DLSAS: Distributed Large-Scale Anti-Spam Framework for Decentralized Online Social Networks

Amira Soliman, Sarunas Girdzijauskas

School of Information and Communication Technology
KTH Royal Institute of Technology, Stockholm, Sweden
{aaeh, sarunasg}@kth.se

***Abstract*—In the last decade, researchers and the open source community have proposed various Decentralized Online Social Networks (DOSNs) that remove dependency on centralized online social network providers to preserve user privacy. However, transitioning from centralized to decentralized environment creates various new set of problems, such as adversarial manipulations. In this paper, we present DLSAS, a novel unsupervised and decentralized anti-spam framework for DOSNs. DLSAS provides decentralized spam detection that is resilient to adversarial attacks. DLSAS typifies massively parallel frameworks and exploits fully decentralized learning and cooperative approaches. Furthermore, DLSAS provides a novel defense mechanism for DOSNs to prevent malicious nodes participating in the system by creating a validation overlay to assess the credibility of the exchanged information among the participating nodes and exclude the misbehaving nodes from the system. Extensive experiments using Twitter datasets confirm not only the DLSAS's capability to detect spam with higher accuracy compared to state-of-the-art approaches, but also the DLSAS's robustness against different adversarial attacks.**

Keywords. Decentralized Online Social Networks, Spam Detection, Distributed Systems, System Integrity and Robustness.

I. Introduction

Online Social Networks (OSNs) represent popular collaborative communication tools for billions of Internet users. For example, Facebook alone has over 1,7 billion users[1] and lies in the third place among the most visited sites on the Internet[2]. However, with the adopted client-server architecture in all current popular OSNs, many privacy and ownership issues appear [1]. In particular, while OSN users generate the data, the OSN providers take over full control over it, in effect becoming centralized "big-brother" authorities. Thus, such centralized architecture has critical consequences such as the necessity for a high degree of trust in the OSN providers, censorship of users behavior and the utilization of user data for business-related purposes [2, 3].

To address the aforementioned problems, in the last decade, researchers and the open source community have proposed various decentralized OSNs (DOSNs) (e.g.,[4, 5]) that remove dependency on a centralized provider. DOSNs operate as distributed information management platforms on top of network of servers or P2P infrastructures [6]. The main objectives behind decentralization are to preserve users privacy in both shared content and communication, and also to provide complete freedom from any form of censorship or profiling. Thus, DOSNs provide a privacy preserving alternative to current OSNs, where users have full control of their data.

Meanwhile, all types of social networks are increasingly used as up-to-the minute information source for public issues such as economy, politics, and crisis. Yet, spam in these networks is explosively increasing and has become an effective vehicle for malware and illegal advertisement distributions. Spam content not only questions the credibility of shared information, but also misleads or even traps legitimate users, resulting in bad user experiences. In the past years, researchers developed approaches to detect spam such as URL blacklisting, spam traps and even crowdsourcing for manual classification [7, 8, 9, 10]. Although previous work has shown the effectiveness of using statistical learning to detect spam [11, 12], there are none for open decentralized environment as DOSNs. Existing work assumes controlled centralized settings as well as employs supervised schemes that require labeled training data that is difficult to obtain in DOSNs.

More recently, in our previous work [13] we proposed unsupervised spam detection scheme with centralized settings. Differently from existing work, our approach constructs a user similarity graph that encodes within its topology a holistic view of all behavioral interactions and patterns among users. Afterwards, our approach performs graph clustering by applying community detection on top of the newly created graph. In particular, the detected communities on top of user similarity graph identify different behavioral patterns existing in the social network. This allows our system to categorize the existing behavioral patterns into more homogeneous and accurate clusters than the sate-of-the-art approaches.

Although, our graph-based approach achieves better results compared to the current state-of-the-art in spam detection, in its current form it is not possible to adopt for DOSNs due to several key challenges. Specifically, the incentive for defeating spam detection in decentralized environments increases due to

[1] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[2] https://en.wikipedia.org/wiki/List_of_most_popular_websites

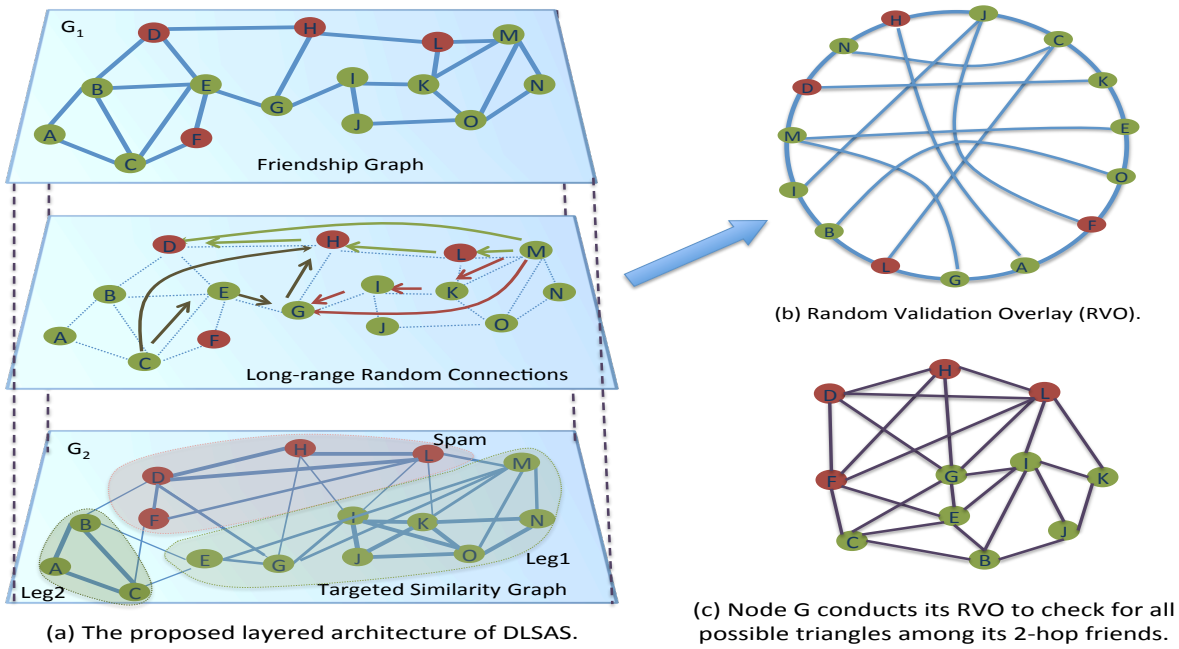


Fig. 1: DLSAS overall process using a social network example. Green nodes represent legitimate users, whereas red nodes represent spammers. (a) The layered architecture of DLSAS, using the social friendship graph to create long-range connections among nodes in order to reach the targeted similarity graph for clustering. (b) The constructed RVO after exchanging long-range random connection among nodes . (c) Nodes A, L, and M check for similarity edges created around node G by finding all possible triangles among G's 2-hop neighbors.

distributed nature of the system and non-avoidable cooperation among participating nodes. For example, malicious parties can destroy communication pathways, prevent the system from stabilizing, equivocate by giving different information to different nodes, and provide false provenance information. Therefore, our graph-based approach requires a defense mechanism to detect any data manipulation during the phases of similarity graph creation and community detection.

To address these issues, we propose a decentralized framework DLSAS (Distributed Large-Scale Anti-Spam) which allows our effective centralized spam-detection method from [13] to be applicable for decentralized environment. Our objective is to preserve the core of spam detection (i.e., similarity graph creation and community detection) from manipulation under the existence of adversarial nodes that deviate from the designed workflow protocol. Therefore, secure cooperation can be achieved when misbehaving or deviating nodes are detected and disconnected from the system. Thus, the exchanged data among nodes requires verification mechanism that is able to find a proof that a node deviated from the prescribed protocol and participated in adversarial activities. Accordingly, in DLSAS every node is required to keep a communication log that records all communication events. We assume the existence of a public key infrastructure, allowing any node to digitally sign its communication log such that any other node can verify it, yet making it computationally infeasible for others to forge.

Consequently, any misbehaving node can be detected by comparing its manipulated log with the logs of the other nodes involved in communication. However, malicious nodes can collude with each other to evade the validation and detection mechanism. Therefore, every validation task has to be assigned randomly in order to prevent adversaries from inferring any details about the participating nodes. Accordingly, DLSAS creates on-the-fly Random Validation Overlay (RVO) to validate the data reported by the participating nodes (as shown in Figure 1). RVO verifies the credibility of exchanged data among nodes, and it allows the system to detect false and inaccurate data provenance. Consequently, RVO enforces every node to report its local results without manipulation, otherwise it is excluded from the system. As shown in Figure 1, RVO is created using a random peer sampling service that creates uniformly random overlay on top of social graph by allowing nodes to have long-range connections to random nodes. Thereafter, RVO is responsible for verifying the correctness of exchanged data during similarity graph creation and clustering phases.

Accordingly, our work offers the following contributions to the problem of decentralized anti-spam frameworks:

- We propose a fully decentralized spam detection framework, where each node independently processes its local data and is required to know only its direct neighbors,
- We propose a uniformly random validation mechanism that is capable of detecting and disconnecting misbehaving nodes from the system,
- Our proposed validation mechanism is fully decentralized and is capable of maintaining DOSNs robustness and integrity against adversarial attacks.

We have performed experiments, using Twitter datasets, to show the effectiveness of our proposed framework. The results show that DLSAS outperforms the state-of-the-art centralized techniques and provides more accurate spam detection rate with accuracy up to 92.3% and false positive rate less than 0.3%. Most importantly, DLSAS ensures spam-detection integrity and reliability by detecting at least 94.7% of adversarial data manipulation, under the condition that 20% of nodes are participating in adversarial attacks.

The remainder of this paper is structured as follows. In Section II we detail the key challenges for graph-based spam detection in DOSNs, whereas, in Section III we illustrate the algorithms used to develop DLSAS validation mechanism. In Section IV we present evaluation of our framework, as well as, we discuss our framework robustness against possible adversarial attempts. Finally we show the related work in Section V and conclude the paper in Section VI.

II. Decentralized Spam Detection

Classification tasks such as spam detection, intrusion detection, and anomaly detection, are being gamed by an adversary who wishes to avoid detection. In this section we identify the potential vulnerabilities arising due to decentralization.

A. User Similarity Graph

In general, the design of spam detection mechanisms is guided by the behavior dissimilarity exhibited by legitimate users than spammers. The central premise as proved in spam detection approaches is that spammer behavior appears anomalous relative to normal user behavior along some features that can be extracted from different users activities. For example, some content-based features can be extracted from user posts such as number of URLs, hashtags and mentions used per post, as well as graph-based features that can be calculated from the friendship graph such as local clustering coefficient and betweenness centrality. Therefore, we define the following:

Definition 2.1: **Friendship Graph**. We consider DOSN as a directed unweighted graph G_1 = (V, E_1), where V is the set of nodes and E is the set of edges. $e_{ij} \in E_1$ denotes a relationship and communication link between nodes v_i and $v_j \in V$.

Furthermore, we consider every node is associated with a feature vector that contains values of content-based and graph-based features computed for that node. Initially, every node uses its local data repository that contains the collection of its direct friends posts and calculates the equivalent feature vector of each neighbor. Afterwards, every node starts the process of similarity graph creation as follows

Definition 2.2: **Similarity Graph**. We consider user similarity graph as weighted directed graph G_2 = (V, E_2), where V is the set of nodes and E_2 is the set of similarity edges, where $e_{ij} \in E_2$ denotes cosine similarity between nodes v_i and $v_j \in V$ that is computed as defined in Equation 1.

$$w(e_{ij}) = \frac{x_i.x_j}{||x_i||.||x_j||} \tag{1}$$

Where, x_i is the feature vector of node *i* and x_j is the feature vector of node *j*. If the weight $w(e_{ij})$ is greater than the threshold ϵ, $w(e_{ij})$= cosine similarity; else $w(e_{ij})$=0. (see Figure 1(a), the thickness of an edge reflects its weight).

Particularly, every node starts the process of G_2 construction from the friendship graph G_1. Initially, every node starts by creating similarity edges among itself and its social neighbors in G_1. Afterwards, every node enlarges the similarity graph further by exploring the possibility of creating more similarity edges with the neighbors of its currently direct neighbors. Once the user similarity graph is created, our objective is to find the topological communities inside the constructed similarity graph. The detected communities represents the different behavioral patterns, then the spam patterns can be identified among the detected ones by applying some lexical analysis. In particular, every node in the similarity graph starts by joining the node with the maximum cosine similarity among its direct friends to form a community. In successive iterations, every node chooses to quit its current community and join one of its neighbour's if this brings some modularity gains. This step is iteratively repeated until no node wants to change its community as it already represents the dominant one of all its neighbors. Thereafter, the topological communities detected in the similarity graph represent the different behavioral patterns associated with the direct friends of a user in the social graph. Additional information and details may be found in [13].

B. Vulnerabilities in Spam Detection for DOSNs

The transition from centralized to distributed environment creates vulnerability of spam detection approaches to adversarial manipulation. Node cooperation in DOSNs is imperative for forming services constituents and reaching the global system goals, yet such system can easily be manipulated. Particularly, DOSNs operate using distributed infrastructure similar to P2P systems, such that the data is fully distributed as nodes are only aware of the exchanged data with their direct neighbors. Furthermore, the communication in DOSNs is more restricted communication as it is allowed only among direct neighbors. To be aligned with the DOSN requirements, spam detection should be designed as node-centric and distributed learning approach such that the system allows to deal with data sets that are naturally distributed. Furthermore, the learning process must be mapped into a set of parallel tasks executed by every node till system reaches convergence. Due to these requirements, decentralized spam detection has two major limitations. The first limitation is that nodes have local knowledge represented in information they have for their direct friends. Secondly, nodes have to collaborate to reach a unified global knowledge of the whole system using their local knowledge. Consequently, the decentralized spam detection has following potential vulnerabilities: (i) equivocation by exchanging different information with different nodes, (ii) false provenance by exchanging falsely information to avoid detection, and (iii) colluding adversary.

Equivocation: In such scenario, spammers want to gain more exposure to legitimate users by increasing the number

of edges connecting them with legitimate clusters.

Example 1. Node *L* (see Figure 1(a)) sends out the information of H as its only neighbor while exchanging neighbors with legitimate nodes (*L* for instance). Whereas, when *L* performs neighbor exchange with *H*, *L* sends information of nodes *K* and *M*.

False Provenance: In this case, spammers falsify the weight of their added similarity edges, such that they increase the weight of edges connecting them to legitimate nodes as well as drop the similarity edges connecting them to other spammers.

Example 2. Node *L* increases the weight of the edges connecting *M*, *K*, and *I*. On the other hand, *L* drops out the similarity edges with *H* and *D* by assigning weight equals to 0 to those edges.

Colluding Adversary: In this scenario, spammers collaborate with each other to coordinate the exchanged information with legitimate nodes. Specifically, during the clustering phase, spammers assign themselves to the closest legitimate cluster.

Example 3. Nodes *L*, *H*, *D*, and *F* assign themselves to cluster *Leg1* instead of creating a separate cluster for themselves.

III. DLSAS Validation Mechanism

In this section, we present the core of validation mechanism implemented in DLSAS. First, we present on-the-fly gossip-based peer sampling service that is used for constructing Random Validation Overlay (RVO). Thereafter, we detail the validation mechanism performed by RVO.

A. Constructing RVO

Our objective is to provide validation mechanism that is hard for adversarial parties to infer any details about it. Therefore, the validation overlay needs to be constructed by obtaining a random sample of nodes from the entire social network. Additionally, validation tasks need to be assigned randomly to the those nodes. Accordingly, we develop a gossip-based random peer sampling service that creates purely random overlay on top of the social graph. This peer sampling service allows nodes to have long range connections to random nodes, hence provide every node in the DOSN with a random sample from the entire social graph.

As aforementioned, DOSNs operate using distributed infrastructure similar to P2P systems, but with more restricted communication as communication in DOSNs is often allowed only among direct neighbors. Therefore, we apply peer sampling where nodes periodically exchange small random subset of the identifiers of their direct friends and paths to reach them. Thus, after sufficient number rounds, nodes are going to have a random sample of nodes in the network and the routing paths towards them. The advantage of gossip-based sampling in our setting is that samples are available locally and without delay. Figure 1 depicts the process of creating RVO from the original social graph. For example, node *M* starts by asking its direct neighbors in the social graphs as shown in Figure 1 (a) (i.e., nodes *L*, *K*, *O*, and *N*) to exchange a subset of their neighbors. Then, *M* receives a reply from *L* containing the node *H* and the routing path towards *H* via the intermediate node *L*. Repeatedly, *M* updates the set of nodes that it can reach and constructed routing paths towards them. After some exchange rounds, *M* reaches node *G* following the routing path [*K*, *I*, *G*]. As illustrated in Figure 1 (b), every node ends by having long range links to random nodes in the constructed RVO.

More formally, each node maintains a fixed-sized cache of c entries (with typical value 20 or 50 entries). A cache entry contains identifier and routing path of another node in the community. Each node repeatedly initiates a neighbor exchange operation, by executing Algorithm 1. As shown, the algorithms consists of two procedures. The first procedure *GossipSampling* is the one responsible for constructing a random sample of the network. Every node maintains a local repository named RS to refer to some random nodes, and stores the paths to reach them. First, every node initializes its RS by inserting some random nodes of its direct neighbors and identify itself as the route to reach those nodes. Afterwards, every node periodically selects a random partner from its RS for the gossip exchange and selects a random subset entries from its RS to be send in the gossip message. On receiving a reply form the contacted node during the gossip exchange, the receiving node updates its RS by adding the entries of the new nodes that are not included in RS as described in procedure *OnReceivedRS*.

Algorithm 1: Peer Sampling Service at node v_i

```
Result: Ensure random sample of network nodes: RS_i
Procedure GossipSampling()
    Loop
        wait(△)
        RS ← select_random_node()
        RCE ← select_random_cache_entries()
        send(RS, RCE)
    EndLoop
Procedure OnReceivedRS(message m)
    forall the e ∈ m.RCE do
        if new_entry(e) then
            add_new_entries (RS_i, e)
        end
    end
```

B. Adversarial Detection with RVO

Our objective is to provide secure cooperation in adversarial settings by maintaining the system integrity and reliability under the existence of adversarial nodes that deviate from the designed workflow protocol. Therefore, DLSAS requires every node to keep a communication log that records all communication events with others (both sent and received) within a given time window of fixed size that ends with sending the log for validation. As aforementioned, we assume the existence of a public key infrastructure, allowing any node to digitally sign its communication log such that any other node can verify it.

In the following, we show how the validation process prevents previously mentioned vulnerabilities (see Section II-B).

Equivocation: To guarantee the correctness of the constructed RVO, nodes periodically preform auditing process on nodes communication logs. For every entry in the node communication log, there should exist an entry for that in another node communication log. An auditor could fetch the communication log from some node *A* and then connect to every node mentioned in communication log of *A* to test for matching entries. This would detect any inconsistencies performed by node *A*, hence, equivocation scenarios similar to Example 1 can be detected. However, node A could collude with other nodes to push falsely subsets doing the execution of *GossipSampling* procedure. To fully audit A, the auditor would need to audit the nodes reachable from communication log of A, and recursively audit the nodes reachable from those logs. Eventually, the audit would discover misbehaving node where the logs are not matched. Implementing such a recursive audit would be prohibitively expensive. Instead, we require all nodes in the system to perform random auditing. In particular, each node should choose a node at random from the contacted ones. The auditor fetches the communication log, and verifies it against the nodes mentioned in that log. Assuming all nodes perform these random audits on a regular schedule, every node is going be audited on a regular basis. Naturally, spammers are expected not to initiate the auditing process. However, they are going to be audited once legitimate nodes that spammers are contacted to initiate the audit process.

False Provenance: To guarantee the correctness of the constructed user similarity graph, nodes periodically conduct RVO to validate the new edges added by their current and new neighbors. In particular, every node initiates validation process conducting a random validators from RVO (i.e., from its neighbors in RVO). For example, as shown in Figure 1(c), node *G* selects random validators from its RVO neighbors nodes *A*, *L* and *M*, to validate the added similarity edges connecting node *G* with other nodes. Those selected subset of validators are responsible for collecting shared posts from list of 2-hop neighbors of *G* and generating the feature vectors of all nodes in this list. Noting that, *G* randomly partition the calculations of feature vectors values among the selected validator, making it hard for spammers to infer any details of the validation process. Validators compute all possible triangles among G's 2-hop neighbors, in order to make sure that all possible similarity edges have been added to the similarity graph with the correct weights. Accordingly, any provenance violations similar to Example 2 can be detected, and adversarial nodes are excluded from RVO. As shown in Algorithm 2, the first procedure *validateEdges* is the one responsible for validating the newly added similarity edges. This procedure takes the list of its 2-hop neighbors and the edges reported by them as an input. Then, the procedure generates the feature vector values for all the nodes in the 2-hop neighbors list, and uses these vectors to verify if the reported edge weights are correct. Accordingly, if the reported weight returned by procedure *getWeight* is not the same as the computed cosine similarity using feature vectors, the reporting node is declared as a malicious node.

Algorithm 2: RVO Validation for node v

Result: Ensure creating similarity edges among 2-hop neighbors in $NList$

Procedure `ValidateEdges` (*neighbors* $NList$, *addedEgde* $EList$)

- **forall the** $u \in NList$ **do**
  - $FV_u \leftarrow CalcFeatureVector(u)$
  - **forall the** $w \in NList$ **do**
    - **if** $w \neq u$ **then**
      - $FV_w \leftarrow CalcFeatureVector(w)$
      - $CS_{uw} \leftarrow cosineSimilarity(FV_u, FV_w)$
      - **if** $getWeight(EList, u, w) \neq CS_{uw}$ **then**
        - declareMalicious(u)
      - **end**
      - **if** $getWeight(EList, w, u) \neq CS_{uw}$ **then**
        - declareMalicious(w)
      - **end**
    - **end**
  - **end**

Procedure `validateMembership` (*node* v, *community* c)

- $SoE \leftarrow$ collectEdges($Neighbor(v)$)
- $C_v \leftarrow$ selectCommunity(v, SoE)
- **if** $C_v \neq c$ **then**
  - declareMalicious(v)
- **end**

Colluding Adversary Attacks: Algorithm 3 illustrates the procedure used for clustering the generated similarity graph. As shown, every node iteratively chooses to quit its current community and join one of its neighbours if this bring some modularity gains. In method *selectCommunity*, nodes select the dominant community in their neighborhood to join (i.e., the community with the maximum sum of weights). This step is iteratively repeated until no node wants to change its community as it already represents the dominant one of all its neighbors. Spammers are expected to misbehave and pretend to be legitimate nodes by joining legitimate communities. Likewise, RVO validates the added similarity edges, RVO verifies community membership assigned by participating nodes. In particular, RVO validates the community membership using the knowledge of all possible triangles that can be created among node's 2-hop neighbors. Therefore, a node can not join a community while it is connected to that community members with weak edges (i.e., edges with low weights). Accordingly, spammers can not assign themselves to legitimate communities and they are forced to create their own community. Thus, colluding adversarial attacks during the clustering phase similar to Example 3 can be detected, and adversarial nodes will be labeled as spam and excluded from RVO. As described in method *validateMembership* in Algorithm 2, RVO verifies node's decision regarding its community membership. The

methods starts by collecting the set of edges (SoE) of the node, and use this set to compute the community membership of that node. Once the computed community membership is different from the computed one, the node is declared as a malicious node.

Algorithm 3: Community Detection Methods

Result: Community Structure C at time t
Procedure `changeCommunity` (*node* u)
  $C_{u_{new}} \leftarrow$ selectCommunity(u)
  if $C_u \neq C_{u_{new}}$ **then**
    $C_u \leftarrow C_{u_{new}}$
    forall the $x \in Neighbor(u)$ **do**
      changeCommunity(x)
    end
  end
  Procedure `selectCommunity` (*node* u)
    forall the $C \in NeighborCommunity(u)$ **do**
      $q(C) \leftarrow sum(w_{e_{uj}}) | C_j = C$
    end
    $C_u \leftarrow C_j | q(C_j) = max(q(C))$

IV. Evaluation

DLSAS applies vertex-centric approach which is proved to be scalable, efficient and most importantly can be applied for DOSNs. Our algorithms are implemented in GraphLab[3], with two different distributed execution modules, such that RVO validation algorithms are integrated in both of them. In the first module, nodes participate in creating the similarity graph using their feature vectors. Thereafter, the control is moved to the second module that performs the community detection algorithm. In the following subsections, we thoroughly evaluate the performance of our framework in terms of the accuracy of spam detection and robustness to adversarial manipulation. We compare our spam detection method with different centralized and supervised binary classification approaches, utilizing the Weka tool[4], namely: K-means (KM) with number of clusters =2, Decision Tree (DT) and Random Forest (RF).

A. Dataset

We have collected our dataset from Twitter using Twitter streaming API[5] during fourteen month period from May 2015 to July 2016. We have access Twitter API using privileged accounts, collecting user tweets as well as user friendship graph. Particularly, we have started by selecting random user identifiers for each dataset, then collect social graph surrounding these initial nodes. We have collected data with different levels of activities. For example, the first two datasets (US_Active and UK_Active) are collected from users with high level of posting tweets located in United States and United Kingdom, respectively. Yet, the third dataset (US_Passive) is collected from users located in United States with low level of posting activity. In order to identify the spammers within our datasets, we query the status of all accounts regularly to check if any got suspended for abusive behavior. Upon suspension, we identify suspended accounts as spammers. Table I lists the details of the collected datasets.

[3]https://turi.com/products/create/
[4]http://www.cs.waikato.ac.nz/ml/weka/
[5]https://dev.twitter.com/rest/public

TABLE I: Twitter datasets used in our experiments.

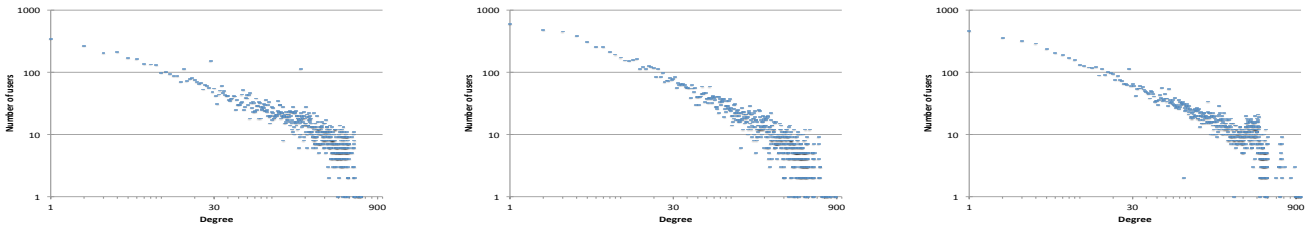
Twitter Dataset	US_Active	UK_Active	US_Passive
Tweets	453,519	489,484	360,927
Legitimate Accounts	17,322	19,312	12,128
Suspended Accounts	2,072	1,617	3,109
Social-graph Edges	1,357,806	1,187,036	2,349,314
Similarity-graph Edges	2,149,414	2,297,150	3,339,617

Figure 2 depicts the degree distribution for the collected datasets in log scale. As shown, the degree distribution follows the power law probability distribution, such that there is uneven distribution of node connections. Some nodes have very high degrees of connectivity (i.e., hubs), while most have small degrees.

B. Degree Distribution in RVO and User Similarity Graph

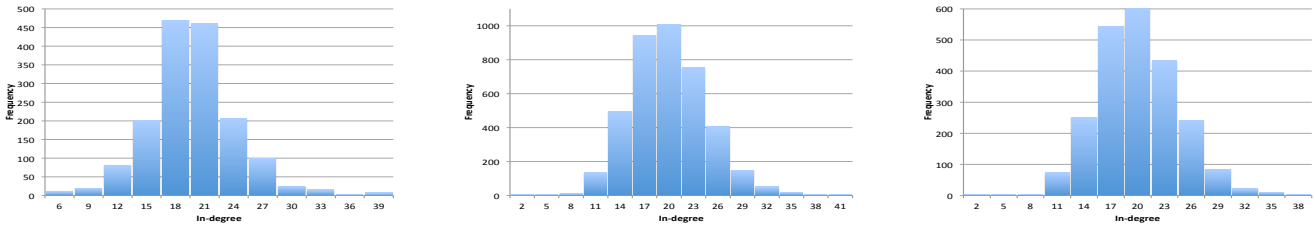
The degree of a node in RVO is the number of links it has to other nodes. The interest in the degree distribution stems from the assurance of fair link distribution among the participating nodes in joining the created RVO. We distinguish between the out-degree and the in-degree of a node, which are the number of edges leaving from and ending at the node, respectively. In our case, the out-degree of every node is fixed, and equal to the cache size. Therefore, we concentrate on observing the in-degree distribution of the constructed RVO. In our experiments we set the cache size to 20 entries and repeat each experiment for three times. Figure 3 shows the in-degree distribution for the constructed ROVs for each dataset. As shown, the peak of the in-degree equal to the cache size, while the number of nodes having larger or smaller in-degrees drops symmetrically from the cache size. Thus, in-degree distribution follows uniform distribution, hence, our gossip-based peer sampling service succeeded in created RVO with fair link distribution among the participating nodes.

Additionally, Figure 4 depicts the similarity weight distribution obtained for each dataset in log scale. As shown, the similarity weight distribution follows power law probability distribution similarly to the degree distribution. Furthermore, the similarity weight distribution spans over wider range in US_Active and UK_Active compared to US_Passive. Particularly, in US_Passive, 91.5% of the similarity weight is less than 0.25, and this resulted from the low post frequency that users in this dataset. Yet, the frequency of users with few posts is low in US_Active and UK_Active datasets. Therefore, we can infer that the more active posting behavior of users, the more strong edges are going be added to the similarity graph. Additionally, our approach successfully adapts to different levels of user social activities, and creates the user similarity graph which reflects the underlying user behavior.



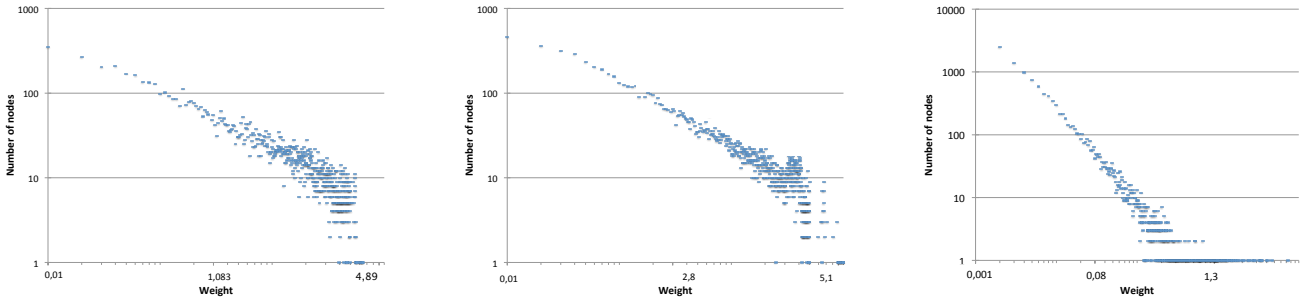
(a) US_Active (b) UK_Active (c) US_Passive

Fig. 2: The degree distribution for the friendship graphs in Twitter datasets in log scale.



(a) US_Active (b) UK_Active (c) US_Passive

Fig. 3: The in-degree distribution for the created RVOs in Twitter datasets.



(a) US_Active (b) UK_Active (c) US_Passive

Fig. 4: The Similarity weight distributions for Twitter datasets in log scale.

C. Extracted Communities

In this section, we explore DLSAS's capability of detecting spammers under the normal settings of no adversary manipulations. As aforementioned, DLSAS performs community detection to extract all the underlying behavioral patterns of users, then identify the spam behaviors among detected ones. Particularly, every node repeatedly runs the community detection, until communities structure does not change any more (i.e., the convergence is reached). Figure 5 depicts the number of rounds required till convergence, and number of extracted communities per round. As shown, in the very beginning the number of communities is very large, every node starts to form a community with one of its direct neighbors. However, over time nodes join the dominant communities in their neighborhood, as a result the communities start to merge and the number of communities continues to decrease.

In order to identify the communities that contain spammers, we have construct a list of 500 words that are commonly used by spammers associated with their semantically similar terms and n-grams (more details about the used lexicon can be found in [13]). Further, for every node we select the most frequent words used in its tweets. Accordingly, the collected word list per community is checked against a list common spam words.

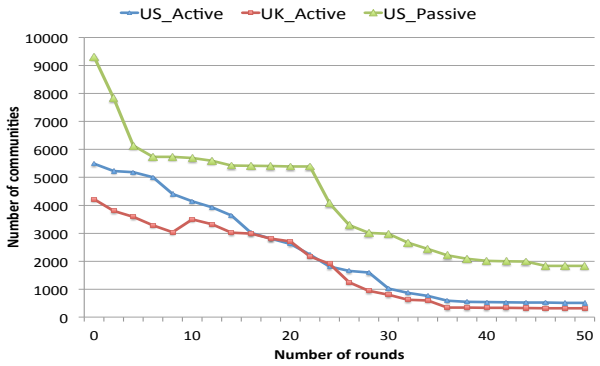


Fig. 5: The number of rounds required for convergence.

A community is identified as spam if majority (i.e., more than 50%) of its members use common spam words in their tweets. The results show that the percentage of spam communities is 17.3%, 21.6% and 23.5% in US_Active, UK_Active and US_Passive, respectively.

D. Performance Comparison

We calculate the accuracy of our approach using True Positive Rate and False Positive Rate, that are defined as the

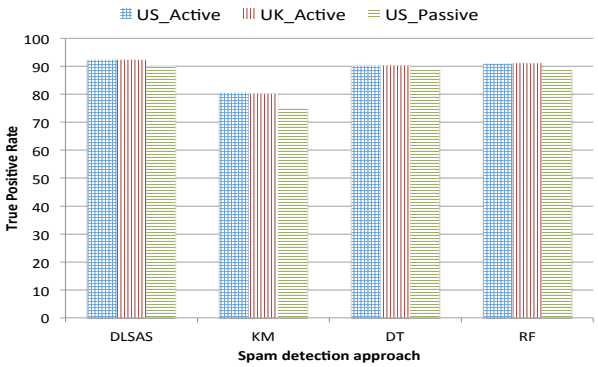


Fig. 6: True positive rate achieved by our approach compared with centralized and supervised methods.

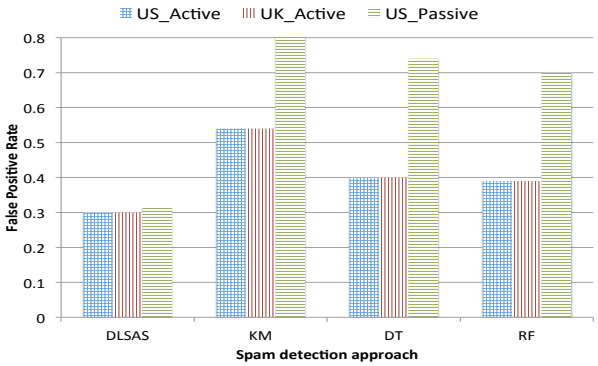


Fig. 7: True positive rate achieved by our approach compared with centralized and supervised methods.

following:

- *True Positive Rate (TPR)*: we calculate TPR as the fraction of spammers that are successfully detected.
- *False Positive Rate (FPR)*: we calculate FPR as the fraction of legitimate users that are identified as spammers.

Figure 6 depicts the detection performance comparison of our approach with the different centralized and supervised classification methods. As shown, our approach outperforms all binary classification methods. Furthermore, though our approach is a decentralized one, yet our performance is slightly better than other centralized methods. Specifically, we can find that the TPR of our approach is the highest (92.3%). Therefore, the reported result of TPR confirms the ability of graph-based clustering to achieve more accurate detection rate compared to the binary classification.

Additionally, our approach has the lowest FPR, which means that graph-based clustering successfully detect spammers with minimum affect on the legitimate users. Specifically, we can see that FPR in our approach can be steadily maintained under 0.3%, as shown in Figure 7, while this rate is 0.39% for the best binary classification method (RF). Consequently, the community detection approach adopted in our approach perfectly categorizes the existing behavioral patterns into more homogeneous and accurate clusters than binary classification.

E. Robustness to Adversarial Manipulations

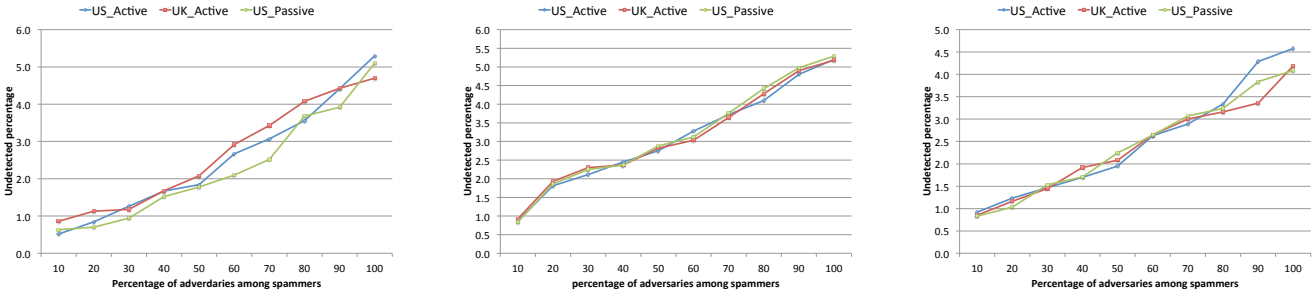
In this section, we explore DLSASs limits in terms of robustness to adversary manipulations. We conducted two different sets of experiments. The first set of experiments we randomly selected the adversary nodes from the set of suspended accounts (i.e., the accounts that are labeled as spam in the ground-truth). We simulated the increase in number of adversaries among spammers by increasing the percentage via a series of 10% growth. Each experiment was repeated for three times and we reported the average of undetected percentage of adversaries after executing the RVO validation process.

Figure 8 depicts the undetected percentage of adversary nodes among spammers for each kind of adversary manipulation act. Figure 8(a) shows reported results for *Equivocation* manipulation. In this experiment, malicious nodes need to gain more exposure to legitimate nodes while creating RVO. Therefore, malicious nodes exchange their own identifiers instead of a random subset of all their neighbors during the gossip-based peer sampling service. So as, DLSAS audits the communication log exchanged among nodes randomly while RVO construction. A node is reported malicious in case of finding unmatched logs. Additionally, to make sure that every node exchanges a random subset of its direct neighbors, the auditing procedure computes the percentage of overlap in exchanged identifiers in node communication logs. In our experiments we define the percentage of overlap in communicated friends to be less than 20%, and percentage of unselected friends to be less than 20%. As shown in Figure 8(a), DLSAS is capable of detecting at least 95% equivocations, as reported on average for all the datasets, under the condition that all spammers misbehave in terms of exchanged neighbors.

Additionally, Figure 8(b) shows the reported detection accuracy in case of false provenance manipulation, while Figure 8(c) reports the results of colluding adversary. In these experiments, we simulated the malicious acts by exchanging falsely weights for the created edges in the user similarity graph, and falsely community membership during the community detection phase. The manipulation detection mechanism is performed by RVO as illustrated in Algorithm 2. As shown, RVO validation can detect on average at least 95.2% of misbehaving nodes. Furthermore, we performed further analysis of the undetected nodes and we found that these undetected nodes lie in loosely connected communities which mostly contain spammers. Thus, these undetected nodes are unreachable from other legitimate nodes to be validated correctly.

These graphs show considerable robustness to different adversary manipulation. This comes as a consequence of the the fact that RVO validation has proven to be highly resilient to different decentralized vulnerabilities of DOSNs.

Additionally, we performed a second set of experiments, in which we explore DLSASs limits in terms of robustness to massive adversary attacks. In this experiments, we allow all nodes to participate in adversary manipulations not only



(a) Equivocation (b) False Provenance (c) Colluding Attack

Fig. 8: RVO validation accuracy reported for different DOSNs vulnerabilities using Twitter datasets.

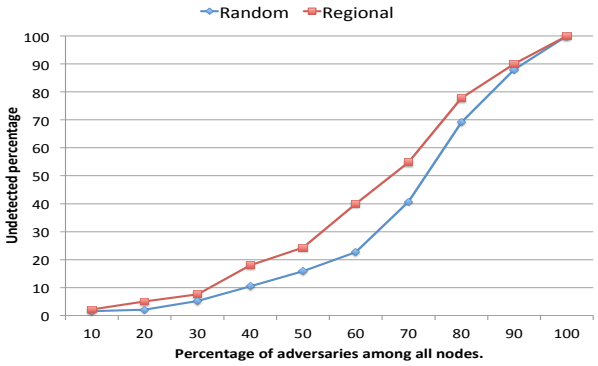


Fig. 9: RVO robustness against organized attacks.

spammers. Similarly to previous experiments, we randomly selected the adversary nodes among all participating nodes. We simulated the increase in number of adversaries in the network by increasing the percentage via a series of 10% growth. Furthermore, we performed two different types of attacks. The first one we selected adversary nodes randomly, whereas in the second one we select randomly within communities boundaries to represent regional attack behavior. Figure 9 shows the reported results for these experiments. As shown, RVO validation is highly resilient such that it can detect 34.2% of random adversaries and 23.8% of regional ones under the condition that half of the nodes in the system are malicious. However, it is expected that RVO validation degrades when the majority of nodes deviate from workflow protocol. Yet, such severe cases are not realistic as it is intractable for adversaries to become majority of a system.

V. Related Work

A. Spam Detection

A rich corpus of spam detection work lies in adopting supervised machine learning based methods using hybrid features extracted from textual content and OSN friendship graph. For example, Hongyu et al. [14] propose to train a binary classifier with user-based features that distinguish spam from legitimate content. The user social degree is used among, yet spammers can increase their social degree by purchasing more followers. Yang et al. [11] propose using graph-based features that are hard to fake such as local clustering coefficient and betweenness centrality. The main objective of spam is reaching a wide audience, so as spammers start to hijack trending topics and include many accessible accounts by intentionally mentioning them in their spam posts. Therefore, Amleshwaram et al. [12] define new content features that measure how spammers entrap victims by counting the number of unique mentions and hijacked trending topics embedded in the posts.

More recently, [15] suggests an unsupervised solution to spam detection based on sybil defense mechanism. The proposed scheme starts by identifying non-spammers (i.e., non-sybils) by applying a clustering algorithm on social graph. Additionally, the authors focus their analysis on intensive URL sharing, yet instead of using URL blacklisting, they add new user-link edges to the social graph by connecting users sharing the same URL. Afterwards, they filter the graph by removing non-sybil nodes and nodes with low degree, then the remaining nodes are identified as spammers. However, the assumption that sybil nodes form tight-knit communities does not presist as shown in recent studies [16].

B. Secure P2P Sampling

A Peer Sampling Service, identified as an important building block of large scale distributed systems, continuously provides each node in the system with a uniform random sample of all nodes in the system. Several implementations of peer sampling services exist [17, 18, 19]. However, they are extremely vulnerable to attacks by malicious nodes. The root cause of this vulnerability is that malicious nodes control the data they send to others, and it is very hard for the receiving nodes to detect their malicious intent in time.

Jelasity et al [20] provided some sketches of how to detect and neutralize malicious nodes in gossip-based protocols. However, these indicated solutions require direct interactions among all participating nodes as well as that malicious intent can be derived from the messages the nodes send. This is not possible in peer sampling where the messages are just lists of node addresses.

The Brahms [18] protocol provided each correct node with a uniform random sample of the system, which requires a global view of the system that can not be reachable in DOSNs settings. Furthermore, the samples of the provided population are rather static. It is therefore not directly suitable in dynamic

social networks nature due to rapidly evolving social activities and interactions among users.

VI. Conclusion

In this paper, we have introduced DLSAS, a novel unsupervised and fully decentralized anti-spam framework for DOSNs. In contrast to existing supervised and centralized approaches, DLSAS is unsupervised and fully decentralized spam detection framework, where each node independently processes its local data. DLSAS employs graph-based spam detection mechanism which constructs user similarity graph by encoding user behavioral patterns within graph topology. Afterwards, DLSAS detects spam by performing community detection on top of the newly created graph. This allows our system to categorize the existing behavioral patterns into more homogeneous and accurate clusters.

More importantly, DLSAS provides a novel defense mechanism for DOSNs to prevent malicious nodes participating in the system. DLSAS creates a validation overlay to assess the credibility of the exchanged information among the participating nodes and exclude the misbehaving nodes from the system. Consequently, DLSAS preserve the core of spam detection (i.e., similarity graph creation and community detection) from manipulation under the existence of adversarial nodes that deviate from the designed workflow protocol. The proposed approach achieves spam detection accuracy upto 92.3% and false positive rate of less than 0.3%. Additionally, DLSAS ensures spam-detection integrity and reliability by detecting at least 94.7% of adversaries, under the condition that 20% of nodes are spammers participating in different adversarial activities.

Acknowledgment

This work is under the umbrella of the iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN).

VII. References

[1] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "Decent: A decentralized architecture for enforcing privacy in online social networks," in *PERCOM'12*. IEEE, 2012, pp. 326–332.

[2] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Facebook and online privacy: Attitudes, behaviors, and unintended consequences," *Journal of Computer-Mediated Communication*, vol. 15, no. 1, pp. 83–108, 2009.

[3] C. Dwyer, "Privacy in the age of google and facebook," *Technology and Society Magazine, IEEE*, vol. 30, no. 3, pp. 58–63, 2011.

[4] D. Koll, J. Li, and X. Fu, "Soup: an online social network by the people, for the people," in *SIGCOMM'14*. ACM, 2014, pp. 193–204.

[5] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia, "Cachet: a decentralized architecture for privacy preserving social networking with caching," in *CoNEXT'12*. ACM, 2012, pp. 337–348.

[6] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized online social networks," in *Handbook of Social Network Technologies and Applications*. Springer, 2010, pp. 349–378.

[7] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *SIGCOMM'10*. ACM, 2010, pp. 35–47.

[8] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@ spam: the underground on 140 characters or less," in *CCS'10*. ACM, 2010, pp. 27–37.

[9] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *SP Symposium*. IEEE, 2011, pp. 447–462.

[10] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, "Social turing tests: Crowdsourcing sybil detection," *arXiv preprint arXiv:1205.3856*, 2012.

[11] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving twitter spammers," *Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, 2013.

[12] A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu, and C. Yang, "Cats: Characterizing automation of twitter spammers," in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*. IEEE, 2013, pp. 1–10.

[13] A. Soliman and S. Girdzijauskas, "Adaptive graph-based algorithms for spam detection in social networks," KTH-Royal Instituite of Technology, Tech. Rep., 2016. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:998690

[14] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. N. Choudhary, "Towards online spam filtering in social networks." in *NDSS*, 2012.

[15] E. Tan, L. Guo, S. Chen, X. Zhang, and Y. Zhao, "Unik: Unsupervised social network spam detection," in *CIKM'13*. ACM, 2013, pp. 479–488.

[16] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *TKDD'14*, vol. 8, no. 1, p. 2, 2014.

[17] Z. Bar-Yossef, R. Friedman, and G. Kliot, "Rawms-random walk based lightweight membership service for wireless ad hoc networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 5, 2008.

[18] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

[19] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, p. 8, 2007.

[20] M. Jelasity, A. Montresor, and O. Babaoglu, "Detection and removal of malicious peers in gossip-based protocols," *FuDiCo II: SOS*, 2004.