

Combating Friend Spam Using Social Rejections

Qiang Cao Michael Sirivianos[†] Xiaowei Yang Kamesh Munagala
Duke University [†]Cyprus University of Technology

Abstract— Unwanted friend requests in online social networks (OSNs), also known as friend spam, are among the most evasive malicious activities. Friend spam can result in OSN links that do not correspond to social relationship among users, thus pollute the underlying social graph upon which core OSN functionalities are built, including social search engine, ad targeting, and OSN defense systems. To effectively detect the fake accounts that act as friend spammers, we propose a system called *Rejecto*. It stems from the observation on social rejections in OSNs, i.e., even well-maintained fake accounts inevitably have their friend requests rejected or they are reported by legitimate users. Our key insight is to partition the social graph into two regions such that the aggregate acceptance rate of friend requests from one region to the other is minimized. This design leads to reliable detection of a region that comprises friend spammers, regardless of the request collusion among the spammers. Meanwhile, it is resilient to other strategic manipulations. To efficiently obtain the graph cut, we extend the Kernighan-Lin heuristic and use it to iteratively detect the fake accounts that send out friend spam. Our evaluation shows that *Rejecto* can discern friend spammers under a broad range of scenarios and that it is computationally practical.

I. INTRODUCTION

Unscrupulous users increasingly find Online Social Networking (OSN) platforms as lucrative targets for malicious activities, such as sending spam and spreading malware. The profitability of such activities and the fact that a large portion of the OSN communication takes place over symmetric social links (e.g., Facebook) motivate attackers to connect to real users. In particular, attackers leverage the open nature of OSNs and send to legitimate users unwanted friend requests, also known as *friend spam* [10], [30].

Friend spam can result in OSN links that do not correspond to social relationship among users, thus it enables the pollution of the underlying undirected social graph. Because OSN providers build on social graphs their core functionalities that often assume a social graph solely consists of links representing social trust of user pairs, the consequences of falsely accepted requests by unsuspected users are severe. In particular, the false OSN links resulting from friend spam can compromise the accuracy of social ad targeting [22], [31] and search [5], [8], and the privacy of shared content by users.

Moreover, friend spam can be used to undermine the effectiveness of defense systems that are either built upon [14], [17], [34] or take input signals [33] from social graphs. For example, the additional OSN links that fake accounts (called Sybils) obtain via friend spam can enable part of them to evade the detection of social-graph-based defense systems [14], [17], [34]. This is because these approaches bound the number of undetected fake accounts to the number of OSN links between Sybils and real users, e.g., $O(\log n)$ accounts per link [14],

[34], where n is the total number of users.

Despite major advances in the suppression of malicious activities and accounts [10], friend spam appears to be more evasive than regular spam [6]. OSN users still experience friend spam on a frequent basis [3], [9]. One of the suggested remedies has been to restrict requests only to friends of friends. This, however, subtracts from the openness of the OSN.

We therefore aim to answer the question: “how can we design a robust system to throttle friend spam in OSNs?” Our hypothesis is that we can uncover the fake accounts (Sybils) that indiscriminately send out friend spam in symmetric OSNs (e.g., Facebook and LinkedIn) by leveraging the rejection of unwanted friend requests. The insight is that although some OSN users accept friend requests from unknown users, cautious users reject, ignore, or report them to the OSN providers. The attackers behind the spamming accounts usually have limited knowledge about the degree of their targets’ security awareness, due to the massive scale of today’s OSNs and their significant efforts in protecting the online privacy and safety of the users. As a result, the spamming accounts inevitably receive a significant number of *social rejections* from legitimate users. We confirmed this in our study on fake Facebook accounts in the wild (§II). In contrast, friend requests from legitimate users are only sporadically rejected because they are often sent to people the senders know [33].

We designed and implemented a system called *Rejecto*. It exploits the readily available social rejections in OSNs and systematically uncovers the fake accounts that act as friend spammers. *Rejecto* monitors the friend requests sent out by users and augments the social graph with directed social rejections. Once an OSN detects the fake accounts used for friend spam, it can prevent them from sending requests in the future. The goal of our system is to be able to effectively identify fractions of users that participate in friend spam with a high recall and very low false positives.

Although using social rejections to combat friend spam is intuitive, designing a robust scheme that is strategy-proof poses an algorithmic challenge. First, the spamming fake accounts can attempt to evade the detection by *collusion*. Specifically, they can accept each other’s requests, decreasing the fraction of the rejected requests of each individual account to that of a legitimate user’s. Second, a part of the fake accounts can mimic legitimate users by rejecting friend requests from other fake accounts. By doing so the attacker sacrifices his accounts that got the rejections. Yet, he whitewashes his rejecting accounts because they now reject requests in the same way legitimate users do. We call this strategy *self-rejection*.

Previous work [15], [32] took request rejections into account

for fake account detection. Like Rejecto, they also rely on the premise that legitimate users reject a portion of unwanted friend requests. However, these designs [15], [32] did not fully utilize the power of social rejection and were not made resilient to the collusion and self-rejection attack strategies. Similarly, Machine Learning classifiers that use request rejections [33] are usually based on individual user features, thus they suffer from manipulation.

Rejecto systematically addresses the above challenges. To be resilient to collusion, it formulates the friend spammer detection as a graph partitioning algorithm (§IV-A). Specifically, while the portion of the accepted friend requests among legitimate accounts is high, the aggregate acceptance rate of all the requests sent from fake to legitimate accounts is substantially lower, regardless of the acceptance of the requests among the fake accounts. Therefore, Rejecto extends the Kernighan-Lin approach [23] and uses it to partition the social graph into two regions such that the aggregate acceptance rate of the requests from one region to the other is minimized. We then declare the accounts in the region with the minimum aggregate acceptance rate as suspicious. Because this aggregate acceptance rate is independent of the requests and links among fake accounts, an attacker cannot arbitrarily boost this rate by having his accounts befriend each other.

To mitigate the impact of the self-rejection attack strategy and cope with the existence of multiple independent groups of fake accounts, we apply the graph partitioning multiple times and iteratively identify fake-account groups after pruning the detected ones from the social graph (§IV-E).

We implemented Rejecto on Spark [36], an efficient in-memory large-data processing platform. We evaluate Rejecto through extensive simulations (§V) on real social graphs, and we show that it can withstand friend spam under a broad range of scenarios and that it is resilient to attack strategies. As a demonstration of Rejecto’s effectiveness on improving OSN-related services, we applied it for an in-depth defense against OSN fake accounts in combination with existing social-graph-based schemes (§II-B, §V-D).

In summary, this work makes the following contributions:

- We formulate the detection of the fake accounts that act as friend spammers as a graph partitioning problem. To the best of our knowledge, this is the first formulation that makes a friend-spam detection system resilient to attack strategies.
- To efficiently partition a rejection-augmented social graph, we transform to a set of partitioning problems, each with a linear objective function, and we extend the Kernighan-Lin approach [23] to solve them.
- We have implemented a scalable prototype of Rejecto that can process multimillion-user social graphs on an EC2 cluster. Our efficiency analysis indicates that the prototype can scale up to OSNs with hundreds of millions of users, provided that the aggregate memory of the cluster suffices.

II. MOTIVATION

We present a study of fake Facebook accounts we obtained from the underground market. This study reveals that social re-

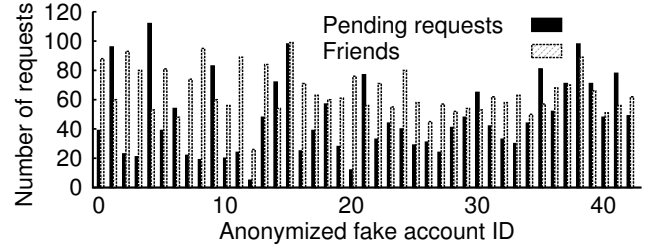


Fig. 1: The numbers of friends and pending requests on purchased accounts. These accounts have a significant fraction of pending requests.

jections are a natural byproduct of friend spam, and motivates us to use them to trace and defend against the spammers.

A. Social rejections by legitimate users

Fake accounts are incentivized to obtain OSN links to real users. However, those that aggressively befriend legitimate users in symmetric OSNs trigger social rejections in the form of rejected, ignored, or reported as abusive friend requests. Since user reports are only accessible by OSN providers, we focus our study on the ignored and rejected friend requests.

Study of purchased Facebook accounts. We conducted a study of well-maintained fake Facebook accounts from the underground market [1], [2] to understand the social rejections in the real world. We purchased fake Facebook accounts from workers on Freelancer [2] for this study. Because our purchases solely played the role of collecting fake accounts that are for sale and nevertheless available in the underground market, the Duke University Institutional Review Board (IRB) informed us that this study does not require a formal approval process. The fake accounts in the underground market are priced according to their lifetime, the number of friends, the number of profile pictures, and other factors. To investigate the trace of social rejections on live fake accounts, we explicitly required that each of the fake accounts we purchase should have “>50 real US friends”. In this study, we use 43 purchased accounts, each of which is at least one year old. They are purchased at different vendors, and in total have 2804 friends and 2065 pending requests, as shown in Figure 1.

The number of friends our purchased accounts have on Facebook is only an upper bound on the number of real users that they successfully befriended. This is because part of the delivered Facebook friends on purchased fake accounts may be also fake, although we required real-user friends when we placed our orders. To this end, we studied the associated attributes of the friends on our purchased accounts. Figure 2 shows the CDF of the friends with respect to their degrees in the social graph. We can see that some of the friends have a social degree >1000 each, indicating that they are either careless Facebook users or abusive fake accounts. Figure 3 and Figure 4 shows the CDFs of the friends with respect to the posts on their walls and the photos they have uploaded. A large portion of the friend users on our purchased accounts are quite active, as they have posted on walls and uploaded

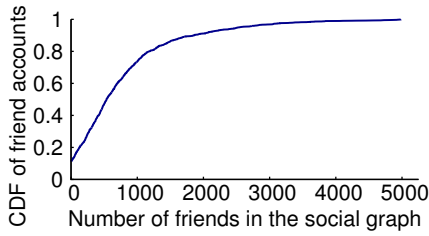


Fig. 2: CDF of the friend accounts of our purchased accounts, with respect to their degrees on the social graph.

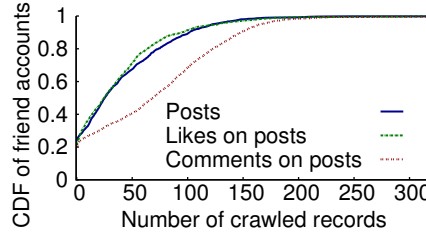


Fig. 3: CDFs of the friend accounts of our purchased accounts, with respect to the number of posts, and to the numbers of the comments and likes on the posts, respectively.

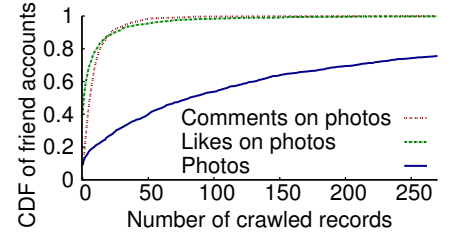


Fig. 4: CDFs of the friend accounts of our purchased accounts, with respect to the number of photos, and to the numbers of the comments and likes on the photos, respectively.

photos, and they got comments and likes from their friends. Again, we cannot determine if the friends of a friend are fake or not. However, we do observe a part of the friends of our purchased accounts form full-mesh communities, which might indicate that they can be fake.

Fake accounts receive rejections. We examine the pending friend requests from each purchased account using Facebook APIs. Figure 1 shows the numbers of friends and pending requests on each account. Although these well-maintained accounts have many OSN links and sufficiently completed profiles, we can see that they all have a significant number of pending requests. Specifically, the fraction of pending requests of each user ranges from 16.7% to 67.9%. This indicates that legitimate OSN users tend to reject requests from unknown and possibly fake users. In addition, a recent study on RenRen [33] also reported excessive rejections on fake accounts, but focusing on those that had been caught.

Non-manipulability of social rejection towards innocent users. In a symmetric OSN, a social rejection is a reliable signal, which is guarded by a feedback loop between two different users, the sender and the receiver. A user is able to signal a social rejection only if she received an unwanted friend request. That is, a legitimate user rarely receives any social rejections if she never sends out spam requests. Therefore, a group of malicious users cannot collude to give an arbitrary number of social rejections to another group of victims who never sent to them friend spam. If such manipulation was effective, it would discredit the defense scheme. This is in contrast to the negative ratings in other online services such as YouTube, where users are allowed to rate arbitrarily.

B. Defense in depth with social-graph-based approaches

A further motivation of our work is to build an in-depth defense against Sybil attacks in OSNs that combines Rejecto with existing social-graph-based approaches. These existing schemes uncover clusters of OSN fake accounts that have a limited number of links to real users [14], [17], [34]. Meanwhile, Rejecto prevents fake accounts from obtaining a large number of OSN links by detecting those friend spammers. Thus, they can constitute a defense in depth. After effectively blocking friend spammers and pruning their OSN links from the graph, a large portion of the clusters of fake accounts with a limited number of OSN links to real users can be uncovered

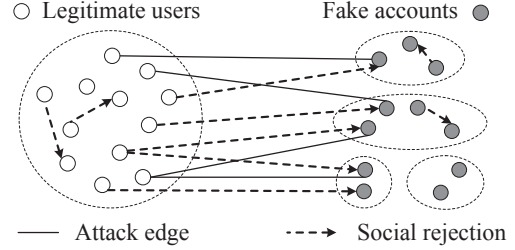


Fig. 5: Legitimate users, fake accounts, and social rejections in an OSN under friend spam. Fake accounts can form distinct groups. The aggregate acceptance rate of requests from a group of friend spammers to legitimate users is low. A fake-account group may choose not to send out friend spam.

by traditional social-graph-based approaches. We use OSN Sybil defense as a demonstration of Rejecto’s effectiveness on improving existing OSN functionality (§V-D).

III. MODELS AND GOALS

We introduce our system model, threat model, and goals.

A. Models

System model. We augment the social graph with directed social rejections, which we call an *augmented social graph*. We model this graph as $G = (V, F, \vec{R})$, where V is the user set in an OSN, F represents the OSN links among users $\{(u, v)\}$, and \vec{R} represents the social rejections $\{\langle u, v \rangle\}$. We consider OSN links bidirectional whose establishment requires mutual agreement as in the real world. The social rejections are directional: a rejection edge $\langle u, v \rangle$ represents a rejection of user v ’s request by user u or a report by user u that flags v ’s request as abusive. As in previous work [14], [17], [34], [35], we refer to the OSN links that straddle the boundary between fake accounts and legitimate users as *attack edges* (Figure 5).

The user set V consists of two disjoint parts: a) L : the subset of legitimate users, and b) M : the subset of fake accounts controlled by attackers, where $V = L \cup M$. For any two disjoint user subsets X and Y ($X \cap Y = \emptyset$), where Y can be \bar{X} ($\bar{X} = V - X$), we define the following:

Group friendship set $F(X, Y)$: the set of friendships each of which connects a user in X with another user in Y . $F(X, Y) = \{(u, v) | u \in X, v \in Y\}$. $F(X, Y) = F(Y, X)$ due to the reciprocation of social connection.

Group rejection set $\vec{R}\langle X, Y \rangle$: the set of rejections casted by users in X to users in Y . $\vec{R}\langle X, Y \rangle = \{\langle u, v \rangle | u \in X, v \in Y\}$.

Aggregate acceptance rate $\vec{A}\langle X, Y \rangle$: the aggregate acceptance rate of the friend requests from X to Y . $\vec{A}\langle X, Y \rangle = \frac{|F(Y, X)|}{|F(Y, X)| + |\vec{R}\langle Y, X \rangle|}$. This rate only considers the OSN links and rejections across the user sets.

Threat model. Fake accounts send out friend spam in the hope of gaining additional OSN links to legitimate users. Meanwhile, they may seek to hide themselves or even disguise as legitimate users. We consider two attack strategies which we believe form the basis of the spectrum of various attacks: *collusion* and *self-rejection*. In a collusion attack, fake accounts arbitrarily improve the acceptance rate of the requests of each individual by sending and accepting requests among themselves. Using the self-rejection strategy, fake accounts mimic legitimate users by rejecting requests from others. Although fake accounts cannot manipulate the requests from legitimate users, they can choose to reject requests from themselves and whitewash the rejecting accounts.

B. Goals

Our aim is to use social rejection to defend against the fake accounts that send out friend spam. We have two main goals:

Accuracy. Rejecto should be able to identify the fake accounts that send out unwanted requests to legitimate users but do not get a large portion accepted, irrespective of the request acceptance among the fake accounts. Rejecto must remain effective even under various strategic attacks. Meanwhile, Rejecto should not flag legitimate users with sporadic rejections.

Efficiency. Today's OSNs serve billions of users. Our system should be able to efficiently detect friend spammers and enable large OSNs to protect their services and users.

IV. SYSTEM DESIGN

We now describe the design of Rejecto. We first provide an overview of the approach.

A. Overview

Rejecto is based on the observation that fake accounts sending out friend spam inevitably receive a significant number of social rejections from the OSN users they attempt to connect to. As a result, the excessive social rejections lead to a low aggregate acceptance rate of the friend requests from fake to legitimate accounts.

Our key insight is that the low aggregate acceptance rate of the spam requests can enable us to reliably differentiate the spammers from other users. Meanwhile, we build into Rejecto's design the resilience to strategic befriending behavior by spammers. In order to be resistant to collusion attacks, Rejecto augments the social graph with directed rejections and formulates the detection of friend spammers as the problem of partitioning an augmented social graph into two regions, such that the aggregate acceptance rate of the requests from one region to the other is minimized. We reduce to this detection problem from the minimum ratio cut in multi-commodity

flow [12], [25], which is known to be NP-hard [20]. We extend a widely-used optimization heuristic, the Kernighan-Lin (KL) algorithm [23], and use it to accurately uncover the fake accounts used for friend spam. To cope with disjoint fake-account groups and the fake accounts mimicking real users by rejecting other Sybils, our system applies KL over multiple rounds, and iteratively identifies groups of friend spammers and their associated links and rejections in the social graph.

By detecting friend spammers and removing the OSN links yielded by friend spam, Rejecto can sterilize the social graph upon which many OSN functionalities rely. In particular, after pruning the attack edges of friend spammers, Rejecto can substantially strengthen social-graph-based Sybil defenses, constituting a significant component of an in-depth Sybil defense system (§II-B).

B. The hardness of the problem

Because the aggregate acceptance rate of the friend requests from a fake-account group to legitimate users is low, we formulate the spammer/legitimate cut as the *minimum aggregate acceptance rate* (MAAR) cut in the augmented social graph. Suppose M' ($M' \subseteq M$) is the group of fake accounts whose requests yield the lowest acceptance rate, we have

$$\frac{|F(\bar{M}', M')|}{|F(\bar{M}', M')| + |\vec{R}\langle \bar{M}', M' \rangle|} \leq \frac{|F(\bar{X}, X)|}{|F(\bar{X}, X)| + |\vec{R}\langle \bar{X}, X \rangle|},$$

that is, $\vec{A}\langle M', \bar{M}' \rangle \leq \vec{A}\langle X, \bar{X} \rangle$, for every $X \subset V$. By solving this MAAR problem, we identify the fake-account group M' . To uncover all friend spammers, we iteratively identify groups of fake accounts by repeatedly solving the MAAR problem and pruning the identified groups from the graph (§IV-E).

The MAAR problem is closely related to the MIN-RATIO-CUT problem [20]. By constructing a linear cost reduction from the MIN-RATIO-CUT problem, we show that it is NP-hard to find a MAAR cut in a social graph.

A primer on the MIN-RATIO-CUT problem. The MIN-RATIO-CUT problem is defined in the context of multi-commodity flow. Let $G = (V, E, c)$ be an undirected graph, where each edge e ($e \in E$) has a non-negative capacity $c(e)$. There is a set of k commodities $\{\vec{K}_1, \vec{K}_2, \dots, \vec{K}_k\}$. Each commodity \vec{K}_i is defined by a tuple $\vec{K}_i = \langle s_i, t_i, d_i \rangle$ ($s_i, t_i \in V$), where s_i and t_i are the *source* and the *sink*, and d_i is a positive *demand*. Given a cut $C = (U, \bar{U})$ such that $U \cup \bar{U} = V$ and $U \cap \bar{U} = \emptyset$, the *capacity* of the cut is the sum of the capacities of the cross-partition edges. The *cut ratio* is the capacity of C divided by the sum of demands of the cross-partition commodities. Therefore, the min ratio cut is a cut $C' = (U', \bar{U}')$ that minimizes:

$$O_{MR}(U) = \frac{\sum_{e \in E \cap (U, \bar{U})} c(e)}{\sum_{\langle s_i, t_i \rangle \in \langle U, \bar{U} \rangle \cup \langle \bar{U}, U \rangle} d_i}$$

Accordingly, the ratio of cut C' is called the *minimum cut ratio*. The MIN-RATIO-CUT problem is NP-hard [20].

Reduction from the 2-approximation MIN-RATIO-CUT problem. To detect friend spammers, we search for a partition

of the user set $C^* = \langle U^*, \bar{U}^* \rangle$ with the minimum aggregate acceptance rate of the friend requests from U^* to \bar{U}^* .

$$U^* = \arg \min_{U \subset V} \frac{|F(\bar{U}, U)|}{|F(\bar{U}, U)| + |\bar{R}(\bar{U}, U)|}$$

Minimizing the aggregate acceptance rate is equivalent to minimizing the aggregate friends-to-rejections ratio $\frac{|F(\bar{U}, U)|}{|\bar{R}(\bar{U}, U)|}$, which we denote as $O_{\text{MAAR}}(U)$. We then reduce to our friend spammer detection problem from the 2-approximation MIN-RATIO-CUT problem, in which the edge capacities and the commodity demands are constants.

Suppose we have a MIN-RATIO-CUT instance $G = (V, E, c)$ with commodities $\{\vec{K}_1, \vec{K}_2, \dots, \vec{K}_k\}$, where the capacity of an edge and the demand of a commodity are the same constant. We construct a corresponding MAAR instance in a social network (V, F, \bar{R}) , where $F = E$ and $\bar{R} = \{\vec{K}_1, \vec{K}_2, \dots, \vec{K}_k\}$. The MIN-RATIO-CUT instance is slightly different from the MAAR instance, in that the former counts the commodities in both directions across partitions, whereas MAAR considers only the rejections from legitimate users to Sybils. We now show that the optimal value of the MAAR instance is within a factor of two of that of the MIN-RATIO-CUT instance.

We compare the objectives of these two instances. We have $O_{\text{MR}}(U) = \frac{|F(\bar{U}, U)|}{|\bar{R}(\bar{U}, U)| + |\bar{R}(\bar{U}, U)|}$ and $O_{\text{MAAR}}(U) = \frac{|F(\bar{U}, U)|}{|\bar{R}(\bar{U}, U)|}$. For any cut $C = (U, \bar{U})$, let U be the partition that has the larger number of incoming rejections, i.e., $|\bar{R}(\bar{U}, U)| \geq \frac{1}{2}(|\bar{R}(\bar{U}, \bar{U})| + |\bar{R}(\bar{U}, U)|)$. Then $O_{\text{MAAR}}(U) \leq 2O_{\text{MR}}(U)$ holds. As a result, after traversing all possible cuts, we have $\min\{O_{\text{MAAR}}\} \leq 2\min\{O_{\text{MR}}\}$ (by contradiction). This indicates that the optimal value of the MAAR instance is larger than that of the MIN-RATIO-CUT, but at most twice as large.

Therefore, the optimal value of the MAAR problem is always within a factor of two of the ratio of the MIN-RATIO-CUT. Meanwhile, the MIN-RATIO-CUT problem is known to be NP-hard. The best existing MIN-RATIO-CUT algorithms achieve only an approximation factor of $O(\log |V|)$ [28].

C. The Kernighan-Lin algorithm

We use the Kernighan-Lin (KL) approach [23], one of the most effective heuristic algorithms for graph bisection. KL bipartitions an undirected graph into balanced parts, while minimizing the number of edges across parts. In particular, on an undirected graph $G = (V, E)$ KL seeks an approximation of the optimal cut $C = (U, \bar{U})$, such that $|U|/|V| \simeq r$ ($0 < r < 1$) and $|\{(u, v) | (u, v) \in E \cap (U \times \bar{U})\}|$ is minimized.

KL is a local optimization heuristic. It is based on the observation that one can always obtain the desired partition from an initial one by interchanging a particular subset of nodes from each part. Figure 6 illustrates that a partition can be refined by switching the misplaced nodes.

KL improves the partition quality by iteratively interchanging node sets. Within each iteration, it interchanges a pair of node sets that are sequentially selected in a greedy fashion according to the reduction of cross-part edges. This reduction

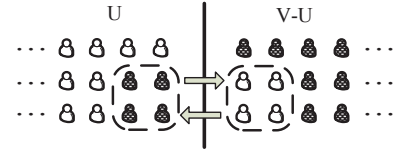


Fig. 6: An illustration of the Kernighan-Lin algorithm that improves a partition by interchanging misplaced nodes.

is called *gain*. Specifically, the selection of node sets to be interchanged consists of two steps. First, it sequentially identifies a series of node-pair interchange operations each of which yields the largest gain if its predecessors are applied. In this series, each subsequence starting from the beginning (prefix) indicates a candidate pair of interchanging node sets. Second, KL finds the interchange prefix that leads to the highest cumulative reduction of cross-part edges. It swaps the pair of node sets specified by the selected interchange prefix, generating an improved partition. KL then initiates the next iteration with this new partition it has obtained. This procedure repeats until no improvement can be made.

To enable efficient lookup for the node with the largest gain during the optimization, Fudiccia et al. [19] improved KL with the use of an array of linked lists, called a *bucket list*, which indexes each node according to its potential gain on the cross-part edge reduction (see [11] for details). In practice, since KL only requires a very small number of iterations, this improvement results in an $O(|V|)$ algorithm [19].

Alternative techniques. There are off-the-shelf approximation algorithms [28] for the MIN-RATIO-CUT problem, with an approximation factor of $O(\log |V|)$. We do not adopt them for large OSNs due to two shortcomings we perceived empirically: a) the approximation factor $O(\log |V|)$ of the cut ratio is not sufficient, which may cause substantial deviation from the optimal partitioning; b) the complexity of the algorithms indicates that it can be difficult for them to handle graphs with millions or billions of nodes. These algorithms [28] have a complexity of $\tilde{O}(|V|^2)$, where the $\tilde{O}(\cdot)$ notation suppresses poly-logarithmic factors. In addition, they do not have parallel implementation, making them hard to scale to today's OSNs.

D. Extending KL to rejection-augmented social graphs

Rejecto aims to find a partition of the entire user set $C^* = \langle U^*, \bar{U}^* \rangle$ that minimizes the friends-to-rejections ratio $\frac{|F(\bar{U}, U)|}{|\bar{R}(\bar{U}, U)|}$. We cannot directly apply KL to our rejection-augmented social graph, because KL is designed to minimize the cross-region edges in an undirected graph, rather than the ratio of different types of edges. Therefore, we transform the objective of minimizing the friends-to-rejections ratio and extend KL to the rejection-augmented social graphs.

The core of our transformation is a conversion of our problem targeting a ratio objective to a set of partitioning problems each with a linear objective function that is compatible with KL. We then extend the KL algorithm to solve each derived problem. Specifically, instead of directly minimizing

Algorithm 1 ExtendedKL ($G(V, F, \vec{R}), k, \text{initPartition}$)

```
1: p = initPartition
2: repeat
3:   // Initiate the switching gain for each node
4:   nodeGainList = BucketList.init(p)
5:   nodeSwitchingSeq =  $\emptyset$ 
6:   p_tmp = p // Used to get the node-switching series
7:   while nodeGainList  $\neq \emptyset$  do
8:     u = nodeGainList.getMaxGainNode()
9:     nodeSwitchingSeq.add(u)
10:    p_tmp.switch(u)
11:    nodeGainList.delete(u)
12:    // Update the gain of u's each neighbor according
13:    // to the objective  $|F(\bar{U}, U)| - k \times |\vec{R}(\bar{U}, U)|$ 
14:    nodeGainList.update(neighbor(u), p_tmp)
15:  end while
16:  // Get the prefix of nodeSwitchingSeq with the largest
17:  // positive decrement in  $|F(\bar{U}, U)| - k \times |\vec{R}(\bar{U}, U)|$ 
18:  prefix = nodeSwitchingSeq.getMaxGainPrefix()
19:  p.switch(prefix)
20: until prefix ==  $\emptyset$ 
21: return p
```

the friends-to-rejections ratio $\frac{|F(\bar{U}, U)|}{|\vec{R}(\bar{U}, U)|}$, we examine a family of cuts that minimize $|F(\bar{U}, U)| - k \times |\vec{R}(\bar{U}, U)|$, where k is a positive parameter. The cut among the family that yields the lowest friends-to-rejections ratio is the solution (Theorem 1). We then extend the KL approach to solve each partition problem that minimizes $|F(\bar{U}, U)| - k \times |\vec{R}(\bar{U}, U)|$.

Theorem 1: In a rejection-augmented social graph, if the cut $C^* = \langle U^*, \bar{U}^* \rangle$ is the minimum aggregate acceptance rate (MAAR) cut, and $\frac{|F(\bar{U}^*, U^*)|}{|\vec{R}(\bar{U}^*, U^*)|} = k^*$ ($k^* > 0$), C^* is the optimal solution to the bipartition problem that minimizes $|F(\bar{U}, U)| - k^* \times |\vec{R}(\bar{U}, U)|$ (see [11] for proof).

Theorem 1 indicates that one can always find the MAAR cut (with the minimum friends-to-rejections ratio) by solving the above described family of problems for varying values of k . To approximate the aggregate friends-to-rejections ratio k^* of a MAAR cut, we iterate k through a geometric sequence, and pick the one that yields the lowest aggregate ratio.

KL Extension. We now describe how the KL approach can be extended to solve the partition problem minimizing $|F(\bar{U}, U)| - k \times |\vec{R}(\bar{U}, U)|$, given a value for k . Our approach is to unify the friendship edges and the rejection edges by assigning them different weights according to the objective function, i.e., weight 1 and weight $-k$ for friendship and rejection, respectively. This conversion translates the objective into minimizing the sum of weight of the cross-region edges. Thus, it enables an extension of KL with edge weights to solve the problem. Algorithm 1 shows the pseudocode of our KL extension. Unlike in the graph partitioning problem that the original KL copes with, we do not assume prior knowledge on the part sizes of a partition. Thus, we replace the node-part

interchanging in KL with single-node switching to allow the change of part sizes. The bucket list dynamically indexes each node based on its potential gain of switching to the other part of the partition. To ensure a desirable approximation of the optimal cut with a ratio k^* , we iterate k through a geometric sequence with a certain scale factor.

E. Iteratively detecting fake-account groups

Fake accounts can form multiple independent groups. These groups may have few OSN links and rejections between them (Figure 5). In such cases, a single run of the extended KL algorithm catches only a few groups if the aggregate acceptance rates of requests from the rest of the fake groups are not equally low. To make it worse, a group of fake accounts can attempt to mimick real users by rejecting accounts from other groups controlled by the same attacker, i.e., by employing self-rejection. In this way, attackers can whitewash a portion of their accounts by fabricating a cut within the collective of their fake accounts and making this cut's friends-to-rejections ratio even lower than the global friends-to-rejections ratio between the legitimate and the fake accounts, as shown in Figure 7. As a result, a scheme simply seeking the MAAR cut can miss the group of the rejecting fake accounts.

To address this challenge, Rejecto iteratively runs the extended KL to identify and cut off spammer groups from the graph until it has detected as many friend spammers as the OSN has estimated that exist (by inspecting sampled users [7], [14]). Attackers cannot change the friends-to-rejections ratio of the global spammer/legitimate cut, but they can craft a cut with a ratio lower than that within their fake accounts. In such cases, Rejecto first finds the lower ratio cuts within the fake accounts, and detects the part of the fake accounts that receive a large number of rejections due to the crafted low ratio. It then prunes these accounts with their links and rejections from the graph, and continues to find the next lowest friends-to-rejections ratio cut. This process repeats until we find the global spammer/legitimate cut.

Other termination conditions. In practice, besides the number of the estimated fake accounts, other conditions are also available to decide on the proper termination of the iterative cut detection procedure. Because we detect the MAAR cut of a residual graph in each round, multiple rounds of MAAR cut detection yield an ordered list of suspicious-account groups, among which those with a low aggregate request acceptance rate are uncovered first. As a result, an OSN provider can set an appropriate aggregate acceptance rate threshold (e.g., an estimate of the acceptance rate of normal users) and use it to terminate the iterative detection procedure.

F. Reducing false positives

The extended KL approach approximates the MAAR cut by interchanging nodes. Due to the large search space in a social graph, it might be stuck in a local minimum ratio cut, if that cut ratio is small. Although such small ratio cuts are rare among legitimate users, false positives can be introduced if one of them is mistakenly detected as a valid MAAR cut.

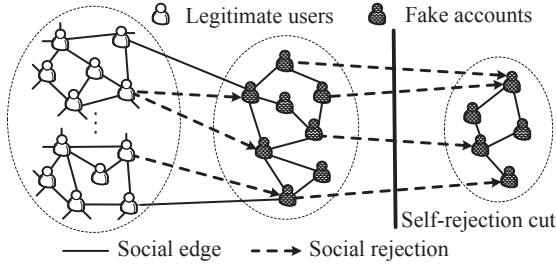


Fig. 7: Attackers whitewash a part of fake accounts by making them reject others as legitimate users do. In this illustration, the attacker attempts to whitewash the group of fake accounts in the middle.

To reduce the false positives, we incorporate OSN providers’ prior knowledge into the KL scheme. An OSN provider can determine whether a few selected users are fake accounts or legitimate through manual inspection. These users are randomly selected and we call them *seeds*. We use this knowledge to guide the KL partitioning to avoid cut mistakes.

Our idea is to pre-place each seed into its corresponding spammer or legitimate region and never switch it during the KL cut search. By distributing seeds over the entire graph, Rejecto can effectively rule out those problematic legitimate-user cuts from its search space. In particular, the undesirable partitions derived from such small cuts within the legitimate region misclassify portions of legitimate users as fake accounts, which, with high probability, conflicts with the pre-placement of the seeds. Therefore, as seeds never switch regions, Rejecto avoids the problematic cuts. To ensure sufficient seed coverage, one could employ the community-based seed selection as in SybilRank [14].

G. Implementation

We have implemented a Rejecto prototype on Spark [36], a generic in-memory large-data processing platform with automated fault tolerance. To obtain a scalable and efficient implementation, we distribute the large social graph structure to the workers, while keeping only a tractable set of algorithm variables and states on the master. This data layout ensures that Rejecto can handle large OSN data sets by scaling the capacity of the workers, and that the master can efficiently schedule the Rejecto algorithm according to the algorithm states kept on it. Using Spark’s key feature, we cache intermediate data sets and results in memory, reducing the disk I/O cost of their future reuse by Rejecto. Furthermore, we employ prefetching to substantially reduce the network I/O. For a more detailed description of the implementation, we refer the reader to [11].

V. EVALUATION

In this section, we evaluate Rejecto’s effectiveness and compare it to VoteTrust, which ranks users on the friend request graph to detect fake accounts [32]. We choose VoteTrust for comparison because it also uses rejections of friend requests. We assess the two systems under: a) the flooding of spam requests; b) the collusion and self-rejection evasion strategies;

and c) the rejection of legitimate requests by spammers. Moreover, we demonstrate the plausibility of combining Rejecto and social-graph-based approaches to form a defense in depth against fake accounts. Lastly, we evaluate the computational efficiency of our parallel implementation.

VoteTrust. It generates a user ranking for fake account detection via two steps. First, it uses a PageRank-like algorithm to assign a trust value for each user, called votes, using the directed friend request graph. Second, it generates a rating for each user that depends on the responses that his requests received, which is called vote aggregation. The rating of a user is a weighted average of 1s (accepted requests) and 0s (rejected requests). The weight of a request is the number of votes of the user that the request goes to times the current rating of that user. This rating computation takes place iteratively.

VoteTrust has two major weaknesses. First, the vote aggregation relies heavily on the request acceptance rate of each individual user, hence it is not resilient to collusion attacks (§V-C). Second, it assumes that because the fake accounts have few incoming friend requests, the votes of a fake account assigned by a PageRank-like algorithm are low. However, attackers can manipulate PageRank values if they control accounts to send requests to each other [16].

TABLE I: Social graphs used in the simulation.

Social Network	Nodes	Edges	Clustering Coefficient	Diameter
Facebook	10,000	40,013	0.2332	17
ca-HepTh	9,877	25,985	0.2734	18
ca-AstroPh	18,772	198,080	0.3158	14
email-Enron	33,696	180,811	0.0848	13
soc-Epinions	75,877	405,739	0.0655	15
soc-Slashdot	82,168	504,230	0.0240	13
Synthetic	10,000	39,399	0.0018	7

A. Simulation setup

Data sets. We simulate friend spam on a Facebook sample graph, five public social graph data sets [4], and a synthetic graph, as shown in Table I. The Facebook graph is a sample graph we obtained on Facebook via the “forest fire” sampling method [26]. The synthetic graph is generated based on the scale-free model [13]. Our analysis focuses on the representative results on the Facebook graph. The results on other graphs are similar. We refer the reader to [11] for a complete report.

On each social graph, we add a large spamming region with 10K fake accounts for stress-test. Upon the arrival of each fake account, it connects to 6 other fake accounts. Scenarios with dense connections among fake accounts are evaluated in §V-C.

Metric. To detect suspicious accounts, Rejecto iteratively cuts off identified accounts until their number reaches 10K (as many as the injected fake accounts). With VoteTrust, we designate the 10K users with the lowest rating as suspicious. The above assumes that the OSN provider has an estimation of the total number of fake accounts in the system. In an OSN deployment, Rejecto can use fine-tuned termination

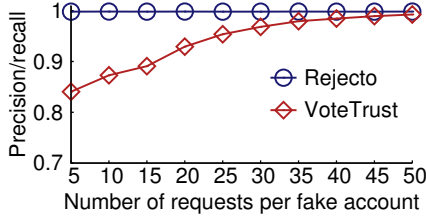


Fig. 8: Precision/recall as a function of the number of requests per fake account, when we let all fake accounts send friend spam.

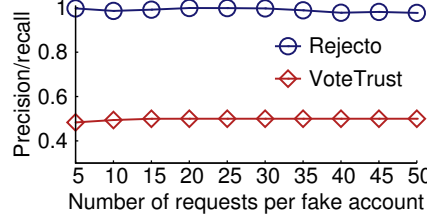


Fig. 9: Precision/recall as a function of the number of requests per fake account, when half of the fake accounts send friend spam.

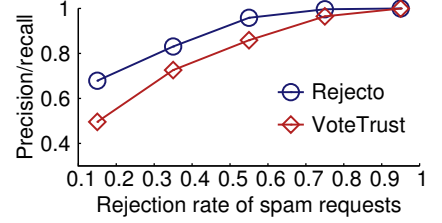


Fig. 10: Precision/recall as a function of the rejection rate of spam requests.

conditions, as described in §IV-E. We compare Rejecto against VoteTrust based on the *precision*: the ratio of the number of the detected fake accounts to the number of declared suspicious users. Since we let each scheme declare the same number of suspicious accounts as that of the fake accounts we have injected, the precision is identical to the *recall* [29]. A high value of precision/recall indicates high detection accuracy.

Simulating rejections. Based on a previous measurement study in RenRen [33], we set the rejection rate to 70% for fake accounts, and to 20% for legitimate users. We set scenarios to study the sensitivity to those rejection rates in §V-B. We let each fake account send out 20 requests to legitimate users. The fraction of these requests that are rejected is the rejection rate. This is a moderate baseline attack setting. As will be shown (§V-B), sending out more requests exposes more fake accounts to the detection systems. To simulate rejections among legitimate users, we compute the number of a user’s rejections by taking into account the rejection rate and the number of friends he has on the social graph.

B. Sensitivity analysis

Impact of the spam request volume. We simulate flooding attacks, where attackers flood friend requests to legitimate users. To this end, we increase the number of requests that each fake account sends out, ranging from 5 to 50, with a fixed rejection rate 70%. Figure 8 shows that Rejecto retains a high detection accuracy regardless of the number of requests per fake account. In contrast, VoteTrust’s accuracy is low when the number of requests is small, and improves as the number of requests increases. This is because VoteTrust does not directly target the aggregate acceptance rate of the requests from fake accounts. Instead, it is sensitive to the volume of the requests due to its use of the PageRank-like vote assignment, i.e. the fake accounts get less votes if they send out more requests.

Attackers may attempt to protect some of their accounts from getting rejections. To this end, they use only a portion of their accounts to sent friend requests to legitimate users. At the same time, to avoid being isolated, the rest of their accounts connect to the spamming accounts. To assess our system under this strategy, we simulate attacks in which only half of the fake accounts flood friend requests. As shown in Figure 9, Rejecto withstands this attack because of the unchanged high rejection rate of the spam requests. Rejecto does not place the non-request-sending fake accounts into the legitimate region,

because doing so would yield a higher acceptance ratio as these accounts are connected to the fake accounts that send out spam requests. VoteTrust is as low as 50%, indicating that it misses almost all the fake accounts that do not send friend requests. This is because VoteTrust’s vote aggregation relies on the request acceptance rate of individual users. It missed the fake accounts behind the spamming accounts.

Sensitivity to the rejection rate of spam requests. We examine the impact of the rejection rate of spam requests by varying its value from 0.5 to 0.95. Figure 10 shows that the increasing rejection rate of spam requests results in improved accuracy for both Rejecto and VoteTrust. Rejecto and VoteTrust do not achieve high detection accuracy when the rejection rate is small. Besides, Rejecto can detect almost all of the fake accounts if the rejection rate of their requests is close to 60%.

Sensitivity to the rejection rate of legitimate requests. We increase the rejection rate of legitimate requests from 0.05 to 0.95, while fixing that of spam requests to 0.7. As shown in figure 11, the accuracy of Rejecto and VoteTrust degrades with the increase of the rejection rate of legitimate requests. This is because the gap of the request rejection rate between fake and legitimate users shrinks, which makes them less distinguishable.

C. Resilience to collusion and self-rejection

We evaluate the resilience of Rejecto to the collusion and self-rejection detection evasion strategies.

Collusion between fake accounts. We evaluate Rejecto’s resilience to the collusion strategy (§III-A) by varying the number of per-account accepted friend requests between fake accounts from 4 to 40. Figure 12 shows that Rejecto’s accuracy remains high as the average rejection rate of a fake account drops from 70% to 23%. This is because the edges between colluders do not change the aggregate acceptance rate of their requests to legitimate users. In contrast, VoteTrust’s accuracy degrades as the edges among spammers become denser, because it relies heavily on the number of requests from and the rejections to individual users.

Self-rejection within fake accounts. We evaluate Rejecto under the self-rejection strategy (§IV-E). In this simulation, attackers attempt to whitewash 5K fake accounts and disguise them as legitimate users by rejecting other fake accounts. We

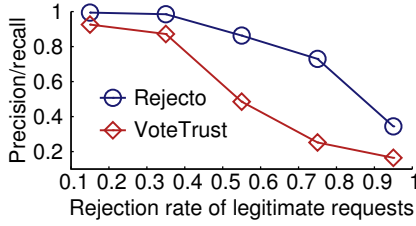


Fig. 11: Precision/recall as a function of the rejection rate of the friend requests among legitimate users.

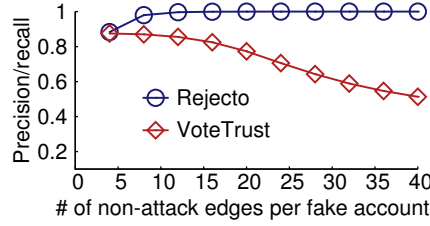


Fig. 12: Resilience to the collusion strategy of forming dense social connections among fake accounts.

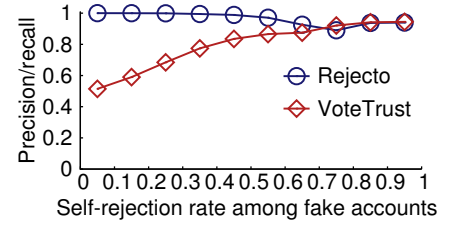


Fig. 13: Resilience to the self-rejection strategy, which whitewashes a portion of fake accounts.

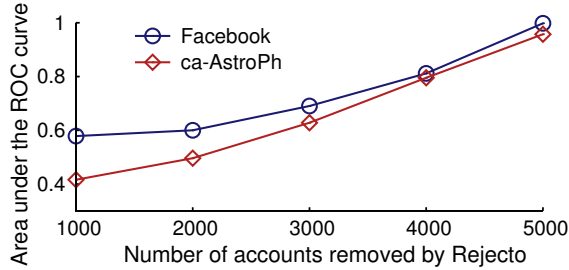


Fig. 14: Ranking quality of SybilRank as a function of the number of suspicious users removed by Rejecto.

let each of the other $5K$ fake accounts send 20 requests to the fake accounts to be whitewashed. We increase the rejection rate of these requests from 0.05 to 0.95. We can see in Figure 13 that the accuracy of Rejecto remains high, except for the slight degradation when the rejection rate is close to 0.7, which is the rejection rate of requests from fake to legitimate users. This is because the ratio of the spammer/legitimate cut is too close to that of the cut within the fake account region, such that our algorithm is not able to precisely pinpoint it. When the self-rejection rate is larger than 0.7, Rejecto repeatedly solves MAAR, which detects the $5K$ request-sending fake accounts first, and then the whitewashed accounts. Unsurprisingly, we can also see that the self-rejection strategy is counterproductive against VoteTrust. We refer to [11] for simulations that show the system’s resilience to the strategic rejection of requests that careless legitimate users send to fake accounts.

D. Defense in depth with social-graph-based Sybil detection

Rejecto can be used in combination with social-graph-based schemes to systematically detect Sybils (§II-B). We choose SybilRank [14] as a representative social-graph-based detection scheme, and run it on social graphs after removing a number of accounts that Rejecto declares as friend spammers. We use the value of area under the ROC curve [14] to measure the quality of the ranking of SybilRank. We add a $10K$ -Sybil set, among which $5K$ Sybils send out 20 spam requests, each at a rejection rate of 70%. Figure 14 shows the improvement of SybilRank’s accuracy on our Facebook sample graph and the ca-AstroPh social graph [4], as the number of users that are removed by Rejecto increases. This improvement is because our system detects spamming accounts whose removal can reduce a significant fraction of attack edges in the social

graph. If we let Rejecto remove $5K$ suspicious users, the area under the ROC curve of SybilRank’s ranked list is close to 1, indicating that SybilRank detects most of the rest of the Sybils by ranking them to the bottom of its ranked list.

E. Computational cost on large graphs

We deployed our Rejecto prototype on a EC2 cluster running Spark 0.9.2 with 5 computational nodes (1 master and 4 workers), each equipped with 60 GB RAM. We generated synthetic input graphs with size varying from $500K$ to $10M$ users. For graph size $500K$, $1M$, $2M$, $5M$ and $10M$, the execution time is 288, 669, 1767, 8049 and 27673 sec, respectively (Table II). These results indicate that the execution time grows gracefully with the graph size, i.e., close to $\Theta(n \log n)$. Because KL is a nearly linear-time algorithm [19], we anticipate that on a cluster with sufficient capacity (e.g., 100 worker nodes each with 40-60GB RAM and a master node with $\sim 200GB$ RAM), Rejecto can process social graphs with hundreds of millions of users.

TABLE II: Rejecto’s execution time on large graphs of different sizes

#users	0.5M	1M	2M	5M	10M
#edges	$\sim 8M$	$\sim 16M$	$\sim 32M$	$\sim 80M$	$\sim 160M$
Time	288 sec	669 sec	1767 sec	8049 sec	7.7 hours

VI. DISCUSSION

Responses to the detected accounts. To prevent detected accounts from sending out friend spam in the future, an OSN provider can take actions, such as sending CAPTCHA challenges, rate-limiting their online activities, or even suspending the accounts [14], [30]. The actions taken before account suspension allow certain degree of tolerance to the false positives (e.g., OSN creepers [30]) in the detection system.

Application to the detection of other malicious accounts. OSN providers can apply Rejecto to the detection of other malicious accounts, such as compromised ones. If compromised accounts are manipulated to pollute the social graph via friend spam, their requests follow Rejecto’s friend spam model. Therefore, they are exposed to Rejecto’s detection. In such a deployment scenario, the OSN provider can shard friend requests and rejections according to the time intervals in which they have occurred, and then run Rejecto on an augmented graph constructed from the sharded requests and rejections

in each interval. This enables Rejecto to detect compromised accounts in post-compromise intervals.

VII. RELATED WORK

Our work is mostly related to VoteTrust [32], which we summarize and experimentally compare to in Section V. Rejecto is also related to social-graph-based Sybil defenses [14], [17], [34]. These proposals rely on social graph properties to distinguish Sybils from non-Sybil users, i.e., that the number of edges connecting Sybil with non-Sybil nodes is relatively small. But, fake accounts benefit from soliciting OSN links via spammy friend requests because a non-negligible portion of users does not scrutinize, but accept them. Rejecto complements the aforementioned defenses. It cleans the social graph of the fake accounts that have obtained social edges, but in the process also have received many social rejections.

There is a plethora of studies on trust propagation in signed social networks [18], [21], [24], [37], where a social network contains both positive and negative edges. Unlike Rejecto, they do not use the aggregate acceptance rate, thus they are susceptible to manipulation. Moreover, they consider negative votes and ratings that malicious users can arbitrarily cast. As a result, they are not resilient to user distortion.

Recent work in signed social networks studies the “structure balance” [27]. The *structural balance theory* examines the signs of the edges of each interconnected three nodes, called a *triad*. A triad is balanced if the signs of edges respect the principles in social psychology, such as “the friend of my friend is my friend” and “the enemy of my enemy is my friend”. Nevertheless, it is unclear how the structure balance theory could be used to detect friend spam.

VIII. CONCLUSION

We contribute to the fight against fake accounts (Sybils) and their abusive friending activity in Facebook-like symmetric OSNs. We designed and implemented Rejecto, a system that detects fake accounts sending out unwanted friend requests. Rejecto augments the social graph with social rejections, and seeks the minimum aggregate acceptance rate cut. With this formulation, our system is able to uncover friend spammers in a manipulation-resistant way. Our evaluation results show that Rejecto is effective in a broad range of scenarios, resilient to attack strategies, and computationally practical.

IX. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback. Qiang Cao and Xiaowei Yang are supported by NSF awards CNS-1017858, CNS-1345283, and CNS-0845858. Kamesh Munagala is supported by NSF awards CCF-0745761, CCF-1008065, CCF-1348696, and IIS-1447554, and by ARO grant W911NF-14-1-0366.

REFERENCES

- [1] BlackHatWorld. <http://www.blackhatworld.com/>.
- [2] Freelancer. <http://www.freelancer.com/>.
- [3] Question: Requests from unknown people. <http://tinyurl.com/phrdm5y>.
- [4] Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data/index.html>.
- [5] Under the Hood: Building Graph Search Beta. <http://tinyurl.com/byjfd9f>.
- [6] Do I Know You? Fake Friends Adding Fresh Danger To Facebook. <http://tinyurl.com/pma2e23>, 2011.
- [7] Facebook estimates fake accounts. <http://tinyurl.com/qgdlekx>, 2012.
- [8] Graph Search powered by Bing. <http://tinyurl.com/bnkvxv>, 2013.
- [9] How do I stop strangers from sending me friend requests? <http://tinyurl.com/qhc4u6f>, 2014.
- [10] Keeping Facebook Activity Authentic. <https://m.facebook.com/business/news/authentic-activity-on-facebook>, 2014.
- [11] Combating Friend Spam Using Social Rejections. Technical report, http://www.cs.duke.edu/~qiangcao/rejecto/rejecto_tr.pdf.
- [12] Y. Aumann and Y. Rabani. An $O(\log k)$ Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm. *SIAM Journal on Computing*, 27(1), 1998.
- [13] A.-L. Bárábási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.
- [14] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the Detection of Fake Accounts in Large Scale Social Online Services. In *NSDI*, 2012.
- [15] Q. Cao and X. Yang. SybilFence: Improving Social-Graph-Based Sybil Defenses with User Negative Feedback. Technical Report, Computer Science Dept., Duke University, 2012. <http://arxiv.org/abs/1304.3819>.
- [16] A. Cheng and E. Friedman. Manipulability of PageRank under Sybil Strategies. In *NetEcon*, 2006.
- [17] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.
- [18] C. de Kerchove and P. van Dooren. The PageTrust Algorithm: How to Rank Web Pages When Negative Links are Allowed? In *SDM*, 2008.
- [19] C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *Design Automation Conference*, 1982.
- [20] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [21] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *WWW*, 2004.
- [22] T. Kendall and D. Zhou. Leveraging Information in a Social Network for Inferential Targeting of Advertisements. US Patent App. 12/419,958, 2010.
- [23] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, 49, 1970.
- [24] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The Slashdot Zoo: Mining a Social Network with Negative Edges. In *WWW*, 2009.
- [25] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6), 1999.
- [26] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *ACM SIGKDD*, 2006.
- [27] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *ACM SIGCHI*, 2010.
- [28] A. Madry. Fast Approximation Algorithms for Cut-Based Problems in Undirected Graphs. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS, 2010.
- [29] D. M. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1), 2011.
- [30] T. Stein, E. Chen, and K. Mangla. Facebook Immune System. In *Proceedings of the 4th Workshop on Social Network Systems*, 2011.
- [31] M. Trusov, A. V. Bodapati, R. E. Bucklin, and W. Mullin. Determining Influential Users in Internet Social Networks. In *AMA Journal of Marketing Research*, 2010.
- [32] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, and Y. Dai. VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils. In *IEEE INFOCOM*, 2013.
- [33] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering Social Network Sybils in the Wild. In *IMC*, 2011.
- [34] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense Against Sybil Attacks. In *IEEE S&P*, 2008.
- [35] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *SIGCOMM'06*.
- [36] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-Memory Cluster Computing. In *NSDI*, 2012.
- [37] C.-N. Ziegler and G. Lausen. Propagation Models for Trust and Distrust in Social Networks. *Information Systems Frontiers*, 7, 2005.