user accounts based on the landing probability of a modified random walk that starts from a known real account. In particular, the walk is "short," as it is terminated early before it converges. The walk is also "supervised," as it is biased towards traversing nodes that are reachable through higher-weight paths. Therefore, this modified random walk is likely to stay within the subgraph consisting of real accounts, and so *most real accounts receive higher ranks than fakes.* Unlike SybilRank [8], which is the state-of-the-art in fake account detection, we do not assume sparse connectivity between real and fake accounts. This makes Íntegro the first fake account detection system that is robust against *social infiltration*, where fakes befriend a large number of real accounts [15].

[1] In Spanish, the word "íntegro" means integrated, which suites our approach of integrating user-level activities into graph-level structures.

1.5. Security Guarantee

For an OSN consisting of n accounts, Íntegro takes $O(n \log n)$ time to complete its computation. For attackers who randomly establish a set E_a of edges between victim and fake accounts, Íntegro guarantees that at most $O(\text{vol}(E_a) \log n)$ fakes are assigned ranks similar to or higher than real accounts in the worst case, where the *volume* $\text{vol}(E_a)$ is the sum of weights on edges in E_a. This bound represents an improvement factor of $O\left(|E_a|/\text{vol}(E_a)\right)$ over SybilRank. In addition, even with a random victim classifier that labels each account as a victim with 0.5 probability, Íntegro ensures that $\text{vol}(E_a)$ is at most equals to $|E_a|$, resulting in the same asymptotic bound as SybilRank.

In other words, Íntegro outperforms SybilRank using a victim classifier that is better than random and yields the same performance if a random victim classifier is employed. For a complete formal analysis of the security guarantee provided by Íntegro, we refer the interested reader to Appendix A and Appendix B.

1.6. Evaluation Results

We systematically evaluated Íntegro against SybilRank under different attack scenarios using real-world datasets collected from Facebook and Tuenti, in addition to a large-scale deployment on the latter OSN. We chose SybilRank among others because it was shown to outperform known contenders [8], including EigenTrust [16], SybilGuard [17], SybilLimit [18], SybilInfer [19], Mislove's method [20], and GateKeeper [21]. In addition, as SybilRank depends on a user ranking scheme that is similar to ours—although on an unweighted graph—evaluating against SybilRank allowed us to clearly show the impact of leveraging victim classification on fake account detection.

The evaluation results show that Íntegro consistently outperforms SybilRank in ranking quality, especially as the fakes infiltrate an increasing number of victims, that is, as E_a grows large. In particular, Íntegro resulted in up to 30% improvement over SybilRank in the *area under ROC curve* (AUC) of the ranking, which represents the probability that a random real account is ranked higher than a random fake account [20]. In fact, Íntegro achieved an AUC greater than 0.92 as $|E_a|$ increased, while SybilRank resulted in an AUC as low as 0.71 under the same setting.

In practice, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher *precision* in fake account detection, where ideally fakes should be located at the bottom of the ranked list. In particular, for the bottom 20K low-ranking users, Íntegro achieved 95% precision, as compared

to 43% by SybilRank and 5% by Tuenti's user-based abuse reporting system. More importantly, the precision significantly decreased as we inspected higher ranks in the list, which means Íntegro consistently placed most of the fakes at the bottom of the list, unlike SybilRank. The only requirement for Íntegro to outperform SybilRank is to train a victim classifier that is better than random. This can be easily achieved during the cross-validation phase by deploying a victim classifier with an AUC greater than 0.5. In our deployment, the victim classifier was 52% better than random with an AUC of 0.76, although it was trained using low-cost features.

We implemented Íntegro on top of Mahout[2] and Giraph[3], which are widely deployed, open-source distributed machine learning and graph processing systems, respectively. Using a synthetic benchmark of an OSN consisting of up to 160M users, Íntegro scaled nearly linearly with number of users. For the largest graph with 160M nodes, it took Íntegro less than 30 minutes to finish its computation on 33 commodity machines.

1.7. Contributions

In summary, this work makes the following contributions:

- *Leveraging victim classification for fake account detection.* We designed and analyzed Íntegro—a fake account detection system that relies on a novel technique for integrating user-level activities into graph-level structures. Íntegro uses supervised machine learning with features extracted from user-level activities to identify potential victims of fakes. By weighting the graph such that edges incident to potential victims have lower weights than others, Íntegro guarantees that most real accounts are ranked higher than fakes. These ranks are derived from the landing probability of a modified random walk that starts from a known real account. To our knowledge, Íntegro is the first detection system that is robust against adverse manipulation of the graph, where fakes follow an adversarial strategy to befriend a large number of accounts, real or fake, in order to evade detection (Sections 3 and 4).

- *Implementation and evaluation.* We implemented Íntegro on top of open-source distributed systems that run on commodity machines with-

[2] `http://mahout.apache.org`
[3] `http://giraph.apache.org`

out specialized hardware. We evaluated Íntegro against SybilRank using real-world datasets and a large-scale deployment at Tuenti. In practice, Íntegro has allowed Tuenti to detect at least 10 times more fakes than their current user-based abuse reporting system, where reported users are not ranked. With an average of 16K flagged accounts a day [8], Íntegro has saved Tuenti hundreds of man hours in manual verification by robustly ranking user accounts (Sections 5 and 6).

2. Background and Related Work

We first outline the threat model we assume in this work. We then present required background and related work on automated fake account detection, social infiltration, analyzing victim accounts in OSNs, abuse mitigation, and maintaining a ground-truth.

2.1. Threat Model

We focus on OSNs such as Facebook, RenRen, and Tuenti, which are open to all and allow users to declare bilateral relationships (i.e., friendships).

2.1.1. Capabilities

We consider attackers who are capable of creating and automating user accounts, or *fakes*, on a large scale [15]. Each fake account, also referred to as a *socialbot* [22], can perform social activities similar to those of real users. This includes sending friend requests and posting social content. We do not consider attackers who are capable of hijacking real accounts, as there are existing detection systems that tackle this threat, such as COMPA [23]. We focus on detecting fake accounts that can befriend a large number of benign users in order to mount subsequent attacks, as we describe next.

2.1.2. Objectives

The objectives of an attacker include distributing spam and malware, misinforming the public, and collecting private user data on a large scale. To achieve these objectives, the attacker has to *infiltrate* the target OSN by using the fakes to befriend a large number of real accounts. Such an infiltration is required because isolated fake accounts cannot directly interact with or promote content to users in the OSN [15]. This is also evident by a thriving underground market for social infiltration. For example, attackers can have their fake accounts befriend 1K users on Facebook for $26 or less [24].

2.1.3. Victims

We refer to accounts whose users have accepted friend requests from fake accounts as *victims*. We refer to friendships between victim and fake accounts as *attack edges*. Victim accounts are a subset of *real* accounts, which are accounts created and controlled by benign users who socialize with others in a non-adversarial setting. Also, we refer to accounts whose users are more susceptible to social infiltration and are likely to be victims as *potential victims*. We use the terms "account," "profile," and "user" interchangeably but make the distinction when deemed necessary.

2.2. Fake Account Detection

From a systems design perspective, most of today's fake account detection mechanisms are either feature-based or graph-based, depending on whether they utilize machine learning or graph analysis techniques in order to identify fakes. Next, we discuss each one of these approaches in detail.

2.2.1. Feature-based detection

This approach relies on user-level activities and its account details (i.e., user logs, profiles). By identifying unique features of an account, one can classify each account as fake or real using various machine learning techniques. For example, Facebook employs an "immune system" that performs real-time checks and classification for each read and write action on its database, which are based on features extracted from user accounts and their activities [9].

Yang *et al.* used a ground-truth provided by RenRen to train an SVM classifier in order to detect fake accounts [25]. Using simple features, such as frequency of friend requests, fraction of accepted requests, and per-account clustering coefficient, the authors were able to train a classifier with 99% true-positive rate (TPR) and 0.7% false-positive rate (FPR).

Stringhini *et al.* used honeypot accounts to collect data describing various user activities in OSNs [26]. By analyzing the collected data, they were able to build a ground-truth for real and fake accounts, with features similar to those outlined above. The authors trained two random forests classifiers to detect fakes in Facebook and Twitter, ending up with 2% FPR and 1% false-negative rate (FNR) for Facebook, and 2.5% FPR and 3% FNR for Twitter.

Wang *et al.* used a click-stream dataset provided by RenRen to cluster user accounts into "similar" behavioral groups, corresponding to real or fake accounts [27]. The authors extracted both session and clicks features, including average clicks per session, average session length, the percentage of clicks

used to send friend requests, visit photos, and share content. With these features, the authors were able to calibrate a cluster-based classifier with 3% FPR and 1% FNR, using the METIS clustering algorithm [28].

Fire *et al.* developed the Social Privacy Protector (SPP) software, which is a set of applications for Facebook that aim to improve user account privacy [29]. The applications analyze a user's friends list in order to determine accounts that may pose a risk to the user's privacy. Such accounts could then be restricted by users from accessing their profile information. Using data collected from the SSP deployment on Facebook, the authors tested several machine learning classifiers to detect fake accounts, including Naïve Bayes, Rotation Forest and Random Forest.

Viswanath *et al.* used unsupervised anomaly detection techniques in order to identify accounts that demonstrate malicious behavior, such as fake, compromised or colluding accounts [30]. The authors presented a technique that is able to accurately model the typical behavior of user accounts and to correctly identify significant deviations from the norm, based on Principle Component Analysis (PCA). The presented technique makes use of temporal, spatial and spatio-temporal features as input features. Using ground-truth data from Facebook, the authors evaluated the PCA-based classifier demonstrating a detection rate of over 66% with a FPR of less than 0.3%.

Other approaches to detect fake accounts are based on the observation that, in a variety of OSN applications, fake accounts tend to perform loosely synchronized actions from a limited set of IP addresses. Based on this observation, Cao *et al.* designed and implemented a feature-based malicious account detection system called SynchroTrap [31]. SynchroTrap uses a single-linkage hierarchical clustering algorithm to detect similarities in user actions in order to identify fake accounts. The authors deployed SynchroTrap at Facebook, where it was able to detect more than two million fake accounts. Similarly, Stringhini *et al.* developed EvilCohort [32], a system that is capable of detecting accounts that are accessed by a common set of infected machines such as a botnet. EvilCohort only requires that a mapping exists between online accounts and IP addresses in order to detect malicious accounts on any type of online web service (e.g., OSNs, webmail). only builds a bipartite graph between the set of online accounts and the set of IP addresses, and performs clustering in order to find account communities that are accessed by a common set of IP addresses. The authors evaluated EvilCohort on two real-world datasets, and it was able to identify more than one million malicious accounts.

Even though feature-based detection scales to large OSNs, it is still relatively easy to circumvent. This is because it depends on features describing activities of known fakes in order to identify unknown ones. In other words, attackers can evade detection by adversely modifying the content and activity patterns of their fakes, leading to an arms race [33, 34, 35]. Also, feature-based detection does not provide any formal security guarantees and often results in a high FPR in practice. This is partly attributed to the large variety and unpredictability of behaviors of users in adversarial settings [8].

With Íntegro, we *use feature-based detection to identify potential victims in a non-adversarial setting.* In particular, the dataset used to train a victim classifier includes features of only known real accounts that have either accepted or rejected friend requests send by known fakes. As real accounts are controlled by benign users who are not adversarial, a feature-based victim account classifier is harder to circumvent than a similarly-trained fake account classifier. As we discuss in Section 4, we only require the victim classification to be better than random in order to outperform the state-of-the-art in fake account detection.

2.2.2. Graph-based detection

As a response to the lack of formal security guarantees in feature-based detection, the state-of-the-art in fake account detection utilizes a graph-based approach instead. In this approach, an OSN is modeled as a finite graph, with nodes representing user accounts and edges between nodes representing social relationship. Assuming that fake accounts can establish a small and limited number of attack edges, the subgraph induced by the set of all real accounts is sparsely connected to fakes, that is, the *cut* over attack edges is sparse.[4] Graph-based detection mechanisms make this assumption, and attempt to find such a sparse cut with formal guarantees [36, 37, 38]. For example, Tuenti employs SybilRank to rank accounts according to their perceived likelihood of being fake, based on structural properties of its social graph [8].

Yu *et al.* were among the first to utilize social networks to defend against Sybil attacks in peer-to-peer and other distributed systems (e.g., DHTs) [17, 18]. The authors developed a technique that labels each account as either fake or real based on multiple, modified random walks. This binary classification

[4] A cut is a partition of nodes into two disjoint subsets. Visually, it is a line that cuts through or crosses over a set of edges in the graph (see Fig. B.2).

is used to partition the graph into two smaller subgraphs that are sparsely interconnected via attack edges, separating real accounts from fakes. They also proved that in the worst case $O(|E_a| \log n)$ fakes can be misclassified, where $|E_a|$ is the number of attack edges and n is the number of accounts in the graph. Accordingly, it is sufficient for the attacker to establish $\Omega(n/\log n)$ attack edges in order to evade this detection scheme with 0% TPR.

Fire *et al.* proposed a technique that allows for the detection of malicious profiles, such as fake and spam accounts, using topological features extracted from a social network [39]. Based on the observation that OSNs are scale-free and have a community structure where legitimate users are typically members of only a small group of communities, the approach uses the social graph's topology to detect malicious accounts that randomly connect to other users from different communities. The proposed technique was evaluated on several social networks, including Google+, and was demonstrated to be effective in detecting fake and spam accounts.

In order to detect fake accounts that engage in friend request spamming in OSNs, Cao *et al.* proposed a system called Rejecto [40]. Rejecto is based on the premise that friend request rejections could potentially be used to identify fake accounts. The authors propose a technique that partitions a social graph into two regions: friend spammers (fake accounts) and legitimate users. The partitioning is done such that the aggregate acceptance rate of friend requests between the two partitions is minimized. The authors extend the Kerninghan-Lin heuristic in order to obtain the graph cut, and thus, were able to detect fake accounts that engage in friend spamming.

Viswanath *et al.* employed traditional community detection techniques to identify fake accounts in OSNs [20]. In general, community detection decomposes a given graph into a number of tightly-knit subgraphs that are loosely connected to each other, where each subgraph is called a *community* [41, 42]. By expanding a community starting with known real accounts [43], the authors were able to identify the subgraph which contains mostly real accounts. Recently, however, Alvisi *et al.* showed that such a local community detection technique can be easily circumvented if fake accounts establish sparse connectivity among themselves [10].

As binary classification often leads to high FPR [20], Cao *et al.* proposed user ranking instead so that most fake accounts are ranked lower than real accounts [8]. The authors developed SybilRank, a fake account detection system that assigns each account a rank describing how likely it is to be fake based on a modified random walk, in which a lower rank means the account is

more likely to be fake. They also proved that $O(|E_a| \log n)$ fakes can outrank real accounts in the worst case, given the fakes establish $|E_a|$ attack edges with victims at random.

Cao et. al proposed SybilFence [44], which improves on prior social graph-based Sybil detection techniques. In essence, SybilFence is based on the observation that fake accounts will inevitably receive a significant amount of negative user feedback over the course of their operation. As such, the authors proposed discounting the weights on the social edges of users that have received negative feedback with the goal of limiting the impact of the attack edges of Sybil accounts. This weighted-graph ranking model thus attempts to reduce the aggregate value of Sybil attack edges thereby improving Sybil account detection accuracy. The authors adapted SybilRank [8] in order to implement the negative feedback user weighting scheme and, through simulations, demonstrated an improvement of up to 20% in terms of the probability of ranking non-Sybil users higher than Sybils.

While graph-based detection offers the desirable security guarantees, real-world social graphs do not conform with the main assumption on which it depends. In particular, various studies confirmed that attackers can infiltrate OSNs on a large scale by deceiving users into befriending their fakes [11, 6, 12]. As we discuss next, social infiltration renders graph-based fake account detection ineffective in practice.

With Íntegro, we *do not assume fake accounts are limited by how many attack edges they can establish.* Instead, we identify potential victims of fakes and leverage this information to weight the graph. Through a user ranking scheme, we bound the security guarantee by the aggregate weight on attack edges, $\text{vol}(E_a)$, rather than their number, $|E_a|$. By assigning lower weights to edges incident to potential victims, we were able to upper bound the value of $\text{vol}(E_a)$ by $|E_a|$, as we show in Section 4.

2.3. Social Infiltration: A Case Study on Facebook

In early 2011, we conducted a study to evaluate how easy it is to infiltrate large OSNs such as Facebook [6]. We used 100 automated fake accounts to send friend requests to 9.6K real users, where each user received exactly one request. We summarize the main results and implications in what follows.

2.3.1. Main results

We found that users are not careful in befriending other users, especially when they share mutual friends with the requester. This behavior was ex-

ploited by the fakes to achieve large-scale social infiltration with a success rate of up to 80%, in which case the fakes shared at least 11 mutual friends with the victims. In particular, we reported two main results that are important for designing fake account detection systems. First, *some users are more likely to be victims than others, which partly depends on factors related to their social structure.* As shown in Fig. B.1a, the more friends a user had, the more likely the user was to accept friend requests sent by fakes posing as strangers, regardless of their gender or number of mutual friends. Second, *attack edges are generally easy to establish in OSN such as Facebook.* As suggested in Fig. B.1b, an attacker can establish enough attack edges such that there is no sparse cut separating real accounts from fakes [38].

2.3.2. Implications

The study suggests that one can identify potential victims in OSNs from user-level activities using low-cost features (e.g., number of friends). In addition, the study shows that graph-based detection mechanisms that rely solely on the graph structure are not robust against social infiltration. As social infiltration is prominent in other OSNs [45, 46], one should extend the threat model of fake account detection to consider attackers who are capable of large-scale social infiltration.

2.4. Analyzing Victim Accounts

While we are the first to use potential victims to separate fakes from real accounts, others have analyzed victim accounts as part of the larger cyber criminal ecosystem in OSNs [47].

Wagner *et al.* developed predictive models to identify users who are more susceptible to social infiltration in Twitter [12]. They found that susceptible users, also called potential victims, tend to use Twitter for conversational purposes, are more open and social since they communicate with many different users, use more socially welcoming words, and show higher affection than non-susceptible users.

Yang *el al.* studied the cyber criminal ecosystem on Twitter [48]. They found that victims fall into one of three categories. The first are *social butterflies* who have large numbers of followers and followings, and establish social relationships with other accounts without careful examination. The second are *social promoters* who have large following-follower ratios, larger following numbers, and a relatively high URL ratios in their tweets. These victims use Twitter to promote themselves or their business by actively following other

accounts without consideration. The last are *dummies* who post few tweets but have many followers. In fact, these victims are dormant fake accounts at an early stage of their abuse.

2.5. Abuse Mitigation and the Ground-truth

Due to the inapplicability of automated account suspension, OSNs employ abuse mitigation techniques, such as CAPTCHA challenges [9] and photo-based social authentication [14], in order to rate-limit accounts that have been automatically flagged as fake. These accounts are pooled for manual inspection by experienced analysts who maintain a ground-truth for real and fake accounts along with their features, before suspending or removing verified fake accounts [9, 25, 8, 49].

While maintaining an up-to-date ground-truth is important for retraining deployed classifiers and estimating their effectiveness in practice, it is rather a time-consuming and non-scalable task. For example, on an average day, each analyst at Tuenti inspects 250–350 accounts an hour, and for a team of 14 employees, up to 30K accounts are inspected per day [8]. It is thus important to rank user accounts in terms of how likely they are to be fake in order to prioritize account inspection by analysts. Íntegro offers this functionality and consequently leads to a faster reaction against potential abuse by fakes, benefiting both OSN operators and their users.

3. Intuition, Goals, and Model

We now introduce Íntegro, an automated fake account detection system that is *robust against social infiltration*. We first present the intuition of our design, followed by its goals and model.

3.1. Intuition

We start with the premise that some users are more likely to be victims than others. If we can train a classifier to identify potential victims with high probability, we may be able to use this information to find the cut which separates fakes from real accounts in the graph. As victims are benign users who are not adversarial, the output of this classifier represents a reliable information which we can integrate in the graph.

To find such a cut, we can define a graph weighting scheme that assigns edges incident to potential victims lower weights than others, where weight values are calculated from victim classification probabilities. In a weighted

graph, the sparsest cut is the cut with the smallest *volume*, which is the sum of weights on edges across the cut. Given an accurate victim classifier, this cut is expected to cross over some or all attack edges, effectively separating real accounts from fakes, even if the number of attack edges is large. We aim to find such a cut using a ranking scheme that ideally assigns higher ranks to nodes in one side of the cut than the other, which represents one way to separate real accounts from fakes. This ranking scheme is inspired by similar graph partitioning algorithms proposed by Spielman *et al.* [50], Yu [36], and recently by Cao *et al.* [8].

3.2. Design Goals

Íntegro aims to help OSN operators in detecting fake accounts through a user ranking scheme. In particular, Íntegro has the following design goals:

- *High-quality user ranking (effectiveness).* The system should ideally assign higher ranks to real accounts than fakes. If not, it should limit the number of fakes that might rank similar to or higher than real accounts. In practice, the ranking should *have an area under ROC curve (AUC) that is greater than 0.5 and closer to 1*, where the AUC represents the probability of a random real accounts to rank higher than a random fake account [20]. Also, the system should be robust against social infiltration under real-world attack strategies. Given a ranked list of users, a high percentage of the users at the bottom of the list should be fake. This percentage, which represents the *precision* of detection, should significantly decrease as we go up in the list.

- *Scalability (efficiency).* The system should have a practical computational cost which allows it to scale to large OSNs. In other words, it should *scale nearly linearly with number of user accounts in the OSN*, and deliver ranking results in only few minutes. The system should be able to extract useful, low-cost features and process large graphs on commodity machines, so that OSN operators can deploy it on their existing computer clusters.

3.3. System Model

As shown in Fig. B.2, we model an OSN as an undirected, finite graph $G = (V, E)$, where each node $v_i \in V$ represents a user account and each edge $\{v_i, v_j\} \in E$ represents a bilateral social relationship among v_i and v_j. In the graph G, there are $n = |V|$ nodes and $m = |E|$ edges.

3.3.1. Attributes

Each node v_i has a *degree* $\deg(v_i)$ that is equal to the sum of weights on edges incident to it. Also, v_i has a *feature vector* $\mathcal{A}(v_i)$, where each entry $a_j \in \mathcal{A}(v_i)$ describes a feature or an attribute of the account v_i. Each edge $\{v_i, v_j\} \in E$ has a *weight* $w(v_i, v_j) \in (0, 1]$, which is initially set to 1.

3.3.2. Regions

The node set V is divided into two disjoint sets, V_r and V_f, representing real and fake accounts, respectively. We refer to the subgraph induced by V_r as the *real region* G_r, which includes all real accounts and the friendships between them. Likewise, we refer to the subgraph induced by V_f as the *fake region* G_f. The regions are connected by a set of attack edges E_a between victim and fake accounts. We assume the OSN operator is aware of a small set of *trusted accounts* V_t, also called *seeds*, which are known to be real accounts and are not victims.

4. System Design

In what follows, we describe in detail the design behind Íntegro. We start with a short overview of our approach, after which we proceed with a detailed description of each system component.

4.1. Overview

At first, Íntegro trains a victim classifier using low-cost features extracted from user-level activities. This feature-based classifier is used to identify potential victims in the graph, each with some probability (Section 4.2). After that, Íntegro calculates new edge weights from the probabilities computed by the victim classifier, so that edges incident to potential victims have lower weights than others. Íntegro then ranks user accounts based on the landing probability of a modified random walk that starts from a trusted account, or a seed node, picked at random from all trusted accounts (Section 4.3).

The random walk is "short" because it is terminated after $O(\log n)$ steps, early before it converges. The walk is "supervised," as it is biased towards traversing nodes which are reachable via higher-weight paths. This modified random walk has a higher probability to stay in the real region of the graph, as it is highly unlikely to escape into the fake region in few steps through low-weight attack edges. Therefore, Íntegro ranks most of real accounts higher

than fakes, given a seed selection strategy that considers existing community structures in the real region (Section 4.4).

Íntegro takes $O(n \log n)$ time to complete its computation (Section 4.5). In addition, it formally guarantees that at most $O(\text{vol}(E_a) \log n)$ fake accounts can have the same or higher ranks than real accounts in the worst case, given the fakes establish $|E_a|$ attack edges at random (Section 4.6).

4.2. Identifying Potential Victims

For each account v_i, Íntegro extracts a feature vector $\mathcal{A}(v_i)$ from its user-level activities. A subset of the feature vectors is selected to train a binary classifier in order to identify potential victims in the OSN using supervised machine learning, as follows.

4.2.1. Feature engineering

Extracting and selecting useful features from user-level activities can be both challenging and time consuming. To be efficient, we seek *low-cost features* which could be extracted in $O(1)$ time per user. One candidate location for extracting such features is the profile page of user accounts, where features are readily available (e.g., a Facebook profile page). However, low-cost features are expected to be statistically "weak," which means they may not strongly correlate with the *label* of a user account (i.e., victim or not). As we explain later, we require the victim classifier to be better than random in order to deliver robust fake account detection. This requirement, fortunately, is easy to satisfy. In particular, we show in Section 5 that an OSN can train and cross-validate a victim classifier that is up to 52% better than random, using strictly low-cost features.

4.2.2. Supervised machine learning

For each user v_i, Íntegro computes a *vulnerability score* $p(v_i) \in (0, 1)$ that represents the probability of v_i to be a victim. Given an *operating threshold* $\alpha \in (0, 1)$ with a default value of $\alpha = 0.5$, we say v_i is a potential victim if $p(v_i) \geq \alpha$. To compute vulnerability scores, Íntegro uses random forests (RF) learning algorithm [51] to train a victim classifier, which given $\mathcal{A}(v_i)$ and α, decides whether the user v_i is a potential victim with a score $p(v_i)$. We picked the RF learning algorithm because it is both efficient and robust against model over-fitting [52]. Íntegro takes $O(n \log n)$ time to extract n feature vectors and train an RF victim classifier. It also takes $O(n)$ to compute a vulnerability score for all users, given their feature vectors and the trained victim classifier.

4.2.3. Robustness

As attackers do not control victims, a victim classifier is inherently more resilient to adversarial attacks than similarly-trained fake account classifier. Let us consider one concrete example. In the "boiling-frog" attack [33], fake accounts can force a classifier to tolerate abusive activities by slowly introducing similar activities to the OSN. Because the OSN operator has to retrain deployed classifiers in order to capture new behaviors, a fake account classifier will learn to tolerate more and more abusive activities, until the attacker can launch a full-scale attack without detection [6]. When identifying potential victims, however, this is only possible if the real accounts used for training the victim classifier have been hijacked. This situation can be avoided by verifying the accounts, as described in Section 2.5.

4.3. Leveraging Victim Classification for Robust User Ranking

To rank user accounts in the OSN, Íntegro computes the probability of a modified random walk to land on each user v_i after k steps, where the walk starts from a trusted user account picked at random. For simplicity, we refer to the probability of a random walk to land on a node as its *trust value*, so the probability distribution of the walk at each step can be modeled as a *trust propagation process* [53]. In this process, a weight $w(v_i, v_j)$ represents the rate at which trust may propagate from either side of the edge $\{v_i, v_j\} \in E$. We next describe this process in detail.

4.3.1. Trust propagation

Íntegro utilizes the *power iteration method* to efficiently compute trust values [54]. This method involves successive matrix multiplications where each element of the matrix is the transition probability of the random walk from one node to another. Each iteration computes the trust distribution over nodes as the random walk proceeds by one step. Let $T_k(v_i)$ denote the trust collected by each node $v_i \in V$ after k iterations. Initially, the *total trust*, denoted by $\tau \geq 1$, is evenly distributed among the trusted nodes in V_t:

$$T_0(v_i) = \begin{cases} \tau/|V_t| & \text{if } v_i \in V_t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The process then proceeds as follows:

$$T_k(v_i) = \sum_{\{v_i, v_j\} \in E} T_{k-1}(v_j) \cdot \frac{w(v_i, v_j)}{\deg(v_j)}, \tag{2}$$

where in iteration k, each node v_i propagates its trust $T_{k-1}(v_i)$ from iteration $k-1$ to each neighbour v_j, proportionally to the ratio $w(v_i, v_j)/\deg(v_i)$. This is required so that the sum of the propagated trust equals $T_{k-1}(v_i)$. The node v_i then collects the trust propagated similarly from each neighbour v_j and updates its trust $T_k(v_i)$. Throughout this process, τ is preserved such that for each iteration $k \geq 1$ we have:

$$\sum_{v_i \in V} T_{k-1}(v_i) = \sum_{v_i \in V} T_k(v_i) = \tau. \tag{3}$$

Our goal is to ensure that most real accounts collect higher trust than fake accounts. That is, we seek to limit the portion of τ that escapes the real region G_r and enters the fake region G_f. To achieve this property, we make the following modifications.

4.3.2. Adjusted propagation rates

In each iteration k, the aggregate rate at which τ may enter G_f is strictly limited by the sum of weights on the attack edges, which we denote by the *volume* $\mathrm{vol}(E_a)$. Therefore, we aim to adjust the weights in the graph such that $\mathrm{vol}(E_a) \in (0, |E_a|]$, without severely restricting trust propagation in G_r. We accomplish this by assigning smaller weights to edges incident to potential victims than other edges. In particular, each edge $\{v_i, v_j\} \in E$ keeps the default weight $w(v_i, v_j) = 1$ if v_i and v_j are not potential victims. Otherwise, we modify the weight as follows:

$$w(v_i, v_j) = \min\left\{1, \beta \cdot (1 - \max\{p(v_i), p(v_j)\})\right\}, \tag{4}$$

where β is a scaling parameter with a default value of $\beta = 2$. Now, when $\mathrm{vol}(E_a) \to 0$, the portion of τ that enters G_f reaches zero as desired.

For proper degree normalization, we introduce a self-loop $\{v_i, v_i\}$ with weight $w(v_i, v_i) = (1 - \deg(v_i))/2$ whenever $\deg(v_i) < 1$. Notice that self-loops are considered twice in degree calculation.

4.3.3. Early termination

In each iteration k, the *trust vector* $T_k(V) = \langle T_k(v_1), \ldots, T_k(v_n)\rangle$ describes the distribution of τ throughout the graph. As $k \to \infty$, the vector converges to a stationary distribution $T_\infty(V)$, as follows [55]:

$$T_\infty(V) = \left\langle \tau \cdot \frac{\deg(v_1)}{\mathrm{vol}(V)}, \ldots, \tau \cdot \frac{\deg(v_n)}{\mathrm{vol}(V)} \right\rangle, \tag{5}$$

where the volume $\text{vol}(V)$ in this case is the sum of degrees of nodes in V. In particular, $T_k(V)$ converges after k reaches the *mixing time* of the graph, which is much larger than $O(\log n)$ iterations for various kinds of social networks [56, 57, 41]. Accordingly, we terminate the propagation process early before it converges after $\omega = O(\log n)$ iterations.

4.3.4. Degree normalization

As described in Equation 5, trust propagation is influenced by individual node degrees. As k grows large, the propagation starts to bias towards high degree nodes. This implies that high degree fake accounts may collect more trust than low degree real accounts, which is undesirable for effective user ranking. To eliminate this bias, we normalize the trust collected by each node by its degree. That is, we assign each node $v_i \in V$ after $\omega = O(\log n)$ iterations a *rank value* $T'_\omega(v_i)$ that is equal to its degree-normalized trust:

$$T'_\omega(v_i) = T_\omega(v_i)/\deg(v_i). \tag{6}$$

Finally, we sort the nodes by their ranks in a descending order.

4.3.5. Example

Fig. B.3 depicts trust propagation in a graph. In this example, we assume each account has a vulnerability score of 0.05 except the victim E, which has a score of $p(E) = 0.95$. The graph is weighted using $\alpha = 0.5$ and $\beta = 2$, and a total trust $\tau = 1000$ in initialized over the trusted nodes $\{C, D\}$.

In Fig. B.3b, after $\omega = 4$ iterations, all real accounts $\{A, B, C, D, E\}$ collect more trust than fake accounts $\{F, G, H, I\}$. Moreover, the nodes receive the correct ranking of $(D, A, B, C, E, F, G, H, I)$, as sorted by their degree-normalized trust. In particular, all real accounts have higher rank values than fakes, where the smallest difference is $T'_4(E) - T'_4(F) > 40$. Notice that real accounts that are not victims have similar rank values, where the largest difference is $T'_4(D) - T'_4(C) < 12$. These sorted rank values, in fact, could be visualized as a stretched-out step function that has a significant drop near the victim's rank value.

If we allow the process to converge after $k > 50$ iterations, the fakes collect similar or higher trust than real accounts, following Equation 5, as shown in Fig. B.3c. In either case, the attack edges $E_a = \{\{E, G\}, \{E, F\}, \{E, H\}\}$ have a volume of $\text{vol}(E_a) = 0.3$, which is 10 times lower than its value if the graph had unit weights, with $\text{vol}(E_a) = 3$.

As we show in Section 5, early termination and propagation rate adjustment are essential for robustness against social infiltration.

4.4. Selecting Trusted Accounts with Existing Community Structures

Íntegro is robust against social infiltration, as it limits the portion of τ that enters G_f by the rate $\text{vol}(E_a)$, regardless of the number of attack edges, $|E_a|$. For the case when there are few attack edges so that G_r and G_f are sparsely connected, $\text{vol}(E_a)$ is already small, even if one keeps $w(v_i, v_j) = 1$ for each attack edge $\{v_i, v_j\} \in E_a$. However, G_r is likely to contain communities [58, 41], where each represents a dense subgraph that is sparsely connected to the rest of the graph. In this case, the propagation of τ in G_r becomes restricted by the sparse inter-community connectivity, especially if V_t is contained exclusively in a single community. We therefore seek a *seed selection strategy* for trusted accounts, which takes into account the existing community structure in the graph.

4.4.1. Seed selection strategy

We pick trusted accounts as follows. First, before rate adjustment, we estimate the community structure in the graph using a community detection algorithm called the *Louvain method* [59]. Second, after rate adjustment, we exclude potential victims and pick small samples of nodes from each detected community at random. Third and last, we inspect the sampled nodes in order to verify they correspond to real accounts that are not victims. We initialize the trust only between the accounts that pass manual verification by experts.

In addition to coping with the existing community structure in the graph, this selection strategy is designed to also reduce the negative impact of *seed-targeting* attacks. In such attacks, fakes befriend trusted accounts in order to adversely improve their ranking, as the total trust τ is initially distributed among trusted accounts. By choosing the seeds at random, however, the attacker is forced to guess the seeds among a large number of nodes. Moreover, by choosing multiple seeds, the chance of correctly guessing the seeds is further reduced, while the amount of trust assigned to each seed in lowered. In practice, the number of seeds depends on available resources for manual account verification, with a minimum of one seed per detected community.

4.4.2. Community detection

We picked the Louvain method as it is both efficient and produces high-quality partitions. The method iteratively groups closely connected communities together to greedily improve the *modularity* of the partition [60], which

is a measure for partition quality. In each iteration, every node represents one community, and well-connected neighbors are greedily combined into the same community. At the end of the iteration, the graph is reconstructed by converting the resulting communities into nodes and adding edges that are weighted by inter-community connectivity. Each iteration takes $O(m)$ time, and only a small number of iterations is required to find the community structure which greedily maximizes the modularity.

While one can apply community detection to identify fake accounts [20], doing so hinges on the assumption that fakes always form tightly-knit communities, which is not necessarily true [25]. This also means fakes can easily evade detection if they establish sparse connectivity among themselves [10]. With Íntegro, we do not make such an assumption. In particular, we consider an attacker *who can befriend a large number of real or fake accounts*, without any formal restrictions.

4.5. Computational Cost

For an OSN with n users and m friendships, Íntegro takes $O(n \log n)$ time to complete its computation, end-to-end. We next analyze the running time of Íntegro in detail.

Recall that users have a limit on how many friends they can have (e.g., 5K in Facebook, 1K in Tuenti), so we have $O(m) = O(n)$. Identifying potential victims takes $O(n \log n)$ time, where it takes $O(n \log n)$ time to train an RF classifier and $O(n)$ time to compute vulnerability scores. Also, weighting the graph takes $O(m)$ time. Detecting communities takes $O(n)$ time, where each iteration of the Louvain method takes $O(m)$ time, and the graph rapidly shrinks in $O(1)$ time. Propagating trust takes $O(n \log n)$ time, as each iteration takes $O(m)$ time and the propagation process iterates for $O(\log n)$ times. Ranking and sorting users by their degree-normalized trust takes $O(n \log n)$ time. So, the running time is $O(n \log n)$.

4.6. Security Guarantees

For the upcoming security analysis, we consider attackers who establish attack edges with victims uniformly at random. For analytical tractability, we assume the real region is *fast mixing*. This means that it takes $O(\log |V_r|)$ iterations for trust propagation to converge in the real region. We refer the reader to Appendix A and Appendix B for a complete formal analysis, including theoretical background and formal proofs.

4.6.1. Sensitivity to mixing time

Similar to SybilRank [8], the ranking quality of Íntegro does not rely on the absolute value of the mixing time in the real region of the social graph. Instead, Íntegro only requires that the whole graph has a longer mixing time than the real region. Under this condition, the early-terminated propagation process results in a gap between the degree-normalized trust of fakes and real accounts. Ideally, the number of iterations that Íntegro performs is set equal to the mixing time of the real region. Regardless of whether the mixing time of the real region is $O(\log n)$ or longer, setting the number of iterations to this value results in almost uniform degree-normalized trust among the real accounts [8]. If the mixing time of the real region is larger than $O(\log n)$, the trust that "escapes" to the fake region is further limited. However, we also run at the risk of starving real accounts that are loosely connected to trusted accounts via few edges. This risk is mitigated by placing the trusted accounts in many communities and by dispersing multiple seeds in each community, thereby ensuring that the trust is initiated somewhere close to those real accounts, as described in Section 4.4.

4.6.2. Main theoretical result

The main security guarantee provided by Íntegro is captured by the following theoretical result:

Theorem 1. *Given a social graph with a fast mixing real region and an attacker who randomly establishes attack edges, the number of fake accounts that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O\left(\mathrm{vol}(E_a) \log n\right)$.*

Proof sketch. Consider an undirected graph $G = (V, E)$ with a fast mixing real region G_r. As weighting a graph changes its mixing time by a constant factor (see Lemma 1), G_r remains fast mixing after rate adjustment.

After $O(\log n)$ iterations, the trust vector $T_\omega(V)$ does not reach its stationary distribution $T_\infty(V)$. Since trust propagation starts from G_r, the fake region G_f gets only a fraction $f < 1$ of the aggregate trust it should receive in $T_\infty(V)$. On the other hand, as the trust τ is conserved during the propagation process (Equation 3), G_r gets $c > 1$ times higher aggregate trust than it should receive in $T_\infty(V)$.

As G_r is fast mixing, each real account $v_i \in V_r$ receives approximately identical rank value of $T'_\omega(v_i) = c \cdot \tau/\mathrm{vol}(V)$, where $\tau/\mathrm{vol}(V)$ is the degree-normalized trust value in $T_\infty(V)$ (Equations 5 and 6). Knowing that G_f is

controlled by the attacker, each fake $v_j \in V_f$ receives a rank value $T'_\omega(v_j)$ that depends on how the fakes inter-connect to each other. However, since the aggregate trust in G_f is bounded, each fake receives on average a rank value of $T'_\omega(v_j) = f \cdot \tau / \mathrm{vol}(V)$, which is less than that of a real account. In the worst case, an attacker can arrange a set $V_m \subset V_f$ of fake accounts in G_f such that each $v_k \in V_m$ receives a rank value of $T'_\omega(v_k) = c \cdot \tau / \mathrm{vol}(V)$, while the remaining fakes receive a rank value of zero. Such a set cannot have more than $(f/c) \cdot \mathrm{vol}(V_f) = O\left(\mathrm{vol}(E_a) \log n\right)$ accounts, as otherwise, f would not be less than 1 and G_f would receive more than it should in $T_\omega(V)$. □

4.6.3. Improvement over SybilRank

Íntegro shares many design traits with SybilRank. In particular, modifying Íntegro by setting $w(v_i, v_j) = 1$ for each $(v_i, v_j) \in E$ will result in a ranking similar to that computed by SybilRank [8]. It is indeed the incorporation of victim classification into user ranking that differentiates Íntegro from other proposals, giving it the unique advantages outlined earlier.

As stated by Theorem 1, the bound on ranking quality relies on $\mathrm{vol}(E_a)$, regardless of how large the set E_a grows. As we weight the graph based on the output of the victim classifier, our bound is sensitive to its classification performance. We next prove that if an OSN operator uses a victim classifier that is *uniformly random*, which means each user account $v_i \in V$ is equally vulnerable with $p(v_i) = 0.5$, then Íntegro is as good as SybilRank in terms of ranking quality:

Corollary 1. *For a uniformly random victim classifier, the number of fakes that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O(|E_a| \log n)$.*

Proof. This classifier assigns each user account $v_i \in V$ a score $p(v_i) = 0.5$. By Equation 4, each edge $\{v_i, v_j\} \in E$ is assigned a unit weight $w(v_i, v_j) = 1$, where $\alpha = 0.5$ and $\beta = 2$. By Theorem 1, the number of fake accounts that rank similar to or higher than real accounts after $\omega = O(\log n)$ iterations is $O\left(\mathrm{vol}(E_a) \log n\right) = O(|E_a| \log n)$. □

By Corollary 1, Íntegro can outperform SybilRank in its ranking quality by a factor of $O\left(|E_a| / \mathrm{vol}(E_a)\right)$, given the used victim classifier is better than random. This can be achieved during the cross-validation phase of the victim classifier, which we thoroughly describe and evaluate in what follows.

5. System Evaluation

We analyzed and evaluated Íntegro against SybilRank using real-world datasets recently collected from Facebook, Tuenti, and arXiv. We also compared both systems through a large-scale deployment at Tuenti in collaboration with its "Site Integrity" team, which has 14 full-time account analysts and 10 full-time software engineers who fight spam and other forms of abuse.

5.1. Compared System

We chose SybilRank for two reasons. First, as discussed in Section 4.6, SybilRank uses a power iteration method on an unweighted version of the graph to rank user accounts. This similarity allowed us to clearly show the impact of leveraging victim classification on fake account detection. Second, SybilRank was shown to outperform other contenders [8], including EigenTrust [16], SybilGuard [17], SybilLimit [18], SybilInfer [19], Mislove's method [20], and GateKeeper [21]. In what follows, we contrast these systems to both SybilRank and Íntegro.

SybilGuard [17] and SybilLimit [18] identify fake accounts based on a large number of modified random walks, where the computational cost is $O(\sqrt{m}n \log n)$ in centralized setting like OSNs. SybilInfer [19], on the other hand, uses Bayesian inference techniques to assign each user account a probability of being fake in $O(n(\log n)^2)$ time per trusted account. The system, however, does not provide analytical bounds on its ranking quality.

GateKeeper [21], which is a flow-based detection approach, improves over SumUp [62]. It relies on strong assumptions that require balanced graphs and costs $O(n \log n)$ time per trusted account, referred to as a "ticket source."

Viswanath *et al.* used Mislove's algorithm [43] to greedily expand a local community around known real accounts in oder to partition the graph into two communities representing real and fake regions [20]. This algorithm, however, costs $O(n^2)$ time and its detection can be easily evaded if the fakes establish sparse connectivity among themselves [10].

Compared to these detection systems, SybilRank provides an equivalent or tighter security bound and is more computationally efficient, as it requires $O(n \log n)$ time regardless of the number of trusted accounts. Compared to SybilRank, Íntegro provides $O(|E_a|/\text{vol}(E_a))$ improvement on its security bound, requires the same $O(n \log n)$ time, and is robust against social infiltration, unlike SybilRank and all other systems.

5.2. Datasets

We used datasets collected from multiple sources. Each dataset contained either profile pages or a social graph consisting of a specific number of users, as summarized in Table B.1. We next describe each dataset in detail.

5.2.1. User profiles

We used two datasets from two different OSNs. The first datasets, denoted by pro-Facebook in Table B.1, contained public profile pages of 9,646 real users who received friend requests from fake accounts. As the dataset was collected in early 2011, we wanted to verify whether these users are still active on Facebook. Accordingly, we revisited their public profiles in June 2013. We found that 7.9% of these accounts were either disabled by Facebook or deactivated by the users themselves. Accordingly, we excluded these accounts, ending up with 8,888 accounts, out of which 32.4% were victims who accepted a single friend request sent by a fake posing as a stranger. As fakes initially targeted users at random, the dataset included a diverse sample of Facebook users. In particular, these users were 51.3% males and 48.7% females, lived in 1,983 cities across 127 countries, practiced 43 languages, and have used Facebook for 5.4 years on average.

The second dataset, denoted by pro-Tuenti in Table B.1, contained profiles of 60K real users who received friend requests from fake accounts, out of which 50% were victims. The dataset was collected in Feb 10, 2014 from live production servers, where data resided in memory and no expensive, backend queries were made. For Tuenti, collecting this dataset was a low-cost and easy process, as it only involved reading cached user profiles of a subset of its *daily active users*, users who logged in to Tuenti on that particular day.

Research ethics. To collect the pro-Facebook dataset, we followed the known practices and obtained the approval of our university's research ethics board [6]. As for the pro-Tuenti dataset, we signed a non-disclosure agreement (NDA) with Tuenti in order to access an anonymized, aggregated version of its user data, with the whole process being mediated by Tuenti's Site Integrity team.

The ground-truth. For the pro-Facebook dataset, we used the ground-truth of the original study [6], which we also re-validated for the purpose of this work, as we described above. For the pro-Tuenti dataset, the accounts were inspected and labeled by its account analysts. The inspection included matching user profile photos to its declared age or address, understanding natural

language in user posts, examining the user's friends, and analyzing the user's IP address and HTTP-related information.

5.2.2. Social graphs

We used four datasets sampled from Facebook and arXiv,[5] the largest collaboration network for scientific research. The Facebook datasets, denoted by gra-FacebookTs and gra-FacebookRd in Table B.1, are part of the same 2011 social infiltration study [6]. This means that each node in either dataset has a corresponding user profile in the pro-Facebook dataset. The gra-FacebookTs dataset consisted of 2,926 real accounts with 9,124 friendships (the real region), 65 fakes with 2,080 friendships (the fake region), and 748 *timestamped* attack edges. The gra-FacebookRd dataset consisted of 6,136 real accounts with 38,144 friendships, which represented the real region only. Both datasets were sampled from Facebook using a stochastic version of the Breadth-First Search method, called "forest fire" [63].

The last two datasets, denoted by gra-HepTh and gra-AstroPh in Table B.1, represent collaboration networks between scientists in two research fields in Physics. These datasets have been widely used to analyze social graph properties [64] and to evaluate graph-based fake account detection algorithms, in which each dataset is used as the real region of the graph [8, 65, 20]. In particular, the gra-HepTh dataset consisted of 8,638 nodes with 24,827 edges, while the gra-AstroPh dataset consisted of 17,903 nodes with 197,031 edges. Both datasets represented the largest connected component (LCC) of each of the corresponding collaboration network.

5.3. Victim Classification

We sought to validate the following claim: An OSN can identify potential victims with a probability that is better than random, using strictly low-cost features extracted from readily-available user profiles. We note, however, that it is possible to achieve better classification performance, at the price of a higher computational cost, by using more sophisticated learning algorithms with temporal activity features [52].

5.3.1. Features

As described in Table B.2, we extracted features from the profiles datasets to generate feature vectors. The only requirement for feature selection was

[5] `http://arxiv.org/`

to have each feature value available for all users in the dataset, so that the resulting feature vectors are complete and have no missing values. For the pro-Facebook dataset, we were able to extract 18 features from public user profiles. For pro-Tuenti, however, the dataset was limited to 14 features, but contained user features that are not publicly accessible.

5.3.2. Classifier tuning

The RF learning algorithm is an *ensemble* algorithm, where a set of decision trees is constructed at training time. When evaluating the classifier on new data (i.e., unlabeled feature vectors), the decisions from all trees are combined using a majority voting aggregator [51]. Each decision tree in the forest uses a small random subset of available features in order to decrease the *generalization error*, which measures how well the classifier generalizes to unseen data [52]. As shown in Fig. B.4, we performed parameter tuning to calibrate the RF classifier. In particular, we used the out-of-bag error estimates computed by the RF algorithm to numerically find the best number of decision trees and the number of features for each tree, so that the prediction variance and bias are controlled across the trees.

5.3.3. Validation method

In order to evaluate the accuracy of the victim classifier, we performed a 10-fold, *stratified cross-validation* method [52] using the RF learning algorithm, after initial parameter tuning. First, we randomly partitioned the dataset into 10 equally-sized sets, with each set having the same percentage of victims as the complete dataset. We next trained an RF classifier using 9 sets and tested it using the remaining set. We repeated this procedure 10 times (i.e., folds), with each of the sets being used once for testing. Finally, we combined the results of the folds by computing the mean of their true-positive rate (TPR) and false-positive rate (FPR).

5.3.4. Performance metrics

The output of the classifier depends on its operating threshold, which is a cutoff value in the prediction probability after which the classifier identifies a user as a potential victim. In order to capture the trade-off between TPR and FPR in single curve, we repeated the cross-validation method under different threshold values using a procedure known as *receiver operating characteristics* (ROC) analysis. In ROC analysis, the closer the curve is to the top-left corner at point $(0, 1)$ the better the classification performance is. The quality of the

classifier can be quantified with a single value by calculating the *area under its ROC curve* (AUC) [52].

We also recorded the *relative importance* (RI) of features used for the classification. The RI score is computed by the RF algorithm, and it describes the relative contribution of each feature to the predictability of the label (i.e., a victim or a non-victim), when compared to all other features [51].

5.3.5. Results

For both datasets, the victim classifier ended up with an AUC greater than 0.5, as depicted in Fig. B.5a. In particular, for the pro-Facebook dataset, the classifier delivered an AUC of 0.7, which is 40% better than random. For the pro-Tuenti dataset, on the other hand, the classifier delivered an AUC of 0.76, which is 52% better than random. Also, increasing the dataset size to more than 40K feature vectors did not significantly improve the AUC during cross-validation, as shown in Fig. B.5b. This means that an OSN operator can train a victim classifier using a relatively small dataset, so fewer accounts need to be manually verified.

We also experimented with another two widely-used learning algorithms for victim classification, namely, Naïve Bayes (NB) and SVM [52]. However, both of these algorithms resulted in smaller AUCs on both datasets. In particular, for the pro-Facebook dataset, the NB classifier achieved an AUC of 0.63 and the SVM classifier achieved an AUC of 0.57. Similarly, for the pro-Tuenti dataset, the NB classifier achieved an AUC of 0.64 and the SVM classifier achieved an AUC of 0.59. This result is not surprising, as ensemble learning algorithms, such as the RF algorithm, achieve better predictive performance in case individual classifiers are "weak," meaning they have small AUCs but are still better than random [52].

5.4. Ranking Quality

We compared Íntegro against SybilRank in terms of their ranking quality under various attack scenarios, where ideally real accounts should be ranked higher than fake accounts. Our results are based on the average of at least 10 runs, with error bars reporting 95% confidence intervals (CI), when applicable. We used the Facebook datasets, namely pro-Facebook, gra-FacebookTs, and gra-FacebookRd, for this comparison because they included feature vectors and graph samples. We also supplement the evaluation using the arXiv datasets, namely gra-HepTh and gra-AstroPh, with synthetic feature vectors.

5.4.1. Infiltration scenarios

We considered two real-world attack scenarios that have been shown to be successful in practice. In the first scenario, attackers establish attack edges by targeting users with whom their fakes have mutual friends [6]. Accordingly, we used the gra-FacebookTs dataset which contained timestamped attack edges, allowing us to replay the infiltration by 65 socialbots (n=2,991 and m=11,952). We refer to this scenario as the *targeted-victim* attack.

In the second scenario, attackers establish attack edges by targeting users at random [8]. We designated the gra-FacebookRd dataset as the real region. We then generated a synthetic fake region consisting of 3,068 fakes with 36,816 friendships using the small-world graph model [66]. We then added 35,306 random attack edges between the two regions (n=9,204 and m=110,266). As suggested by related work [36], we used a relatively large number of fakes and attack edges in order to stress-test both systems under evaluation. We refer to the this scenario as the *random-victim* attack.

We supplemented the evaluation under the random-victim attack using the arXiv datasets. In particular, each dataset represented the real region of the graph. We generated synthetic fake regions using the same small-world graph model, where each fake region consisted of 5K fakes and 20K friendships. For each graph, the real and fake regions were connected through 2K random attack edges. This resulted in two graphs, where the first consisted of $n = 13,638$ nodes and $m = 46,827$ edges using the ca-HepTh dataset, and the second graph consisted of $n = 22,903$ nodes and $m = 219,031$ edges using the ca-AstroPh dataset.

5.4.2. Edge weights

For each infiltration scenario, we deployed the previously trained victim classifier in order to assign new edge weights. As we injected fakes in the second scenario, we generated their feature vectors by sampling each feature distribution of fakes from the first scenario.[6] For the supplementary datasets, we used the same procedure to weight the edges using synthetic feature vectors and the trained victim classifier. We also assigned edge weights using another victim classifier that simulates two operational modes. In the first mode, the classifier outputs the *best* possible victim predictions with an AUC$\approx$1 and probabilities greater than 0.95. In the second mode, the classifier outputs

[6]We excluded the "friends" feature, as it can be computed from the graph.

uniformly *random* predictions with an AUC≈0.5. We used this classifier to evaluate the theoretical best and practical worst case performance of Íntegro, as specified by the legend of Fig. B.6.

5.4.3. Evaluation method

In order to evaluate each system's ranking quality, we ran the system using both infiltration scenarios starting with a single attack edge. We then added another attack edge, according to its timestamp if available, and repeated the experiment. We kept performing this process until there were no more edges to add. At the end of each run, we measured the ranking AUC resulted by each system, as explained next.

5.4.4. Performance metric

For the resulting ranked list of accounts, we performed ROC analysis by moving a pivot point along the list, starting from the bottom. If an account is behind the pivot, we marked it as fake; otherwise, we marked it as real. Given the ground-truth, we measured the TPR and the FPR across the whole list. Finally, we computed the corresponding AUC, which in this case *quantifies the probability that a random real account is ranked higher than a random fake account.*

5.4.5. Seeds and iterations

In order to make the chance of guessing seeds very small, we picked 100 trusted accounts that are non-victim, real accounts. We used a total trust that is equal to n, the number of nodes in the given graph. We also performed $\lceil \log_2(n) \rceil$ iterations for both Íntegro and SybilRank.

5.4.6. Results

Íntegro consistently outperformed SybilRank in ranking quality, especially as the number of attack edges increased. Using the RF classifier, Íntegro resulted in an AUC which is always greater than 0.92, and is up to 30% improvement over SybilRank in each attack scenario, as shown in Fig B.6.

In each infiltration scenario, both systems performed well when the number of attack edges was relatively small. In other words, the fakes were sparsely connected to real accounts and so the regions were easily separated. As SybilRank limits the number of fakes that can outrank real accounts by the number of attack edges, its AUC degraded significantly as more attack

edges were added to each graph. Íntegro, however, maintained its performance, with at most 0.07 decrease in AUC, even when the number of attack edges was relatively large. Notice that Íntegro performed nearly as good as SybilRank when a random victim classifier was used, but performed much better when the RF classifier was used instead. This clearly shows the impact of leveraging victim classification on fake account detection.

5.5. Sensitivity to Seed-targeting Attacks

Sophisticated attackers might obtain a full or partial knowledge of which accounts are trusted by the OSN operator. As the total trust is initially distributed among these accounts, an attacker can adversely improve the ranking of the fakes by establishing attack edges directly with them. We next evaluate both systems under two variants of this seed-targeting attack.

5.5.1. Attack scenarios

We focus on two main attack scenarios. In the first scenario, the attacker targets accounts that are *k nodes away from all trusted accounts*. This means that the length of the shortest path from any fake account to any trusted account is exactly k+1, representing the *distance* between the seeds and the fake region. For k=0, each trusted account is a victim and located at a distance of 1. We refer to this scenario, which assumes a resourceful attacker, as the *distant-seed* attack.

In the second scenario, attackers have only a partial knowledge and target k trusted accounts picked at random. We refer to this scenario as the *random-seed* attack.

5.5.2. Evaluation method

In order to evaluate the sensitivity of each system to a seed-targeting attack, we used the gra-FacebookTs dataset to simulate each attack scenario. We implemented this by replacing the endpoint of each attack edge in the real region with a real account picked at random from a set of candidates. For the first scenario, a candidate account is one that is k nodes away from all trusted accounts. For the second scenario, a candidate account is simply any trusted account. We ran experiments for both systems using different values of k and measured the corresponding AUC at the end of each run.

5.5.3. Results

In the first attack scenario, both systems had a poor ranking quality when the distance was small, as illustrated in Fig. B.7a. Because Íntegro assigns low weights to edges incident to victim accounts, the trust that escapes to the fake region is less likely to come back into the real region. This explains why SybilRank had a slightly better AUC for distances less than 3. However, once the distance was larger, Íntegro outperformed SybilRank ,as expected from earlier results.

In the second attack scenario, the ranking quality of both systems degraded, as the number of victimized trusted accounts increased, where Íntegro consistently outperformed SybilRank, as shown in Fig. B.7b. Notice that by selecting a larger number of trusted accounts, it becomes much harder for an attacker to guess which account is trusted, while the gained benefit per victimized trusted account is further reduced.

5.6. Deployment at Tuenti

We deployed both systems on a snapshot of Tuenti's daily active users graph in February 6, 2014. The graph consisted of several million nodes and tens of millions of edges. We had to mask out the exact numbers due to a non-disclosure agreement with Tuenti. After initial analysis of the graph, we found that 96.6% of nodes and 94.2% of edges belonged to one *giant connected component* (GCC). Thus, we focused our evaluation on this GCC.

5.6.1. Preprocessing

Using a uniform random sample of 10K users, we found that new users have weak connectivity to others due to the short time they have been on Tuenti, as shown in Fig. B.8a. If these users were included in our evaluation, they would end up receiving low ranks, which would lead to false positives.

To overcome this hurdle, we estimated the period after which users accumulate at least 10% of the average number of friends in Tuenti. To achieve this, we used a uniformly random sample of 10K real users who joined Tuenti over the last 77 months. We divided the users in the sample into buckets representing how long they have been active members. We then calculated the average number of new friendships they made after every other month. As illustrated in Fig. B.8b, users accumulated 53% of their friendships during the first 12 months. In addition, 18.6% of friendships were made after one month since joining the network. To this end, we decided to defer the

consideration of users who have joined in the last 30 days since Feb 6, 2014, which represented only 1.3% of users in the GCC.

5.6.2. Community detection

We applied the Louvain method on the preprocessed GCC. The method finished quickly after just 5 iterations with a high modularity score of 0.83, where a value of 1 corresponds to a perfect partitioning. In total, we found 42 communities and the largest one consisted of 220,846 nodes. In addition, 15 communities were relatively large containing more than 50K nodes. Tuenti's account analysts verified 0.05% of the nodes in each detected community, and designated these nodes as trusted accounts for both systems.

5.6.3. Performance metric

As the number of users in the processed GCC is large, it was infeasible to manually inspect and label each account. This means that we were unable to evaluate the system using ROC analysis. Instead, we attempted to determine the percentage of fake accounts at equally-sized intervals in the ranked list. We accomplished this in collaboration with Tuenti's analysts by manually inspecting a user sample in each interval in the list. This percentage is directly related to the *precision* of fake account detection, which is a performance metric typically used to measure the ratio of relevant items over the top-k highest ranked items in terms of relevance [67].

5.6.4. Evaluation method

We utilized the previously trained victim classifier in order to weight a copy of the graph. We then ran both systems on two versions of the graph (i.e., weighted and unweighted) for $\lceil \log_2(n) \rceil$ iterations, where n is number of nodes in the graph. After that, we examined the ranked list of each system by inspecting the first lowest-ranked one million users. We decided not to include the complete range due to confidentiality reasons, because otherwise one could precisely estimate the actual number of fakes in Tuenti. We randomly selected 100 users out of each 20K user interval for inspection in order to measure the percentage of fakes in the interval, that is, the precision.

5.6.5. Results

As shown in Fig. B.9a, Íntegro resulted in 95% precision in the lowest 20K ranking user accounts, as opposed to 43% by SybilRank and 5% by Tuenti's user-based abuse reporting system. This precision dropped dramatically as

we went up in the list, which means our ranking scheme placed most of the fakes at the bottom of the ranked list, as shown in Fig. B.9b.

Let us consider SybilRank's ranking shown in Fig. B.9a and Fig. B.9c. The precision, starting with 43% for the first interval, gradually decreased until rising again at the 10th interval. This pattern repeated at the 32nd interval as well. We inspected the fake accounts at these intervals and found that they belonged to three different, large communities. In addition, these fakes had a large number of friends, much larger than the average of 254 friends. In particular, the fakes from the 32nd interval onwards had more than 300 friends, with a maximum of up to 539. Fig. B.9d shows the degree distribution for both *verified* fake and real accounts. This figure suggests that fakes tend to create many attack edges with real accounts, which confirms earlier findings on other OSNs such as Facebook [6]. Also, this behavior explains why Íntegro outperformed SybilRank in user ranking quality; these high degree fakes received lower ranks as most of their victims were identified by the classifier.

5.6.6. SybilRank in retrospect

SybilRank was initially evaluated on Tuenti, where it effectively detected a significant percentage of the fakes [8]. The original evaluation, however, pruned excessive edges of nodes that had a degree greater than 800, which include a non-disclosed number of fakes that highly infiltrated Tuenti. Also, the original evaluation was performed on the whole graph, which included many dormant accounts. In contrast, our evaluation was based on the daily active users graph in order to focus on active fake accounts that could be harmful. While this change limited the number of fakes that existed in the graph, it has evidently revealed the ineffectiveness of SybilRank under social infiltration. Additionally, the original evaluation showed that 10–20% of fakes received high ranks, a result we also attest, due to the fact that these fake accounts had established many attack edges. On the other hand, Íntegro has 0–2% of fakes at these high intervals, and so it delivers an order of magnitude better precision than SybilRank.

6. Implementation and Scalability

We implemented Íntegro in Mahout and Giraph, which are widely used, open-source distributed machine learning and graph processing platforms. We next test the scalability of Íntegro using a synthetic benchmark.

6.0.1. Benchmark

We deployed Íntegro on an Amazon Elastic MapReduce[7] cluster. The cluster consisted of a single *m1.small* instance serving as a master node and 32 *m2.4xlarge* instances serving as slave nodes. We also employed the small-world graph model [66] to generate 5 graphs with an exponentially increasing number of nodes. For each one of these graphs, we used the Facebook dataset to randomly generate all feature vectors with the same distribution for each feature. We then ran Íntegro on each of the generated graphs and measured its execution time.

6.0.2. Results

Íntegro achieved a nearly linear scalability with the number of nodes in a graph, as illustrated in Fig. B.10. Excluding the time required to load the 160M node graph into memory, 20 minutes for a non-optimized data format, it takes less than 2 minutes to train an RF classifier and compute vulnerability scores for nodes, and less than 25 minutes to weight the graph, rank nodes, and finally sort them. This makes Íntegro computationally practical even for large OSNs such as Facebook.

7. Discussion

As mentioned in Section 4.6, Íntegro's security guarantee is sensitive to the performance of the deployed victim classifier, which is formally captured by the volume $\mathrm{vol}(E_a)$ in the bound $O(\mathrm{vol}(E_a) \log n)$, and can be practically measured by its AUC.

7.1. Sensitivity to Victim Classification and Social Infiltration

As illustrated in Fig. B.6, improving the AUC of the victim classifier from random with AUC ≈ 0.5, to actual with AUC$= 0.7$, and finally to best with AUC ≈ 1 consistently improved the resulting ranking in terms of its AUC. Therefore, a higher AUC in victim classification leads to a higher AUC in user ranking. This is the case because the ROC curve of a victim classifier monotonically increases, so a higher AUC implies a higher true positive rate (TPR). In turn, a higher TPR means more victims are correctly identified, and so more attack edges are assigned lower weights, which evidently leads to a higher AUC in user ranking.

[7] `http://aws.amazon.com/elasticmapreduce`

Regardless of the used victim classifier, the ranking quality decreases as the number of attack edges increases, as illustrated in Fig. B.6. This is the case because even a small false negative rate (FNR) in victim classification means more attack edges, which are indecent to misclassified victims, are assigned high weights, leading to a lower AUC in user ranking.

7.2. Maintenance and Impact

While an attacker does not control real accounts nor their activities, it can still trick users into befriending fakes. In order to achieve a high-quality ranking, the victim classifier should be regularly retrained to capture new and changing user behavior in terms of susceptibility to social infiltration. This is, in fact, the case for supervised machine learning when applied to computer security problems [9]. Also, as the ranking scheme is sensitive to seed-targeting attacks, the set of trusted accounts should be regularly updated and validated in order to reduce the negative impact of these attacks, even if they are unlikely to succeed, as discussed in Section 4.4.

By using Íntegro, Tuenti requires nearly 67 man hours to manually validate the 20K lowest ranking user accounts, and discover about 19K fake accounts instead of 8.6K fakes with SybilRank. With its user-based abuse reporting system that has 5% hit rate, and assuming all fakes get reported, Tuenti would need 1,267 man hours instead to discover 19K fake accounts. This improvement has been useful to Tuenti and its users.

7.3. Resilience to Adversarial Attacks

There are several design factors that make Íntegro inherently more resilient to adversarial attacks compared to techniques which only rely on similarly-trained fake account classifiers.

First, Íntegro leverages victim classification for the detection of fake accounts. Since attackers do not control victims, this makes Íntegro more robust to adversarial attack strategies such as the "boiling-frog" attack [33] which can force fake account classifiers to tolerate abusive activities by slowly introducing such activities to the OSN. One potential approach for attackers to control victim accounts is through account hijacking, however that is impractical on a large scale and can be avoided by verifying the accounts as described in Section 2.5.

Second, the seed selection strategy employed by Íntegro (as outlined in Section 4.4.1) is designed to reduce the negative impact of seed targeting attacks. Given that the total trust is initially distributed among trusted

seed accounts, in such a scenario, an attacker may attempt to direct fake accounts to befriend trusted accounts in order to adversely improve the fake accounts' ranking. However, Íntegro selects the seeds at random thus a potential attacker is forced to guess the seeds from among a large number of users. Additionally, Íntegro employs multiple seeds which further reduces the probability of an attacker correctly guessing the trusted seeds. In practice, the number of seeds is only limited by the available resources necessary for manual account verification of the trusted seeds. We analyze such an attack scenario in Section 5.5 and we demonstrate that Íntegro typically outperforms SybilRank when subjected to a seed targeting attack.

8. Limitations

We next outline two design limitations which are inherited from SybilRank [8] and similar random walk-based ranking schemes [36]:

- *Íntegro's design is limited to only undirected social graphs.* In other words, OSNs whose users declare lateral relationships, such as Twitter and Google+, are not expected to benefit from our proposal. This is the case because directed graphs, in general, have a significantly smaller mixing time than their undirected counterparts [65], which means that a random walk on such graphs will converge in a much small number of steps, rendering short random walks unsuitable for robust user ranking.

- *Íntegro delays the consideration of new user accounts.* This means that an OSN operator might miss the chance to detect fakes at their early life-cycle. However, as shown in Figure B.8a, only 7% of new users who joined Tuenti in the last month had more than 46 friends. To estimate the number of fakes in new accounts, we picked 100 new accounts at random for manual verification. We found that only 6% of these accounts were fake, and the most successful fake account had 103 victims. In practice, the decision of whether to exclude these accounts is operational, and it depends on the actions taken on low-ranking users. For example, an operator can enforce abuse mitigation technique, as discussed in Section 2.5, against low-ranking users, where false positives can negatively affect user experience but slow down fake accounts that just joined the network. This is a security/usability trade-off which we leave to the operator to manage. Alternatively, the operator can use

fake account detection systems that are designed to admit legitimate new users using, for example, a vouching process [68].

Íntegro is not a stand-alone fake account detection system. It is intended to complement existing detection systems and is designed to detect automated fake accounts that befriend many victims for subsequent attacks. Íntegro is deployed at Tuenti along side a feature-based detection system and a user-based abuse reporting system.

9. Conclusion

OSNs today are faced with the problem of detecting fake accounts in a highly adversarial environment. The problem is becoming more challenging as such accounts have become sophisticated in cloaking their operation with patterns resembling real user behavior. In this paper, we presented Íntegro, a scalable defense system that helps OSN operators to detect fake accounts using a meaningful user ranking scheme.

Our evaluation results show that SybilRank, the state-of-the-art in fake account detection, is ineffective when the fakes infiltrate the target OSN by befriending a large number of real users. Íntegro, however, has proven more resilient to this effect by leveraging in a novel way the knowledge of benign victim accounts that befriend fakes. We have implemented Íntegro on top of standard data processing platforms, Mahout and Giraph, which are scalable and easy to deploy in modern data centers. In fact, Tuenti, the largest OSN in Spain with more than 15M active users, has deployed our system in production to thwart fakes in the wild with at least 10 time more precision.

Acknowledgement

We would like to thank Gianluca Stringhini and our colleagues at LERSSE and NetSysLab for their help and feedback on an earlier version of this article. The first author is thankful to the University of British Columbia for a generous doctoral fellowship.

References

[1] J. R. Douceur, The sybil attack, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag, 2002, pp. 251–260.

[2] Facebook, Quarterly earning reports (Jan 2014).
URL http://goo.gl/YujtO

[3] CBC, Facebook shares drop on news of fake accounts (Aug 2012).
URL http://goo.gl/6s5FKL

[4] K. Thomas, C. Grier, D. Song, V. Paxson, Suspended accounts in retrospect: an analysis of Twitter spam, in: Proceedings of the 2011 ACM SIGCOMM Internet Measurement Conference, ACM, 2011, pp. 243–258.

[5] G. Yan, G. Chen, S. Eidenbenz, N. Li, Malware propagation in online social networks: nature, dynamics, and defense implications, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, 2011, pp. 196–206.

[6] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, The socialbot network: when bots socialize for fame and money, in: Proceedings of the 27th Annual Computer Security Applications Conference, ACM, 2011, pp. 93–102.

[7] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, F. Menczer, Truthy: mapping the spread of astroturf in microblog streams, in: Proceedings of the 20th International Conference Companion on World Wide Web, ACM, 2011, pp. 249–252.

[8] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: USENIX conference on Networked Systems Design and Implementation, USENIX Association, 2012, pp. 15–15.

[9] T. Stein, E. Chen, K. Mangla, Facebook immune system, in: Proceedings of the 4th Workshop on Social Network Systems, ACM, 2011, pp. 8–14.

[10] L. Alvisi, A. Clement, A. Epasto, U. Sapienza, S. Lattanzi, A. Panconesi, SoK: The evolution of sybil defense via social networks, In Proceedings of the IEEE Symposium on Security and Privacy.

[11] L. Bilge, T. Strufe, D. Balzarotti, E. 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, ACM, 2009, pp. 551–560.

[12] C. Wagner, S. Mitter, C. Körner, M. Strohmaier, When social bots attack: Modeling susceptibility of users in online social networks, in: Proceedings of the WWW, Vol. 12, 2012.

[13] M. N. Ko, G. P. Cheek, M. Shehab, R. Sandhu, Social-networks connect services, Computer 43 (8) (2010) 37–43.

[14] S. Yardi, N. Feamster, A. Bruckman, Photo-based authentication using social networks, in: Proceedings of the first workshop on Online social networks, ACM, 2008, pp. 55–60.

[15] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, Design and analysis of a social botnet, Computer Networks 57 (2) (2013) 556–578.

[16] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The EigenTrust algorithm for reputation management in P2P networks, in: Proceedings of the 12th international conference on World Wide Web, ACM, 2003, pp. 640–651.

[17] H. Yu, M. Kaminsky, P. B. Gibbons, A. Flaxman, Sybilguard: defending against sybil attacks via social networks, ACM SIGCOMM Computer Communication Review 36 (4) (2006) 267–278.

[18] H. Yu, P. B. Gibbons, M. Kaminsky, F. Xiao, Sybillimit: A near-optimal social network defense against sybil attacks, in: Proceedings of IEEE Symposium on Security and Privacy, IEEE, 2008, pp. 3–17.

[19] G. Danezis, P. Mittal, Sybilinfer: Detecting sybil nodes using social networks., in: Proceedings of the 9th Annual Network & Distributed System Security Symposium, ACM, 2009.

[20] B. Viswanath, A. Post, K. P. Gummadi, A. Mislove, An analysis of social network-based sybil defenses, in: Proceedings of ACM SIGCOMM Computer Communication Review, ACM, 2010, pp. 363–374.

[21] N. Tran, J. Li, L. Subramanian, S. S. Chow, Optimal sybil-resilient node admission control, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 3218–3226.

[22] T. Hwang, I. Pearce, M. Nanis, Socialbots: Voices from the fronts, interactions 19 (2) (2012) 38–45.

[23] M. Egele, G. Stringhini, C. Kruegel, G. Vigna, Compa: Detecting compromised accounts on social networks., in: NDSS, 2013.

[24] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, G. M. Voelker, Dirty jobs: The role of freelance labor in web service abuse, in: Proceedings of the 20th USENIX Security Symposium, USENIX Association, 2011, pp. 14–14.

[25] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, Y. Dai, Uncovering social network sybils in the wild, in: Proceedings of the 2011 ACM SIGCOMM Internet Measurement Csonference, ACM, 2011, pp. 259–268.

[26] G. Stringhini, C. Kruegel, G. Vigna, Detecting spammers on social networks, in: Proceedings of the 26th Annual Computer Security Applications Conference, ACM, 2010, pp. 1–9.

[27] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, B. Y. Zhao, You are how you click: Clickstream analysis for sybil detection, in: Proceedings of the 22nd USENIX Security Symposium, USENIX Association, 2013, pp. 1–8.

[28] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed computing 48 (1) (1998) 96–129.

[29] M. Fire, D. Kagan, A. Elyashar, Y. Elovici, Friend or foe? fake profile identification in online social networks, Social Network Analysis and Mining 4 (1) (2014) 1–23.

[30] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, A. Mislove, Towards detecting anomalous user behavior in online social networks, in: 23rd USENIX Security Symposium (USENIX Security 14), USENIX Association, San Diego, CA, 2014, pp. 223–238.

[31] Q. Cao, X. Yang, J. Yu, C. Palow, Uncovering large groups of active malicious accounts in online social networks, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, ACM, New York, NY, USA, 2014, pp. 477–488.

[32] G. Stringhini, P. Mourlanne, G. Jacob, M. Egele, C. Kruegel, G. Vigna, Evilcohort: Detecting communities of malicious accounts on online services, in: 24th USENIX Security Symposium (USENIX Security 15), USENIX Association, Washington, D.C., 2015, pp. 563–578.

[33] J. Tygar, Adversarial machine learning., IEEE Internet Computing 15 (5).

[34] D. Lowd, C. Meek, Adversarial learning, in: Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining, ACM, 2005, pp. 641–647.

[35] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, Key challenges in defending against malicious socialbots, in: Proceedings of the 5th USENIX Workshop on Large-scale Exploits and Emergent Threats, Vol. 12, 2012.

[36] H. Yu, Sybil defenses via social networks: a tutorial and survey, ACM SIGACT News 42 (3) (2011) 80–101.

[37] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, A. Post, Exploring the design space of social network-based sybil defenses, in: In Proceedings of the 4th International Conference on Communication Systems and Networks, IEEE, 2012, pp. 1–8.

[38] Y. Boshmaf, K. Beznosov, M. Ripeanu, Graph-based sybil detection in social and information systems, in: Proceedings of 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, IEEE, 2013.

[39] M. Fire, G. Katz, Y. Elovici, Strangers intrusion detection-detecting spammers and fake proles in social networks based on topology anomalies, HUMAN 1 (1) (2012) pp–14.

[40] Q. Cao, M. Sirivianos, X. Yang, K. Munagala, Combating friend spam using social rejections, in: Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on, 2015, pp. 235–244.

[41] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney, Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, Internet Mathematics 6 (1) (2009) 29–123.

[42] S. Fortunato, Community detection in graphs, Physics Reports 486 (3) (2010) 75–174.

[43] A. Mislove, B. Viswanath, K. P. Gummadi, P. Druschel, You are who you know: inferring user profiles in online social networks, in: Proceedings of the third ACM international conference on Web search and data mining, ACM, 2010, pp. 251–260.

[44] Q. Cao, X. Yang, Sybilfence: Improving social-graph-based sybil defenses with user negative feedback, Tech. Rep. Duke CS-TR-2012-05, Department of Computer Science, Duke University (March 2012).

[45] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, K. P. Gummadi, Understanding and combating link farming in the twitter social network, in: Proceedings of the 21st international conference on World Wide Web, ACM, 2012, pp. 61–70.

[46] A. Elyashar, M. Fire, D. Kagan, Y. Elovici, Homing socialbots: intrusion on a specific organization's employee using socialbots, in: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ACM, 2013, pp. 1358–1365.

[47] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, B. Y. Zhao, Follow the green: growth and dynamics in twitter follower markets, in: Proceedings of the 2013 conference on Internet measurement conference, ACM, 2013, pp. 163–176.

[48] C. Yang, R. Harkreader, J. Zhang, S. Shin, G. Gu, Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter, in: Proceedings of the 21st international conference on World Wide Web, ACM, 2012, pp. 71–80.

[49] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, B. Y. Zhao, Social turing tests: Crowdsourcing sybil detection, in: Proceedings of the 20th Annual Network & Distributed System Security Symposium, ACM, 2013.

[50] D. A. Spielman, S.-H. Teng, Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, in: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, ACM, 2004, pp. 81–90.

[51] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[52] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: Data mining, inference, and prediction, second edition, Springer, 2009.

[53] Z. Gyöngyi, H. Garcia-Molina, J. Pedersen, Combating web spam with trustrank, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 576–587.

[54] G. H. Golub, H. A. Van der Vorst, Eigenvalue computation in the 20th century, Journal of Computational and Applied Mathematics 123 (1) (2000) 35–65.

[55] E. Behrends, Introduction to Markov chains with special emphasis on rapid mixing, Vol. 228, Vieweg, 2000.

[56] M. Dellamico, Y. Roudier, A measurement of mixing time in social networks, in: Proceedings of the 5th International Workshop on Security and Trust Management, Saint Malo, France, 2009.

[57] A. Mohaisen, A. Yun, Y. Kim, Measuring the mixing time of social graphs, in: Proceedings of the 10th annual conference on Internet measurement, ACM, 2010, pp. 383–389.

[58] J. Leskovec, K. J. Lang, A. Dasgupta, M. W. Mahoney, Statistical properties of community structure in large social and information networks, in: Proceedings of the 17th international conference on World Wide Web, ACM, 2008, pp. 695–704.

[59] V. Blondel, J. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10).

[60] M. E. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences 103 (23) (2006) 8577–8582.

[61] A. Sinclair, Improved bounds for mixing rates of Markov chains and multicommodity flow, in: Proceedings of the 1st Latin American Symposium on Theoretical Informatics, Springer-Verlag, 1992, pp. 474–487.

[62] D. N. Tran, B. Min, J. Li, L. Subramanian, Sybil-resilient online content voting., in: NSDI, Vol. 9, 2009, pp. 15–28.

[63] J. Leskovec, C. Faloutsos, Sampling from large graphs, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 631–636.

[64] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Transactions on Knowledge Discovery from Data (TKDD) 1 (1) (2007) 2.

[65] A. Mohaisen, H. Tran, N. Hopper, Y. Kim, On the mixing time of directed social graphs and security implications, in: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ACM, 2012, pp. 36–37.

[66] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks, nature 393 (6684) (1998) 440–442.

[67] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 5–53.

[68] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, Z. M. Mao, Innocent by association: early recognition of legitimate users, in: Proceedings of the 2012 ACM conference on Computer and communications security, ACM, 2012, pp. 353–364.

In what follows, we provide the required background on random walks after which we analyze the main security guarantee of Íntegro.

Appendix A. Background

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ undirected edges. Also, let $w : E \to \mathbb{R}^+$ be a function that assigns each edge $(v_i, v_j) \in E$ a weight $w(v_i, v_j) > 0$. The transition matrix P is an $n \times n$ matrix, where each entry $p_{ij} \in [0, 1]$ represents the probability of moving from node $v_i \in V$ to node $v_j \in V$, as defined by:

$$p_{ij} := \begin{cases} \dfrac{w(v_i, v_j)}{\deg(v_i)} & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{A.1}$$

The transition matrix P might not be symmetric but it is right-stochastic, as P is a square matrix of non-negative real numbers and for each node $v_i \in V$,

$$\sum_{(v_i,v_j)\in E} p_{ij} = 1. \tag{A.2}$$

The event of moving from one node to another in G is captured by a Markov chain representing a random walk over G. In turn, a random walk $W = \langle v_1, \ldots, v_i \rangle$ of length $i \geq 1$ over G is a sequence of nodes that starts at the initial node v_1 and ends at the terminal node v_i, following the transition probability defined in Equation A.1. The Markov chain is called ergodic if it is irreducable and aperiodic. In this case, the Markov chain has a unique stationary distribution to which the random walk converges as $i \to \infty$. The stationary distribution π of a Markov chain is a probability distribution that is invariant to the transition matrix, that is, whenever $\pi P = \pi$. The stationary distribution of the Markov chain over G is a $1 \times n$ probability vector, and is defined by

$$\pi := \left[\frac{\deg(v_1)}{\mathrm{vol}(V)} \quad \cdots \quad \frac{\deg(v_n)}{\mathrm{vol}(V)} \right], \tag{A.3}$$

where $\pi(v_j)$ is the jth entry in π and represents the landing probability of node $v_j \in V$, and $\mathrm{vol}(U)$ is the volume of a node set $U \subseteq V$, which is defined by

$$\mathrm{vol}(U) := \sum_{v_j \in U} \deg(v_j). \tag{A.4}$$

The marginal distribution π_i of the Markov chain over G is a $1 \times n$ probability vector, where $\pi_i(v_j)$ is the landing probability of node $v_j \in V$ at step i of the random walk. Given an initial distribution π_0, the marginal distribution π_i is iteratively defined by

$$\pi_i := \pi_{i-1} P = \pi_0 P^i, \tag{A.5}$$

and accordingly, $\pi_i(v_j)$ can be computed by [54]

$$\pi_i(v_j) = \sum_{(v_k,v_j)\in E} \pi_{i-1}(v_k) \cdot \frac{w(v_k, v_j)}{\deg(v_k)}. \tag{A.6}$$

The total variation distance $||\pi_i - \pi||_{\mathrm{TV}}$ between the marginal and stationary distributions is a measure of how "close" these distribution are, and

is defined by

$$||\pi_i - \pi||_{\text{TV}} := \frac{1}{2} \sum_{v_j \in V} |\pi_i(v_j) - \pi(v_j)|. \tag{A.7}$$

The mixing time $\mathcal{T}(\epsilon)$ of the Markov chain over G, when parametrized by a relative variation error $\epsilon > 0$, is the minimal length of the random walk required for the marginal distribution to be ϵ-close to the stationary distribution in total variation distance, and is defined by

$$\mathcal{T}(\epsilon) := \min \{i : ||\pi_i - \pi||_{\text{TV}} \leq \epsilon\}. \tag{A.8}$$

It thus follows that if $i \geq \mathcal{T}(\epsilon)$, we have $\pi_i = \pi$.

Appendix B. Analysis

We next start by proving that reassigning edge weights in an undirected graph changes its mixing time by only a constant factor. We subscript the used notation in order to differentiate between different graphs when necessary. For a given OSN, *we refer to its social graph after rate adjustment as the defense graph D.*

Lemma 1. *Given a social graph G with a mixing time $\mathcal{T}_G(\epsilon)$, the corresponding defense graph D after rate adjustment has a mixing time $\mathcal{T}_D(\epsilon) = O(\mathcal{T}_G(\epsilon))$.*

Proof. Recall that the mixing time of an undirected graph $G = (V, E)$ is bounded by [61]

$$\frac{\lambda}{2(1-\lambda)} \log\left(\frac{1}{2\epsilon}\right) \leq \mathcal{T}(\epsilon) \leq \frac{\log(n) + \log\left(\frac{1}{\epsilon}\right)}{1-\lambda}, \tag{B.1}$$

where $\lambda \in (-1, 1)$ is the second largest eigenvalue of the transition matrix P of the graph G. For a social graph G and its defense graph D, we have

$$\frac{\mathcal{T}_D(\epsilon)}{\mathcal{T}_G(\epsilon)} \leq \frac{1-\lambda_G}{1-\lambda_D} = O(1),$$

and thus, $\mathcal{T}_D(\epsilon) = O(\mathcal{T}_G(\epsilon))$. □

Given the bound in Equation B.1, a Markov chain over a graph G is fast mixing if $\mathcal{T}(\epsilon)$ is polynomial in $\log n$ and $\log(1/\epsilon)$. In the context of fake account detection, we consider the stricter case when $\epsilon = O(1/n)$, and so we call G fast mixing if $\mathcal{T}(\epsilon) = O(\log n)$.

Let us refer to the landing probability $\pi_i(v_j)$ of a node $v_j \in V$ as its trust value so that π_i is the trust distribution in step i.[8] Moreover, let the expansion $\chi(U)$ of a node set $U \subseteq V$ be defined by

$$\chi(U) := \frac{\mathrm{vol}\left(\partial(U)\right)}{\mathrm{vol}(U)}, \tag{B.2}$$

where $\partial(U) = \{(v_i, v_j) \in E : v_i \in U,\ v_j \in V \setminus U\}$ is the edge boundary of U. In our threat model, we have $\partial(V_r) = \partial(V_f) = E_a$, where edges in E_a are established at random.

We now prove that during a random walk on the defense graph $D = (V, E)$, where the walk starts from a known real node in the real region, the expected aggregate trust in the fake region D_f monotonically increases by diminishing increments until it converges to its stationary value.

Lemma 2. *Given a defense graph D with $g = |E_a|$ randomly established attack edges, $n_0 \geq 1$ trusted real nodes, a total trust $\tau \geq 1$, and an initial trust distribution*

$$\pi_0(v_j) = \begin{cases} \tau/n_0 & \text{if } v_j \text{ is a trusted node,} \\ 0 & \text{otherwise,} \end{cases}$$

the expected aggregate trust over the fake region in the $(i+1)$-th iterations increases by an amount of $\left(\chi(V_r) \cdot \tau\right)\left(1 - \chi(V_r) - \chi(V_f)\right)^i$ for each $i \geq 0$.

Proof. We prove the lemma by induction. We use the iterative method described in Equation A.5 to compute trust distribution for a random walk that starts from a trusted node. We first define some notation. Let $\pi_i(V_r)$ be the aggregate trust in the real region D_r after iteration i, as defined by

$$\pi_i(V_r) := \sum_{v_j \in V_r} \pi_i(v_j). \tag{B.3}$$

[8] In the main text, we denoted the trust value π_i by T_i. The reason we use π_i herein is because it is the standard notation used in analyzing stochastic processes [55].

Similarly, let $\pi_i(V_f)$ be the aggregate trust in D_f after iteration i. As defined by π_0, initially, we have $\pi_0(V_r) = \tau$ and $\pi_0(V_f) = 0$. Moreover, the total trust τ is reserved during the iterations, that is, $\pi_i(V_r) + \pi_i(V_f) = \tau$ for each $i \geq 0$.

In each iteration i, the total trust is redistributed in the graph. Consider iteration $i+1$. For each $v_i, v_j \in V_r$, the edge $(v_i, v_j) \in E$ carries $w(v_i, v_j)\,(\pi_i(V_r)/\mathrm{vol}(V_r))$ trust on average. As the $|\partial(V_r)|$ attack edges are established at random, it is expected that $\mathrm{vol}\,(\partial(V_r))\,(\pi_i(V_r)/\mathrm{vol}(V_r))$ trust, that is, $\chi(V_r) \cdot \pi_i(V_r)$, is passed through these edges to the fake region. The same also holds for the fake region, which means we can model the trust exchange between D_r and D_f by

$$\begin{aligned}\pi_{i+1}(V_r) &= \pi_i(V_r) + \chi(V_f) \cdot \pi_i(V_f) - \chi(V_r) \cdot \pi_i(V_r) \text{ and}\\ \pi_{i+1}(V_f) &= \pi_i(V_f) + \chi(H) \cdot \pi_i(V_r) - \chi(V_f) \cdot \pi_i(V_f),\end{aligned}$$

where the total trust τ is conserved throughout the process, as follows:

$$\pi_{i+1}(V_r) + \pi_{i+1}(V_f) = \pi_i(V_r) + \pi_i(V_f) = \tau.$$

We now consider the base case of this lemma. Initially, for iteration $i = 0$, we have $\chi(V_r) \cdot \pi_0(V_r) = \chi(V_r) \cdot \tau$ and $\chi(V_f) \cdot \pi_0(V_f) = 0$. Therefore,

$$\pi_1(V_f) - \pi_0(V_f) = \chi(V_r) \cdot \tau.$$

We next state the induction hypothesis. For each $i \geq 1$, let us assume the following statement is true:

$$\pi_i(V_f) - \pi_{i-1}(V_f) = (\chi(V_r) \cdot \tau)\,(1 - \chi(V_r) - \chi(V_f))^{i-1}\,.$$

Now, let us consider the trust exchange in iteration $i+1$:

$$\pi_{i+1}(V_f) - \pi_i(V_f) = \chi(V_r) \cdot \pi_i(V_r) - \chi(V_f) \cdot \pi_i(V_f)$$

By substituting $\pi_i(V_r)$ by $\pi_{i-1}(V_r) + \chi(V_f) \cdot \pi_{i-1}(V_f) - \chi(V_r)\tau_{i-1}(V_r)$, and doing similarly so for $\pi_{i-1}(V_f)$, we get

$$\begin{aligned}\pi_{i+1}(V_f) - \pi_i(V_f) &= \chi(V_r)\,((1 - \chi(V_r)) \cdot \pi_{i-1}(V_r) + \chi(V_f) \cdot \pi_{i-1}(V_f)) -\\ &\qquad \chi(V_f)\,((1 - \chi(V_f)) \cdot \pi_{i-1}(V_f) + \chi(V_r) \cdot \pi_{i-1}(V_r))\\ &= (\chi(V_r) \cdot \pi_{i-1}(V_r) - \chi(V_f) \cdot \pi_{i-1}(V_f))\,(1 - \chi(V_r) - \chi(V_f))\,.\end{aligned}$$

We know that $\pi_i(V_f) = \pi_{i-1}(V_f) + \chi(V_r) \cdot \pi_{i-1}(V_r) - \chi(V_f) \cdot \pi_{i-1}(V_f)$, and therefore

$$\pi_{i+1}(V_f) - \pi_i(V_f) = (\pi_i(V_f) - \pi_{i-1}(V_f)) (1 - \chi(V_r) - \chi(V_f)) .$$

Finally, by the induction hypothesis, we end up with

$$\pi_{i+1}(V_f) - \pi_i(V_f) = (\chi(V_r) \cdot \tau) (1 - \chi(V_r) - \chi(V_f))^i ,$$

which, by induction, completes the proof. □

Corollary 2. *In the i-th iteration, the expected increment of aggregate trust in the fake region in upper bounded by* $(\chi(V_r) \cdot \tau) (1 - \chi(V_r))^i$ *for each* $i \geq 0$.

We next bound the aggregate trust in the fake region $\pi_i(V_f)$ after β iterations, where $1 \leq \beta \leq \mathcal{T}(\epsilon) - \Delta$ and $\Delta > 1$ is a positive number. We achieve this by directly comparing $\pi_\beta(V_f)$ to its stationary value $\pi_{\mathcal{T}(\epsilon)}(V_f)$, where $\mathcal{T}(\epsilon) \geq \beta + \Delta$. In fact, this result holds as long as there is at least a constant difference between the mixing time $\mathcal{T}(\epsilon)$ and β, or in other words, whenever $\mathcal{T}(\epsilon) - \beta = \Omega(1)$ and $\mathcal{T}(\epsilon)$ is not arbitrarily large.

Lemma 3. *Given a defense graph D with a mixing time* $\mathcal{T}(\epsilon) \geq 1$ *and a positive integer* $\beta \in [1, \mathcal{T}(\epsilon) - \Delta]$ *where* $\Delta > 1$*, the aggregate trust in the fake region* $\pi_\beta(V_f)$ *after* β *iterations gets a fraction* $f \in (0, 1)$ *of that in the stationary distribution, that is,* $\pi_\beta(V_f) = f \cdot \tau \cdot (\mathrm{vol}(V_f)/\mathrm{vol}(V))$*, where*

$$f = \frac{\beta \cdot \sum_{0 \leq i \leq \beta - \Delta} (1 - \chi(V_r) - \chi(V_f))^i}{\mathcal{T}(\epsilon) \cdot \sum_{0 \leq i \leq \mathcal{T}(\epsilon) - \Delta} (1 - \chi(V_r) - \chi(V_f))^i} \tag{B.4}$$

Proof. By Lemma 2, we know that the aggregate trust in the fake region monotonically increases with the number of iterations in the process defined by Equation A.5. For iteration $i = \beta$, where $1 \leq \beta \leq \mathcal{T}(\epsilon) - \Delta$, we have

$$\begin{aligned} \pi_\beta(V_f) &= \sum_{0 \leq i \leq \beta - \Delta} (\chi(V_r) \cdot \tau) (1 - \chi(V_r) - \chi(V_f))^i \\ &= (\beta \cdot \chi(V_r) \cdot \tau) \sum_{0 \leq i \leq \beta - \Delta} (1 - \chi(V_r) - \chi(V_f))^i . \end{aligned}$$

Similarly, for iteration $i = \gamma$, where $\gamma = \mathcal{T}(\epsilon) \geq \beta + \Delta$, we have

$$\begin{aligned}\pi_\gamma(V_f) &= \sum_{0 \leq i \leq \gamma - \Delta} (\chi(V_r) \cdot \tau) \left(1 - \chi(V_r) - \chi(V_f)\right)^i \\ &= (\gamma \cdot \chi(V_r) \cdot \tau) \sum_{0 \leq i \leq \gamma - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i .\end{aligned}$$

Now let us consider the ratio $\pi_\beta(V_f)/\pi_\gamma(V_f)$. We have

$$\frac{\pi_\beta(V_f)}{\pi_\gamma(V_f)} = \frac{(\beta \cdot \chi(V_r) \cdot \tau) \sum_{0 \leq i \leq \beta - \Delta} \left(1 - \chi(V_f) - \chi(V_f)\right)^i}{(\gamma \cdot \chi(V_r) \cdot \tau) \sum_{0 \leq i \leq \gamma - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i}.$$

By multiplying both side by $\pi_\gamma(V_f)$, we get

$$\pi_\beta(V_f) = \frac{\beta \cdot \sum_{0 \leq i \leq \beta - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i}{\gamma \cdot \sum_{0 \leq i \leq \gamma - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i} \cdot \pi_\gamma(V_f). \tag{B.5}$$

Now, recall that $\pi_\gamma(v_j) = \tau \cdot \pi(v_j) = \tau \cdot (\deg(v_j)/\mathrm{vol}(V))$ for each $v_j \in V_f$, where π is the stationary distribution of the graph D (see Equation A.3), Accordingly,

$$\begin{aligned}\pi_\beta(V_f) &= \frac{\beta \cdot \sum_{0 \leq i \leq \beta - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i}{\gamma \cdot \sum_{0 \leq i \leq \gamma - \Delta} \left(1 - \chi(V_r) - \chi(V_f)\right)^i} \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)} \\ &= f \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)}\end{aligned}$$

Finally, as $\beta \leq \gamma - \Delta$, we have $\beta/\gamma \leq (\gamma - \Delta)/\gamma$. As γ is not arbitrarily large, $\beta/\gamma < 1$ holds. Therefore, $f < 1$. □

As the total trust τ is conserved, Corollary 3 below directly follows.

Corollary 3. *For a positive number $\Delta > 1$, if the aggregate trust in the fake region after $1 \leq \beta \leq \mathcal{T}(\epsilon) - \Delta$ iterations is a fraction $f \in (0, 1)$ of that in the stationary distribution, then the aggregate trust in the real region during the same iteration is $c > 1$ times of that in the stationary distribution, that is, $\pi_\beta(V_r) = c \cdot \tau \cdot (\mathrm{vol}(V_r)/\mathrm{vol}(V))$, where $c = 1 + (1 - f)\,(\mathrm{vol}(V_f)/\mathrm{vol}(V_r))$.*

Given a fraction $f > 1$ and a multiplier $c > 1$, as defined by Lemma 3 and Corollary 3, the trust distribution over nodes in D after β iterations is defined by

$$\pi_\beta(v_j) = \begin{cases} f \cdot \tau \cdot \dfrac{\deg(v_i)}{\mathrm{vol}(V)} < 1 & \text{if } v_j \in V_f, \\ c \cdot \tau \cdot \dfrac{\deg(v_i)}{\mathrm{vol}(V)} > 1 & \text{if } v_j \in V_r. \end{cases} \tag{B.6}$$

Moreover, let $\bar{\pi}_\beta(v_j) = \pi_\beta(v_j)/\deg(v_i)$ be the degree-normalized trust for each $v_j \in V$, as derived from Equation B.6. We next prove that at most $(f/c) \cdot \mathrm{vol}(V_f)$ fake nodes can have degree-normalized trust or *rank* values higher than or equal to $(c \cdot \tau)/\mathrm{vol}(V)$.

Lemma 4. *Consider a defense graph D with a mixing time $\gamma = \mathcal{T}(\epsilon)$, a fraction $f < 0$, and a multiplier $c > 1$ such that $\pi_\beta(V_r) = c \cdot \pi_\gamma(V_r)$ and $\pi_\beta(V_f) = f \cdot \pi_\gamma(V_f)$ after $1 \leq \beta \leq \mathcal{T}(\epsilon) - \Delta$ power iterations for some $\Delta > 1$. Regardless to how an attacker organizes the fake region, there can be only a set $U \subset V_f$ of at most $(f/c) \cdot \mathrm{vol}(V_f)$ fake nodes such that each $v_j \in U$ has a degree-normalized trust $\bar{\pi}_\beta(v_j) \geq (c \cdot \tau)/\mathrm{vol}(V)$.*

Proof. For an attacker, the optimal strategy to maximize the cardinality of the set U is to assign $(c \cdot \tau)/\mathrm{vol}(V)$ degree-normalized trust to as many fake nodes as possible, and then leave the rest of the fake nodes with zero trust.

We now prove by contradiction that $|U| \leq (f/c) \cdot \mathrm{vol}(V_f)$. Assume the opposite, where $|U| > (f/c) \cdot \mathrm{vol}(V_f)$. Since each node $v_j \in U$ is connected to at least another node $v_k \in U$, or otherwise it would be disconnected and $\bar{\pi}_\beta(v_j) = 0$, then the aggregate degree-normalized trust $\bar{\pi}_\beta(V_f)$ in the fake region is

$$\bar{\pi}_\beta(V_f) = |U| \cdot \frac{c \cdot \tau}{\mathrm{vol}(V)} \tag{B.7}$$

$$> \frac{f}{c} \cdot \mathrm{vol}(V_f) \cdot \frac{c \cdot \tau}{\mathrm{vol}(V)} \tag{B.8}$$

$$> f \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)}, \tag{B.9}$$

which by Lemma 3 is a contradiction. □

Finally, we prove an upper bound on $(f/c) \cdot \mathrm{vol}(V_f)$, as follow.

Theorem 2. *Given a social graph with a fast mixing real region and an attacker that randomly establishes attack edges, the number of fake nodes that rank similar to or higher than real nodes after $\beta = O(\log n)$ iteration is $O(\mathrm{vol}(E_a) \cdot \log n)$.*

Proof. Consider the social graph G and its defense graph D. By Lemma 1, we know that re-weighting G changes its mixing time by only a constant factor. Therefore, we have $\mathcal{T}_D(\epsilon) = O(\mathcal{T}_G(\epsilon))$ and $\mathcal{T}_{D_r}(\epsilon) = O(\mathcal{T}_{G_r}(\epsilon))$. As G_r is fast mixing, we also have $\mathcal{T}_{G_r}(\epsilon) = O(\log n)$ by definition. This also means $\mathcal{T}_{D_r}(\epsilon) = O(\log n)$. As $V_f \neq \emptyset$, then by the bound in Equation B.1, we have $\mathcal{T}_D(\epsilon) - \mathcal{T}_{D_r}(\epsilon) = \Omega(1)$. So, $\mathcal{T}_D(\epsilon) - \beta = \Omega(1)$ as $\beta = \mathcal{T}_{D_r}(\epsilon) = O(\log n)$. Finally, by Lemma 4, we know that at most $(f/c) \cdot \mathrm{vol}(V_f)$ fake nodes can rank same or equal to real nodes. We now attempt to prove an upper bound on this quantity.

As the total trust τ is conversed after $\beta = O(\log n)$ iteration, we have

$$f \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)} + c \cdot \tau \cdot \frac{\mathrm{vol}(V_r)}{\mathrm{vol}(V)} = \tau$$

That is,

$$\frac{f}{c} \cdot \mathrm{vol}(V_f) = \frac{\mathrm{vol}(V_r)}{\frac{\mathrm{vol}(V)}{f \cdot \mathrm{vol}(V_f)} - 1}$$

By Lemma 3, we have

$$\frac{f}{c} \cdot \mathrm{vol}(V_f) = \frac{\mathrm{vol}(V_r)}{(\tau/\pi_\beta(V_f)) - 1} \tag{B.10}$$

Now, by Lemma 2 and Corollary 2, we have

$$\begin{aligned}
\pi_\beta(V_f) &= \sum_{0 \leq i \leq \beta - 1} (\chi(V_r) \cdot \tau) \left(1 - \chi(V_r) - \chi(V_f)\right)^i \\
&< \sum_{0 \leq i \leq \beta - 1} (\chi(V_r) \cdot \tau) \left(1 - \chi(V_r)\right)^i \\
&= \sum_{0 \leq i \leq \beta - 1} \tau \cdot \left(\left(1 - \chi(V_r)\right)^i - \left(1 - \chi(V_r)\right)^{i+1} \right) \\
&= \tau \cdot \left(1 - \left(1 - \chi(V_r)\right)^\beta \right) \qquad \text{(B.11)}
\end{aligned}$$

By combining Equations B.10 and B.11, we get

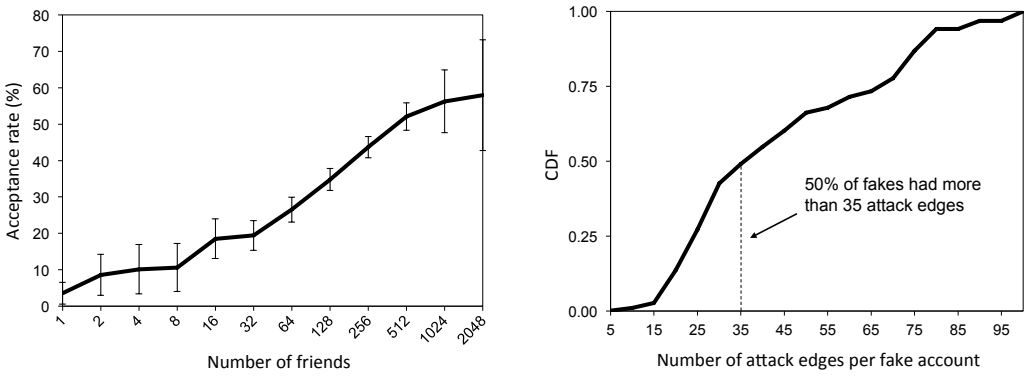
$$\frac{f}{c} \cdot \text{vol}(V_f) < \text{vol}(V_r) \left((1 - \chi(V_r))^{-\beta} - 1 \right)$$

By replacing $(1 - \chi(V_r))^{-\beta}$ with $1+\beta \cdot \chi(V_r)+o\left(\chi^2(V_r)\right)$, which is its Maclaurin series, we end up with the following

$$\begin{aligned} \frac{f}{c} \cdot \text{vol}(V_f) &< \text{vol}(V_r) \left((1 - \chi(V_r))^{-\beta} - 1 \right) \\ &= \text{vol}(V_r) \cdot O\left(\chi(V_r)\right) \cdot \beta \\ &= \text{vol}(V_r) \cdot O\left(\chi(V_r)\right) \cdot O(\log n) \\ &= O\left(\text{vol}(\partial(V_r)) \cdot \log n\right) \\ &= O\left(\text{vol}(E_a) \cdot \log n\right), \end{aligned}$$

which completes the proof. □

List of Figures



(a) User Susceptibility **(b)** Attack scalability

Figure B.1: Social infiltration in Facebook. In (a), while the fakes did not share mutual friends with the invited users, the more friends these users had the more likely they were to accept friend requests sent by fakes (CI=95%). In (b), contrary to what is often assumed, fake accounts can use simple automated social engineering tactics to establish a large number of attack edges.

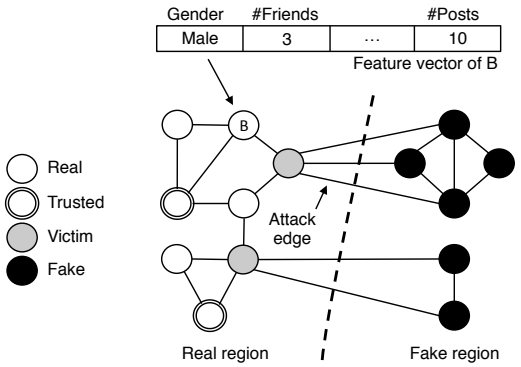
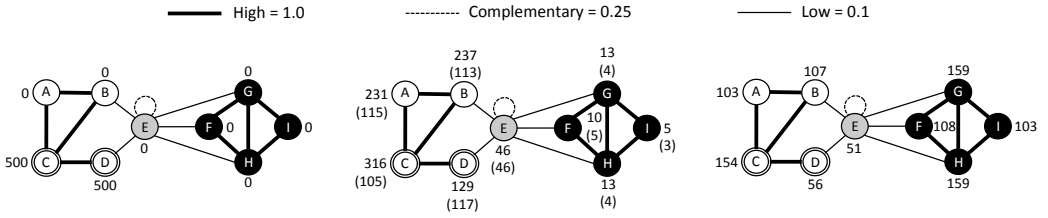


Figure B.2: System model. In this figure, the OSN is represented as a graph consisting of 14 users. There are 8 real accounts, 6 fake accounts, and 5 attack edges. The cut, represented by a dashed-line, partitions the graph into two regions, real and fake. Victim accounts are real accounts that are directly connected to fakes. Trusted accounts are accounts that are known to be real and not victims. Each account has a feature vector representing basic account information. Initially, all edges have a unit weight, so user B for example has a degree of 3.



(a) Initialization **(b)** After 4 iterations **(c)** Stationary distribution

Figure B.3: Trust propagation. Each value is rounded to its nearest natural number. Values in parentheses represent degree-normalized trust (i.e., rank values). In this example, we set α=0.5, β=2, τ=1,000, $p(\cdot)$=0.05 except for $p(E)$=0.95, and $\omega=\lceil \log_2(9) \rceil$=4.

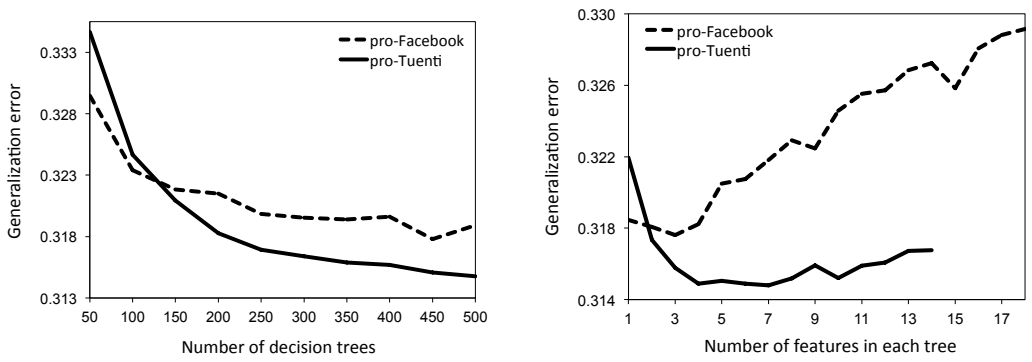
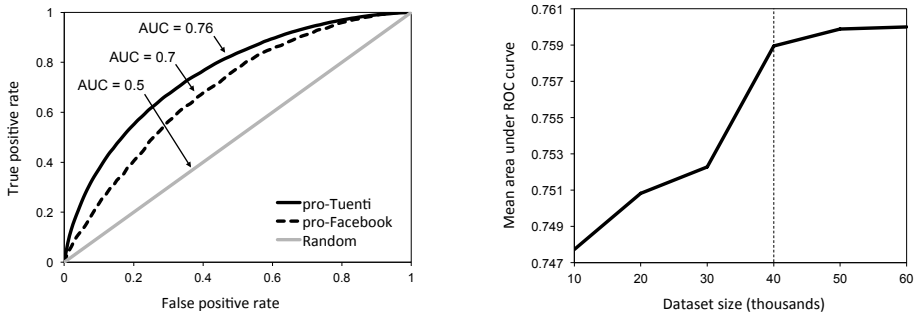
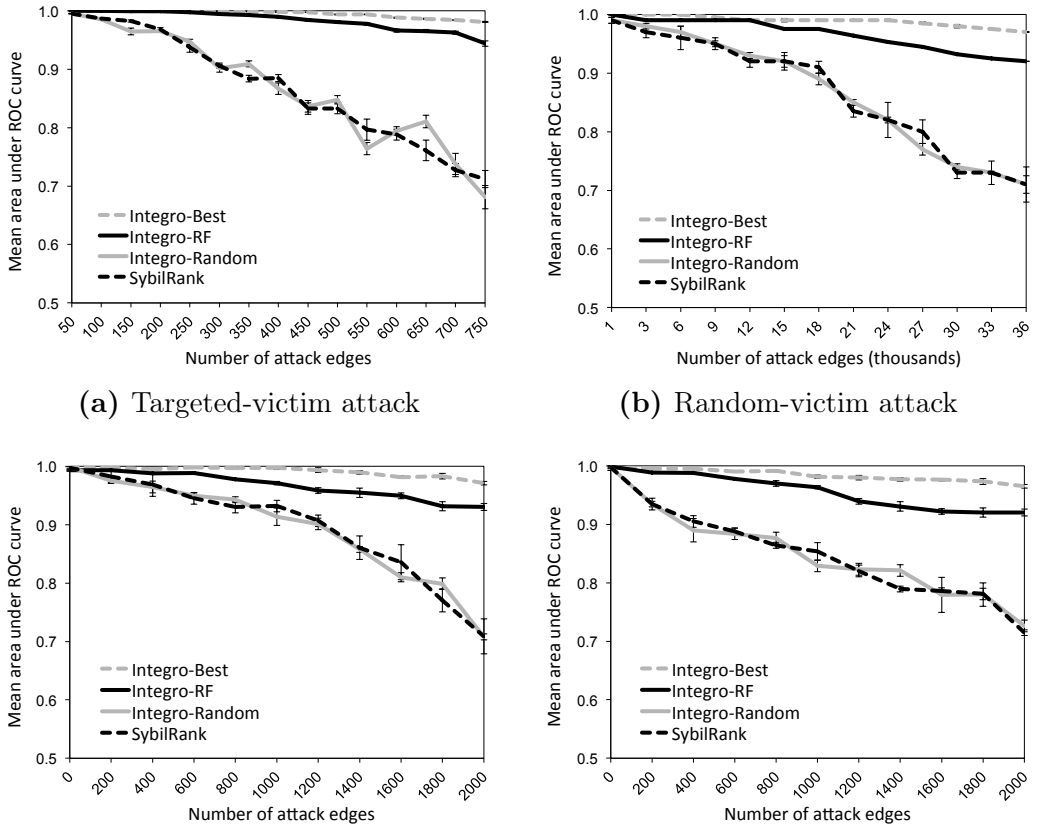


Figure B.4: Classifier tuning with random forests. For the pro-Facebook dataset, we used 450 decision trees. Each tree had 3 features picked at random out of 18 features. For the pro-Tuenti dataset, we used 500 decision trees. Each tree had 7 features picked at random out of 14 features.



(a) ROC Analysis **(b)** Sensitivity to dataset size

Figure B.5: Victim classification using the RF algorithm. In (a), the ROC curves show the tradeoff between FPR and TPR for both datasets. In ROC analysis, the closer the curve is to the upper-left corner the more accurate it is. The area under the ROC curve (AUC) summarizes the classifier's performance. Therefore, an AUC of 1 means a perfect classifier, while an AUC of 0.5 means a random classifier. We require the victim classifier to be better than random. In (b), during cross validation on pro-Tuenti dataset, we observed that increasing the dataset size to more than 40K vectors did not significantly increase the AUC.



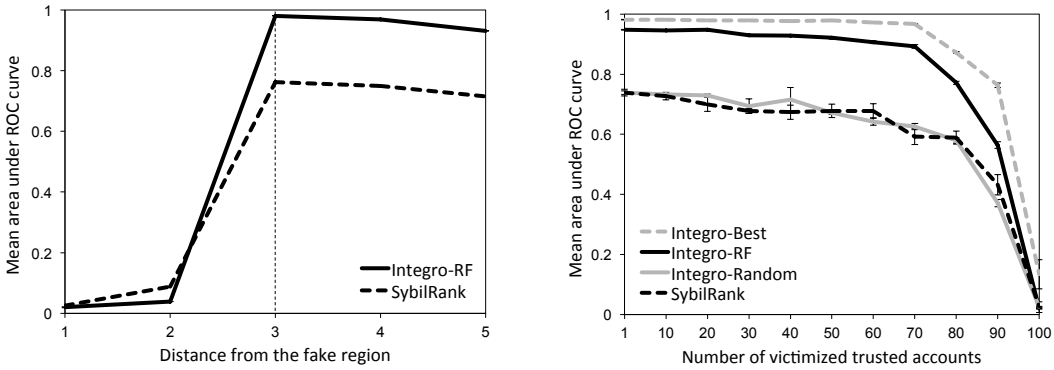
(a) Targeted-victim attack

(b) Random-victim attack

(c) Supplementary, gra-AstroPh

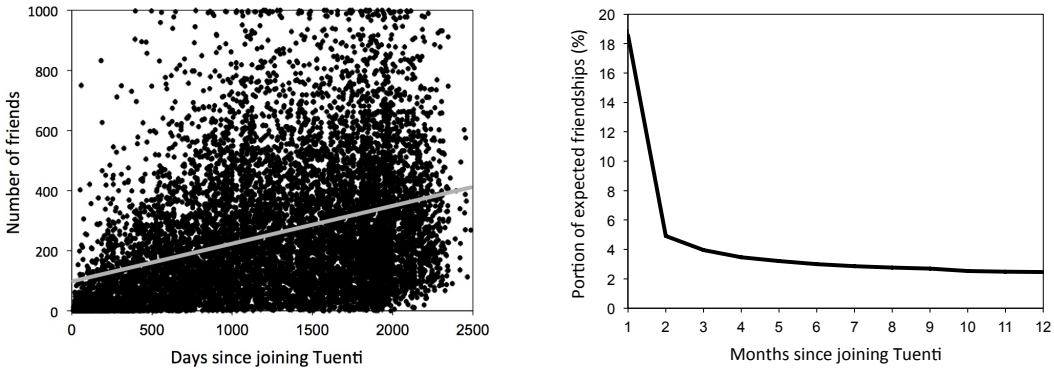
(d) Supplementary, gra-HepTh

Figure B.6: The ranking quality of both systems in terms of its AUC under each infiltration scenario (CI=95%). SybilRank and Íntegro resulted in a similar performance when a random victim classifier is used, which represents a practical baseline for Íntegro. As the number of attack edges increased, SybilRank's AUC decreased down to 0.7, while Íntegro sustained its high performance with AUC > 0.92. The figures in (c–d) show the ranking quality of both systems under the random-victim attack strategy, in which the real regions represent the two supplementary datasets.



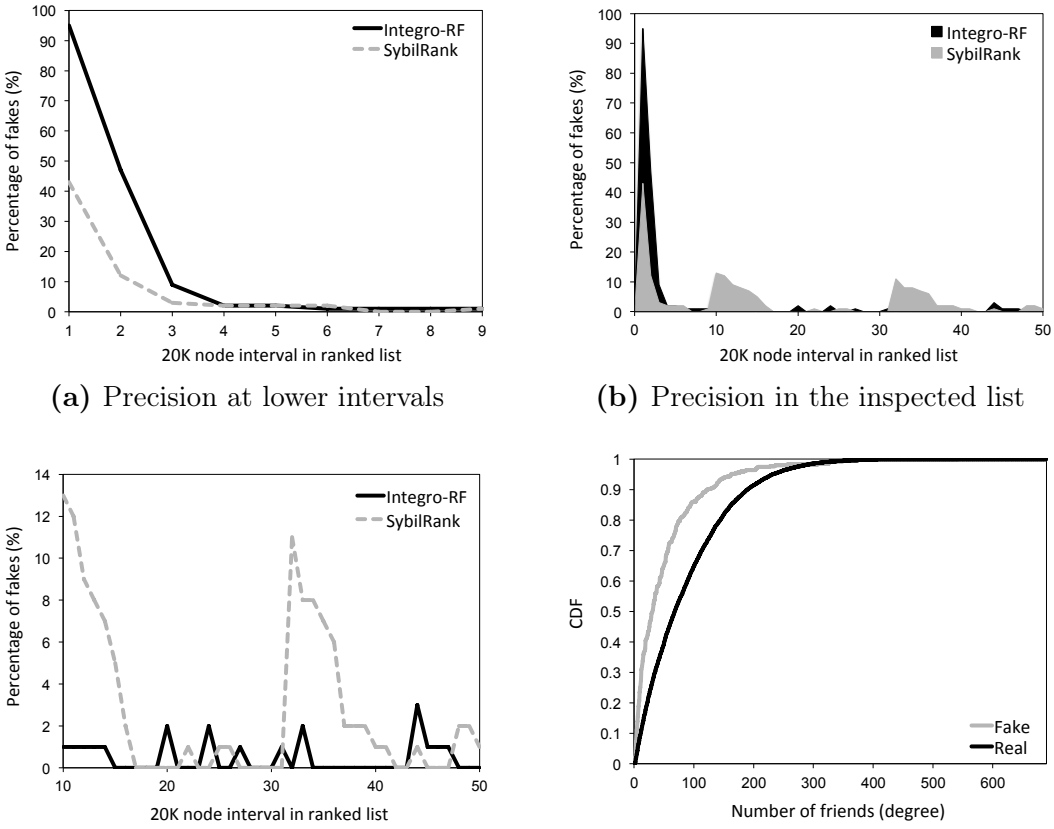
(a) Distant-seed attack **(b)** Random-seed attack

Figure B.7: The sensitivity of both systems to each seed-targeting attack (CI=95%). In distant-seed attack, an attacker befriends users that are at a particular distance from *all* trusted accounts, which represents a practical worst case scenario for both system. In the random-seed attack, the attacker directly befriends a *subset* of the trusted accounts. Overall, both systems are sensitive to seed-targeting attacks.



(a) Users connectivity **(b)** Friendship growth over time

Figure B.8: Preprocessing. In (a), there is a positive correlation between number of days since a user joined Tuenti and how well-connected the user is in terms of number of friends (Pearson's $r = 0.36$). In fact, *93% of all new users who joined Tuenti in the last 30 days had weak connectivity of 46 friends or less*, much smaller than the average of 254 friends. In (b), we found that most of the friendship growth happens in the first month since joining the network, where users on average establish 18.6% of their friendships. We accordingly defer the consideration of users who joined Tuenti in the last 30 days, as they will likely be assigned low ranks.



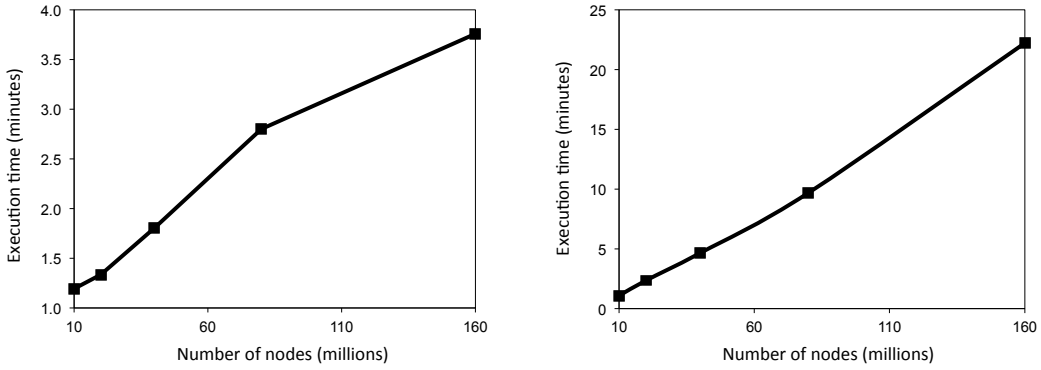
(a) Precision at lower intervals

(b) Precision in the inspected list

(c) Precision at higher intervals

(d) Node degree distribution

Figure B.9: Deployment results at Tuenti. The ranking quality of both systems is summarized in (b). Ideally, all fake accounts should be in the bottom of the ranked list (i.e., left side of the horizontal axis). In (a) and (c), we observed that Íntegro consistently outperforms SybilRank in term of fake account detection precision (i.e., the percentage of fakes in each sample). In particular, most of the fake accounts identified by Íntegro were located at significantly lower locations in the ranked list, unlike SybilRank. Upon further inspection of fakes at higher intervals, we found that they established a large number of attack edges, as suggested by the degree distribution in (d).



(a) Mahout **(b)** Giraph

Figure B.10: System scalability on both platforms. In (a), the execution time includes the time to train an RF classifier and compute a vulnerability score for each node in the graph. In (b), the execution time includes the time to weight the graph, rank nodes, and finally sort them.

List of Tables

Name	Type	Brief description	Source
pro-Facebook	Profiles	8,888 users (32.4% victims)	Boshmaf et al. [6]
pro-Tuenti	Profiles	60,000 users (50% victims)	Tuenti
gra-FacebookTs	Graph	2,991 users (2.2% fake)	Boshmaf et al. [6]
gra-FacebookRd	Graph	6,136 users (0% fake)	Boshmaf et al. [6]
gra-HepTh	Graph	8,638 users (0% fake)	Leskovec et al. [64]
gra-AstroPh	Graph	17,903 users (0% fake)	Leskovec et al. [64]

Table B.1: Datasets summary.

Feature	Brief description	Type	RI Score (%)	
			pro-Facebook	pro-Tuenti
User activity:				
Friends	Number of friends the user had	Numeric	100.0	84.5
Photos	Number of photos the user shared	Numeric	93.7	57.4
Feed	Number of news feed items the user had	Numeric	70.6	60.8
Groups	Number of groups the user was member of	Numeric	41.8	N/A
Likes	Number of likes the users made	Numeric	30.6	N/A
Games	Number of games the user played	Numeric	20.1	N/A
Movies	Number of movies the user watched	Numeric	16.2	N/A
Music	Number of albums or songs the user listened to	Numeric	15.5	N/A
TV	Number of TV shows the user watched	Numeric	14.2	N/A
Books	Number of books the user read	Numeric	7.5	N/A
Personal messaging:				
Sent	Number of messages sent by the user	Numeric	N/A	53.3
Inbox	Number of messages in the user's inbox	Numeric	N/A	52.9
Privacy	Privacy level for receiving messages	5-Categorical	N/A	9.6
Blocking actions:				
Users	Number of users blocked by the user	Numeric	N/A	23.9
Graphics	Number of graphics (photos) blocked by the user	Numeric	N/A	19.7
Account information:				
Last updated	Number of days since the user updated the profile	Numeric	90.77	32.5
Highlights	Number of years highlighted in the user's time-line	Numeric	36.3	N/A
Membership	Number of days since the user joined the OSN	Numeric	31.7	100
Gender	User is male or female	2-Categorical	13.8	7.9
Cover picture	User has a cover picture	2-Categorical	10.5	< 0.1
Profile picture	User has a profile picture	2-Categorical	4.3	< 0.1
Pre-highlights	Number of years highlighted before 2004	Numeric	3.9	N/A
Platform	User disabled third-party API integration	2-Categorical	1.6	< 0.1

Table B.2: Low-cost features extracted from pro-Facebook and pro-Tuenti datasets. The RI score is the relative importance of the feature. A value of "N/A" means the feature was not available for this dataset. A k-Categorical feature means this feature can have one value out of k categories (e.g., boolean features are 2-Categorical).