Graph-based Sybil Detection in Social and Information Systems

Yazan Boshmaf, Konstantin Beznosov, Matei Ripeanu
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
{boshmaf,beznosov,ripeanu}@ece.ubc.ca

***Abstract*—Sybil attacks in social and information systems have serious security implications. Out of many defence schemes, Graph-based Sybil Detection (GSD) had the greatest attention by both academia and industry. Even though many GSD algorithms exist, there is no analytical framework to reason about their design, especially as they make different assumptions about the used adversary and graph models. In this paper, we bridge this knowledge gap and present a unified framework for systematic evaluation of GSD algorithms. We used this framework to show that GSD algorithms should be designed to find local community structures around known non-Sybil identities, while incrementally tracking changes in the graph as it evolves over time.**

I. Introduction

Today's online systems are *open*: any user can join the system by providing an identity that is issued by either the system itself or by a trusted third-party. For example, Facebook users can join more than 9 million other websites and online services by simply authenticating with their existing user accounts [1]. In such identity-based systems, each user is intended to have a single identity and is expected to use this identity when interacting with other users in the system. Without tight offline-online identity binding [2], these systems are vulnerable to the *Sybil attack* [3]: the situation where an attacker forges many identities, each called a *Sybil*, and joins a target system for various adversarial objectives. For example, *socialbots* in online social networks control hijacked or adversary-owned user accounts in order to infiltrate these networks, steal private user data, spread misinformation, and distribute malware [4], [5].

A large body of work [6]–[10] on defending against the Sybil attack models identity-based systems as *graphs*, where nodes represent identities and edges between nodes represent well-defined relationships (e.g., user profiles and mutual friendships in Facebook, respectively). These defence schemes utilize *Graph-based Sybil Detection* (GSD) algorithms to find identities that are likely to be Sybil based on their topological properties in the graph [11], [12].

Even though many GSD algorithms exist, it is still unclear how these algorithms can be systematically evaluated, especially given that they make different assumptions about the used adversary and graph models. In this paper, we aim to bridge this knowledge gap. We present an analytical framework for systematic evaluation of GSD algorithms, along with an open-source implementation. In this framework, we define formal models to design, analyze, implement, and evaluate existing and new GSD algorithms under different adversary and graph models (Section III). We use this framework, along with a dataset of a real-world Sybil activity in Facebook [4], to systematically tackle important questions about the design and analysis of GSD algorithms in social and information systems (Section IV). We make the following contributions:

- We propose a unified framework for evaluating GSD algorithms along with an open-source implementation.
- We analytically show that GSD algorithms should be designed to find a local community structure around known honest identities (i.e., non-Sybil), regardless of the global community structure in the graph.
- We empirically show that GSD algorithms should be run regularly in order to capture the evolution of the graph, during which Sybils can be detected before they "strongly connect" to honest identities.

In the context of GSD algorithms, our work sheds light on the importance of graph evolution over time, introducing a paradigm shift from graph statistics to dynamics [13].

II. Background and related work

We now present the background required for the framework we define later in this paper (Section II-A). We also survey related work and compare it to ours (Section II-B).

A. Systems and graphs

We focus on open, online, and identity-based social and information systems such as Facebook, Amazon, and Dropbox. In these systems, user identities and their relationships can be modeled as a graph $G = (V, E)$, where every node $u \in V$ represents a user identity and every edge $e = (u, v) \in E$ represents a well-defined relationship between two identities (e.g., friendship, financial transaction, or file sharing among users in Facebook, Amazon, and Dropbox, respectively). In the graph G, there are $|V| = n$ nodes and $|E| = m$ edges. A node $u \in V$ has a degree $\deg(u)$, which is the sum of the weights on all edges incident to u. In unweighted graphs, every edge is assigned a unit weight. In what follows, we present three basic properties of graphs that are crucial for understanding GSD algorithms. From now on, *all graphs we describe are assumed to be undirected, unweighted, non-bipartite, and connected, unless we state otherwise*.

1) Centrality: Given a graph of a system, there is typically a subset of nodes that exhibit desirable topological features which make them important for proper system operation [14]. For example, high-degree nodes in social networks represent influential users who are good candidates for initial free promotion of products in viral marketing campaigns [15]. To

capture this intuition, we define the node *centrality* κ of a graph $G = (V, E)$ as a mapping

$$\kappa : G \to \{(u_i, c_i), (u_j, c_j) : \\ u_i, u_j \in V,\ c_i, c_j \in \mathbb{R},\ c_i \geq c_j,\ i < j\} \tag{1}$$

which assigns a *centrality index* $c_i \in [0, 1]$ to every node $u_i \in V$ quantifying how important the node is when compared to all other nodes in G, where a larger centrality index indicates higher importance. In other words, κ defines a total order $(u_1, c_1) \succeq \cdots \succeq (u_n, c_n)$ among nodes in G by their centrality indices. For example, the *degree centrality* κ_D of a graph G assigns an index c_i to every $u_i \in V$ as defined by

$$c_i := \frac{\deg(u_i)}{\max\{\deg(u_j) : u_j \in V\}}. \tag{2}$$

2) Mixing time: The mixing time of a graph is tightly related to its connectivity, where a well-connected graph is fast mixing and a poorly-connected graph is slow mixing [16], [17]. We formalize this property in what follows. Given a graph $G = (V, E)$, the *transition matrix* P of G is an $n \times n$ stochastic matrix, where every entry p_{ij} represents the probability of moving from node $u_i \in V$ to node $u_j \in V$, as defined by

$$p_{ij} = \begin{cases} \dfrac{1}{\deg(u_i)} & \text{if } (u_i, u_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The event of moving from one node to another in G is captured by a *Markov chain* representing a random walk over G. A *random walk* $W = (u_i, \ldots, u_t)$ of length k over G is a sequence of nodes that starts at the initial node u_i and ends at the terminal node u_t, following the transition probability defined in Equation 3. The Markov chain is called *ergodic* if it is irreducable and aperiodic. In that case, the Markov chain has a unique stationary distribution to which the random walk converges as $k \to \infty$. The *stationary distribution* π of a Markov chain is a probability distribution that is invariant to the transition matrix, that is, whenever $\pi P = \pi$. The stationary distribution of the Markov chain over G is a $1 \times n$ probability vector, and is defined as

$$\pi = \left[\frac{\deg(u_1)}{2m} \ \cdots \ \frac{\deg(u_n)}{2m} \right], \tag{4}$$

where $\pi(u_i)$ is the ith entry in π and represents the *landing probability* of node $u_i \in V$. The *marginal distribution* π_k of the Markov chain over G is a $1 \times n$ probability vector, where $\pi_k(u_i)$ is the landing probability of node $u_i \in V$ at step k of the random walk. The *total variation distance* $||\pi_k - \pi||_{\text{TV}}$ between the marginal and stationary distributions is a measure of how "close" these distribution are, and is defined by

$$||\pi_k - \pi||_{\text{TV}} := \frac{1}{2} \sum_{u_i \in V} |\pi_k(u_i) - \pi(u_i)|. \tag{5}$$

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

$$T(\epsilon) := \min\left\{k : ||\pi_k - \pi||_{\text{TV}} \leq \epsilon\right\}, \tag{6}$$

and bounded by [16]

$$\frac{\lambda}{2(1-\lambda)} \log\left(\frac{1}{2\epsilon}\right) \leq T(\epsilon) \leq \frac{\log(n) + \log\left(\frac{1}{\epsilon}\right)}{1-\lambda}, \tag{7}$$

where $\lambda \in (-1, 1)$ is the second largest eigenvalue of P. We say the Markov chain over G is *fast mixing* if $T(\epsilon)$ is polynomial in $\log n$ and $\log(1/\epsilon)$.

3) Community structure: Today's open, identity-based social and information systems are referred to as *complex networks*, as their graphs exhibit non-trivial topological features [18]. One of these features is the community structure, which was first formalized in mathematical sociology [19]. Given a graph $G = (V, E)$, a *community* is a subgraph $G' = (V', E')$ representing a tightly-knit set of nodes that are sparsely connected to the rest of the nodes in G. Accordingly, we define *community detection* χ, also known as *graph clustering*, as a mapping

$$\chi : G \to G'_1 \times \cdots \times G'_c \tag{8}$$

that partitions G into c non-empty, node-disjoint subgraphs $G'_1 \times \cdots \times G'_c$ representing a set of communities or *clusters*. A widely used quality measure for community detection is the *modularity* Q of the clustering $\chi(G)$ [20], which is a mapping

$$Q : \chi(G) \to \mathbb{R} \tag{9}$$

that assigns a quality value $q \in [-1, 1]$ to the clustering $\chi(G)$, as defined by

$$q := \sum_{G'_i \in \chi(G)} \left(\frac{\deg_{G'_i}(V'_i)}{\deg_G(V)} - \frac{\deg_G(V'_i)^2}{\deg_G(V)^2} \right), \tag{10}$$

where $\deg_G(V)$ is the sum of weights on all edges in G that are incident to every node in V. The higher the quality value q is, the better the detected community is. One possible definition for χ is to maximize Q over all possible clustering $\chi(G)$ [21], which was shown to be an NP-hard problem [22].

A related quality measure for community detection is the *conductance* ϕ of the graph, which is a mapping

$$\phi : G \to \mathbb{R} \tag{11}$$

that assigns a value $h > 0$ to G, known as the *Cheeger constant* of the graph, as defined by

$$h := \arg\min_{\mathcal{C} \in \mathcal{H}} \varphi(\mathcal{C}), \tag{12}$$

where $\mathcal{H}$ is the set of all pairs $\mathcal{C} = \{G'_L, G'_R\} \in \chi(G)$, each called a *cut*, representing all possible clustering in G of length $c = 2$. In turn, the *cut conductance* φ is a mapping

$$\varphi : \mathcal{C} \to \mathbb{R} \tag{13}$$

that assigns a value $\omega > 0$ to the cut $\mathcal{C} = \{G'_L, G'_R\}$ quantifying how well-connected the graph is across the corresponding left and right subgraphs of the cut, and is defined by

$$\omega := \frac{|\{(u, v) : u \in V'_L,\ v \in V'_R,\ (u, v) \in E\}|}{\min\{\deg_G(V'_L), \deg_G(V'_R)\}}. \tag{14}$$

The Cheeger constant h of the graph G quantifies how well-connected G is when all cuts in G are considered. A small h indicates the existence of a relatively sparse cut separating two well-knit subgraphs. For all clustering $\chi(G)$ of length $c = 2$,

a possible definition for χ is to compute ϕ, which was shown to be an NP-complete problem [23].

Given a graph G, our previous intuition on the tight relationship between connectivity, conductance, and mixing time is reinforced by the bound [16]

$$\frac{1-\lambda}{2} \leq \phi \leq \sqrt{2(1-\lambda)}, \tag{15}$$

where λ is the second largest eigenvalue in Equation 7. The graph conductance controls how fast a random walk in G converges to its stationary distribution, which in turn quantifies how well-connected the graph is. This observation, as we show next, underlies state-of-the-art GSD algorithms.

B. Graph-based Sybil detection

We consider Sybil defences that leverage the system graph to uncover latent Sybil identities. Ideally, each user in the system is intended to have a single *honest identity* and is expected to use this identity when interacting with other users in the system. We call a user with multiple identities a *Sybil user* and each of the used identities a *Sybil identity*.

Given a graph G of a system, state-of-the-art *Graph-based Sybil Detection* (GSD) algorithms operate on G by making the following assumptions [11], [12]: First, the system owner knows at least one honest identity in G. Second, the adversary cannot establish arbitrarily many relationships between Sybil and honest identities. Third, the subgraph induced by the set of honest identities is fast mixing. Given these assumptions, GSD algorithms try to find a sparse cut $\mathcal{C} = \{G'_L, G'_R\}$ in G that minimizes the graph conductance such that G'_L consists of mostly honest identities, as verified by the known ones, and G'_R consists of mostly Sybil identities. We formalize a generalization of this intuition in Section III.

Recent research showed that graphs of real-world social and information systems do not necessarily conform to the assumptions above. For example, Leskovec et al. [24] showed that such graphs have many small periphery communities that do not form one big community or cluster. Likewise, Mohaisen et al. [25] showed that these graphs are generally not fast mixing. Moreover, Boshmaf et al. [5] showed that an adversary can infiltrate a target online social network at a large scale by tricking users into establishing relationships with Sybils. To this end, it is unclear how GSD algorithms perform when these assumptions do not hold in practice. Viswanath et al. [9] were among the first to study this problem, and they empirically showed that existing GSD algorithms work by detecting communities around known honest nodes. We build on their work, and present a novel analytical framework for systematic evaluation of GSD algorithms. Our framework enables the design and analysis of existing and new GSD algorithms with analytically-sound security guarantees that can be easily evaluated empirically. To the best of our knowledge, we are the first to provide a comprehensive treatment of this topic.

III. A Unified Framework for GSD Algorithms

We now define the models underlying our framework. We implemented all of the discussed models for graph-based *S*ybil detection in a *Py*thon package called *SyPy*.[1] The package offers simple abstractions to design, implement, and evaluate GSD algorithms. For a short overview of the implementation, we point the reader to our technical report [26]. We also summarize all of the notations used in this paper in Table I.

Symbol	Description
G	Undirected, unweighted, non-bipartite graph
κ	Centrality index of a graph
π	Stationary distribution of a Markov chain
$\|\|\cdot\|\|_{\text{TV}}$	Total variation distance
ϵ	Error in total variation distance
$T(\epsilon)$	Mixing time of the Markov chain
λ	Second largest eigenvalue of a matrix
χ	Community detection in a graph
Q	Modularity of communities in a graph
$\mathcal{C}$	Cut in a graph
ϕ	Conductance of a graph
φ	Cut conductance in a graph
N	Network of a social or information system
H	Honest region of a network
S	Sybil region of a network
A	Attack edges of a network
δ	Sybil detector for a given network graph
τ	Threshold of a Sybil detector
ψ	Partitioner for a Sybil detector
Δ	GSD algorithm
ρ	Detection performance of a GSD algorithm
α	Adversary in a system

TABLE I: Notation index

A. System model

Every identity in the system is either Sybil or honest. We refer to this classification as the *label* of the identity. A *region* $R = (V_R, E_R, K_R)$ of a system consists of a *region graph* $G_R = (V_R, E_R)$ and a node set $K_R \subset V_R$. In the region graph G_R, every node in V_R represents a unique identity and every edge in E_R represents a well-defined relationship between two identities. The set K_R contains nodes in V_R that have known labels as indicated by the *ground truth* ℓ, which is a mapping

$$\ell : V_R \to \{0, 1\}, \tag{16}$$

that is defined by

$$\ell(u) = \begin{cases} 1 & \text{if } u \in V_R \text{ is a Sybil identity,} \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Given the ground truth ℓ, we define two non-overlapping regions. The first is the *honest region* $H = (V_H, E_H, K_H)$, and it consists of only honest identities along with their relationships. In this region, we assume the system owner knows at least $|K_H| \geq 1$ identities such that $\ell(u) = 0$ for every $u \in K_H$. Similarly, the second region $S = (V_S, E_S, K_S)$ is the *Sybil region*, and it consists of only Sybil identities along with their relationships. We also assume the system owner knows $|K_S| \geq 0$ identities such that $\ell(u) = 1$ for every $u \in K_S$. Typically, the set $K_S = \emptyset$. To this end, every identity in the system belongs to either H or S but not both.

[1] http://boshmaf.github.io/sypy/

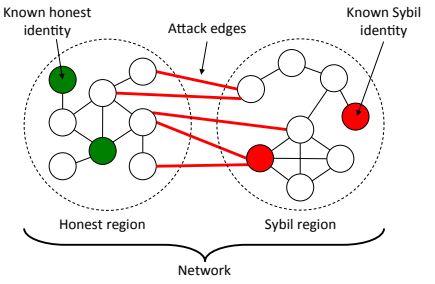


Fig. 1: The system model.

The *network* $N = (H, S, A)$ of a system consists of an honest region H, a Sybil region S, and a non-empty set A of *attack edges* that connect the two regions, and is defined by

$$A := \{(u, v) : u \in V_H,\ v \in V_S\}. \tag{18}$$

Typically, the system owner has knowledge of only the *network graph* $G_N = (V_N, E_N, K_H, K_S)$, where $V_N = V_H \cup V_S$ and $E_N = E_H \cup E_S \cup A$. The network graph models bilateral and triadic relationships found in social and information systems [27]. The system owner does not know the ground truth ℓ nor the network N, but aims to label the nodes in G_N correctly. We demonstrate the system model in Figure 1.

B. Adversary model

The objective of an adversary is to manipulate the network graph using some strategy in order to successfully mount a Sybil attack. An attack, for example, may involve harvesting private user data in online social networks [5], polluting the voting scheme of online reputation systems [28], or subverting routing and replication services in DHTs [29]. We formalize this intuition in what follows. Given a snapshot of a network N and its graph G_N at some particular point in time, an *adversary* α is a mapping

$$\alpha : S \times A \to S^\alpha \times A^\alpha \tag{19}$$

that alters the Sybil region S and the attack edges A, using some adversarial strategy. For the defender (i.e., the system owner), the adversary α is capable of transforming G_N to a new graph $G_N^\alpha = (V_N^\alpha, E_N^\alpha, K_H, K_S)$, which reflects all alterations made by the adversary α, where $V_N^\alpha = V_H \cup V_S^\alpha$ and $E_N^\alpha = E_H \cup E_S^\alpha \cup A^\alpha$. We disregard changes made to the honest region H, as they originate from honest identities rather than Sybil ones.

We define two adversaries in the network N: A *regular adversary* α_r, and a *social adversary* α_s. The regular adversary α_r has a complete control over the Sybil region S, but cannot establish more than $O(n/T(\epsilon))$ attack edges, where $T(\epsilon)$ is the mixing time of the Markov chain over G_N. This bound is typically assumed in GSD algorithms [11], [12], as one expects the adversary to require non-trivial social engineering capabilities in order to establish many attack edges. The social adversary α_s, however, is more resourceful and has the same capabilities of α_r but can establish arbitrarily large number of attack edges, which is motivated by recent empirical results in online social networks [4], [30]. In practice, the adversary α is limited by the scale of changes made between G_N and G_N^α, in addition to the behaviour of identities in the honest region H.

C. Detection model

Following the intuition of node centrality, Sybil identities often exhibit unique topological features which make them distinguishable from honest identities. For example, *outliers*, nodes that have proportionally smaller degree centrality indices when compared to others, represent isolated identities that are likely to be Sybil [9]. Accordingly, we define a *detector* δ in a given network graph G_N as a mapping

$$\delta : G_N \to \{(u_i, r_i), (u_j, r_j) : u_i, u_j \in V_N,\ r_i, r_j \in \mathbb{R},\ r_i \geq r_j,\ i < j\}, \tag{20}$$

which assigns a *rank* $r_i \in [0, 1]$ to every $u_i \in V_N$ that describes how Sybil-like the corresponding identity is, where a higher rank value indicates the identity is more likely to be Sybil. Thus, δ defines a total order $(u_1, r_1) \succeq \cdots \succeq (u_n, r_n)$ among nodes in V_N by their rank values, which was shown to be the case for GSD algorithms [9]. The detector δ is called *binary* if for every $u_i \in V_N$, the rank value $r_i \in \{0, 1\}$. Otherwise, the detector δ is called *scaler*.

A detector by itself does not label identities, but rather finds a linear ordering of the corresponding nodes. Accordingly, one needs to define a cutoff position in the ordering after which nodes are considered to represent Sybil identities. To capture this intuition, we define a *detection threshold* $\tau \in [0, 1]$ for every detector δ operating on G_N. This allows us to define a *partitioner* ψ for δ using τ, which is as a mapping

$$\psi : \delta(G_N) \times \mathbb{R} \to G_H^\delta \times G_S^\delta \tag{21}$$

that splits the network graph G_N into two non-empty, node-disjoint subgraphs, $G_H^\delta = (V_H^\delta, E_H^\delta)$ and $G_S^\delta = (V_S^\delta, E_S^\delta)$, such that

$$\begin{aligned} V_S^\delta &:= \{u_i : r_i \geq \tau,\ (u_i, r_i) \in \delta(G_N)\}, \\ V_H^\delta &:= \{u_i : r_i < \tau,\ (u_i, r_i) \in \delta(G_N)\}, \\ & \qquad V_H^\delta \cap K_H \neq \emptyset. \end{aligned} \tag{22}$$

The resulting cut $\mathcal{C} = \{G_H^\delta, G_S^\delta\}$ in G_N represents the honest region graph G_H^δ and the Sybil region graph G_S^δ, as identified by δ using the detection threshold τ. To this end, we define a GSD *algorithm* by the triple $\Delta = (\delta, \tau, \psi)$, which in turn defines the cut $\mathcal{C} := \psi(\delta(G_N), \tau)$ in G_N.

We assume the system owner can estimate the threshold τ for the used detector δ. For binary detectors, we set $\tau = 1$. For scaler detectors, the owner can manually verify random samples of identities labeled as Sybil using different values of τ, and then pick τ that achieves the best detection performance [8]. This approach, however, is both expensive and non-scalable. We tackle this problem based on the following observation: As the nodes are ranked by how Sybil-like they are, we can estimate τ by minimizing the conductance over all subgraphs induced by the top-k ranked nodes, for $0 < k < n$. Formally, let $\mathcal{R} = \{r_i : (u_i, r_i) \in \delta(G_N)\}$ be the set of ranks computed by a scaler detector δ, we accordingly define its detection threshold τ by

$$\tau := \left\{ r_k : \arg\min_{r_i \in \mathcal{R}} \varphi\left(\{G'_{r_i}, G_N \setminus G'_{r_i}\}\right) \right\}, \tag{23}$$

where $G'_{r_i} = (V'_{r_i}, E'_{r_i})$ is a subgraph of G_N induced by the node set $V'_{r_i} \subset V_N$, which is defined by

$$V'_{r_i} := \{u_j : (u_j, r_j) \in \delta(G_N),\ r_j \geq r_i\}. \tag{24}$$

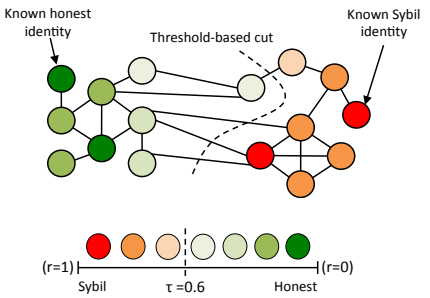


Fig. 2: The detection model. Node colors represent ranks. A greener color indicates an identity that is less likely to be Sybil. A threshold over the rank values partitions the network graph into two non-empty, node-disjoint subgraphs or regions.

Input: Network graph $G_N = \{V_N, E_N, K_H, K_S\}$
Output: Total order $O := (u_1, r_1) \succeq \cdots \succeq (u_n, r_n)$
while G_N **is not** *disconnected* **do**
  $B \leftarrow$ edge betweenness centrality of G_N
  $e_i \leftarrow$ edge with highest centrality index c_i in B
  remove e_i from G_N
$V_C \leftarrow$ connected component $V_C \subset V_N$ in G_N that maximizes $|V_C \cap K_S|$ or minimizes $|V_C \cap K_H|$
$O \leftarrow \emptyset$
for $u_i \in V_C$ **do**
  append $(u_i, 1)$ to O
for $u_i \in V_N \setminus V_C$ **do**
  append $(u_i, 0)$ to O
return O

Fig. 3: GLOBALCOMMDETECTOR δ_G

Although computing the graph conductance is an NP-complete problem [23], the instance in Equation 23 considers only $O(n)$ unique cuts in G_N. This means that τ can be computed in $O(\eta n)$ time, where $\eta = O(n)$ is the time required to compute the conductance of a given cut, as defined in Equation 14. We demonstrate the detection model in Figure 2.

D. Performance model

Given a GSD algorithm $\Delta = (\delta, \tau, \psi)$ and its corresponding cut $\mathcal{C} := \psi(\delta(G_N), \tau)$ in the graph G_N of the network N, we define the *detection performance* ρ of Δ as a mapping

$$\rho : \mathcal{C} \times H \times S \to \mathbb{R} \tag{25}$$

that computes a performance score $p \in [0, 1]$ describing how well δ performs in labeling identities when partitioned by ψ using τ, as compared to the correct labels identified by the ground truth ℓ in H and S. A higher score means a better performance, which also indicates a labeling that is "closer" to the one defined by ℓ. The ground truth labeling has a unit performance score $p = 1$.

IV. Framework Analysis and Discussion

Armed with the framework presented in Section III, we tackle three important questions regarding the systematic evaluation of GSD algorithms in social and information systems. Our aim here is to relax the strong assumptions made about the used adversary and graphs models by exploring the design space of GSD algorithms, and then picking the techniques that offer both analytically-sound security properties and practically-efficient computation.

A. How should one measure the detection performance?

GSD algorithms can be treated as *classifiers* that label nodes as either Sybil; the *positive* class, or honest; the *negative* class. This allows us to use well-known performance measures that are suited for classification problems [31]. As presented in Section III-D, let $\mathcal{C} := \psi(\delta(G_N), \tau)$ be a cut found by the algorithm $\Delta = (\delta, \tau, \psi)$ in the graph G_N of a network N. We accordingly define the algorithm's *true negative rate* ρ_{N}, or its *specificity*, by

$$\rho_{\text{N}} := \frac{|V_H^\delta \cap V_H|}{|V_H|}, \tag{26}$$

and its *true positive rate* ρ_{P}, or its *sensitivity*, by

$$\rho_{\text{P}} := \frac{|V_S^\delta \cap V_S|}{|V_S|}. \tag{27}$$

Given these performance measures, the analysis using *Receiver Operating Characteristic* (ROC) becomes possible [32]. In the absence of a ground truth ℓ, which is the case for online classification, GSD algorithms should provide formal bounds on how many Sybil identities could be ranked lower than honest identities, and thus limiting the *false positive rate* $1-\rho_{\text{N}}$. Alternatively, the algorithm should provide provable bounds on how many Sybils per attack edge can exist in the system in the worst case. We show an example of such a desirable guarantee in what follows.

B. How should one design GSD algorithms?

GSD algorithms operate by finding a community structure around known honest nodes [9]. This can be achieved by using either global or local community detection. In what follows, we present two GSD algorithms based the global community detection algorithm by Girvan and Newman [33], and the local community detection algorithm by Andersen, Chung, and Lang [34]. We analytically show why the latter is more Sybil-resilient in the context of GSD algorithms, while the earlier is still needed for the case when the honest region has existing community structures, and therefore, is not fast mixing.

We start with global community detection. The intuition of node centrality of graphs, which we defined in Section II-A1, can be extended to edges as well. This allows us to define an edge centrality measure that is useful for detecting global community structures. Given a graph G, the edge *betweenness centrality* κ_B of G is a mapping that assigns every edge $e_i \in E$ a centrality index $c_i \in [0, 1]$ as defined by

$$c_i := \frac{1}{n(n-1)} \sum_{s,t \in V} \frac{\sigma(s, t \mid e_i)}{\sigma(s, t)} \tag{28}$$

where $\sigma(s, t)$ is the number of shortest paths between the nodes s and t, and $\sigma(s, t \mid e_i)$ is the number of shortest paths between the same nodes that pass through the edge e_i. In Equation 28, let $\sigma(s, t) = 1$ if $s = t$, $\sigma(s, t \mid e_i) = 0$ if $e_i = (s, t)$, and $0/0 = 0$ by convention [35].

Input: Network graph $G_N = \{V_N, E_N, K_H, K_S\}$
Output: Total order $O := (u_1, r_1) \succeq \cdots \succeq (u_n, r_n)$
$\pi_0 \leftarrow \emptyset$
for $u_i \in V_N$ **do**
  if $u_i \in K_H$ **then**
    $\pi_0(u_i) \leftarrow \frac{1}{|K_H|}$
  else
    $\pi_0(u_i) \leftarrow 0$
for $k \leftarrow 1$ **to** $O(\log n)$ **do**
  $\pi_k \leftarrow \emptyset$
  for $u_i \in V_N$ **do**
    $\pi_k(u_i) \leftarrow \sum_{(u_j,u_i)\in E_N} \frac{\pi_{k-1}(u_j)}{\deg(u_j)}$
$O_t \leftarrow \emptyset$
for $u_i \in V_N$ **do**
  append (u_i, r_i) to O_t where $r_i = 1 - \frac{\pi_k(u_i)}{\deg(u_i)}$
$O \leftarrow \emptyset$
while O_t **is not** *empty* **do**
  append (u_i, r_i) to O where u_i has maximum r_i in O_t
  remove (u_i, r_i) from O_t
return O

Fig. 4: LocalCommDetector δ_L

From graph connectivity perspective, edges that have high betweenness centrality represent bridge-like connectors between two dense subgraphs or communities. The removal of these edges will affect the connectivity between many pairs of nodes through the shortest paths between them, and potentially will disconnect the graph into two communities. Based on this observation, we present a GSD algorithm $\Delta_G = (\delta_G, \tau, \psi)$ whose detector δ_G is binary and ranks the nodes in G_N by their membership to the subgraphs defined by a sparse cut in G_N. The cut is found by repeatedly removing edges with the highest betweenness centrality until the graph becomes disconnected, as described in Algorithm 3.

We can compute the edge betweenness of G_N and decide whether it is disconnected in $O(nm)$ time [35], [36]. If the regions H and S are relatively balanced and fast mixing, we expect the attack edges established by a regular adversary α_r to have the highest betweenness centrality. In this case, the detector δ_G will take $O(gnm)$ time to rank all of the nodes in G_N, where $g = O(n/T(\epsilon))$ is the number of attack edges. In fact, this is the best *attack scenario* for the system owner. In GSD algorithms [12], only the tighter case of $\epsilon = \Theta(1/n)$ is considered, and thus $T(\epsilon) = O(\log n)$ leading to a running time of $O(mn^2/\log n)$ or $O(n^4/\log n)$ in dense graphs.

One drawback of global community detection is that we start by processing the largest possible input: The whole network graph G_N. A more serious drawback, in the context of GSD algorithms, is that we make unrealistic assumptions about the structure of the Sybil region S, which we know is under the complete control of the adversary α_r. For example, if α_r carefully alters S such that it consists of two identical but disconnected components that are only connected to H via attack edges, then δ_G will misclassify half of the Sybils. This leads us to the direction of *local* community detection, where we aim to find a sparse cut near a trusted set of nodes, independently from the global community structure of the

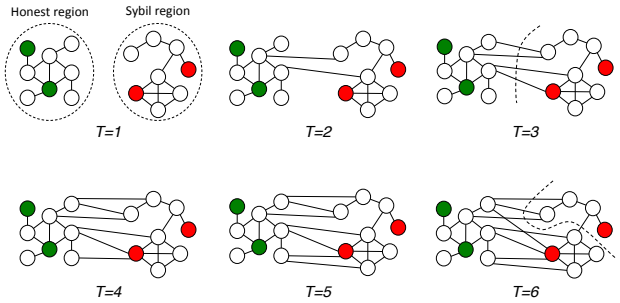


Fig. 5: Graph dynamics and GSD algorithms. At timestamp $T = 1$, the honest and the Sybil regions are disconnected. Over time, from $T = 2$ to $T = 6$, the network graph transforms through a series of alterations made by a social adversary α_s. When compared to $T = 3$, running a GSD algorithm at timestamp $T = 6$ results in a lower true negative rate, where the number of attack edges doubles.

graph. We now formalize this intuition. Let us consider a random walk $W = (u_i, \ldots, u_t)$ of length k over a network graph G_N, where the initial node is $u_i \in K_H$. We call W a *lazy* random walk if $k < T(\epsilon)$. Assuming a regular adversary α_r and a fast mixing honest region H, there are $g = O(n/T(\epsilon))$ attack edges that represent a sparse cut between H and S. In this case, a lazy random walk W in G_N is expected to contain mostly nodes from H, as the walk is unlikely to traverse one of the relatively few attack edges and enter S. In other words, the landing probability $\pi_k(u_i)$ of every u_i at step k of the walk is expected to be relatively large if $u_i \in V_H$, as compared to $\pi_k(u_j)$ for every $u_j \in V_S$. Recall that π_k can be defined iteratively by

$$\pi_k := \pi_{k-1}P = \pi_0 P^k, \tag{29}$$

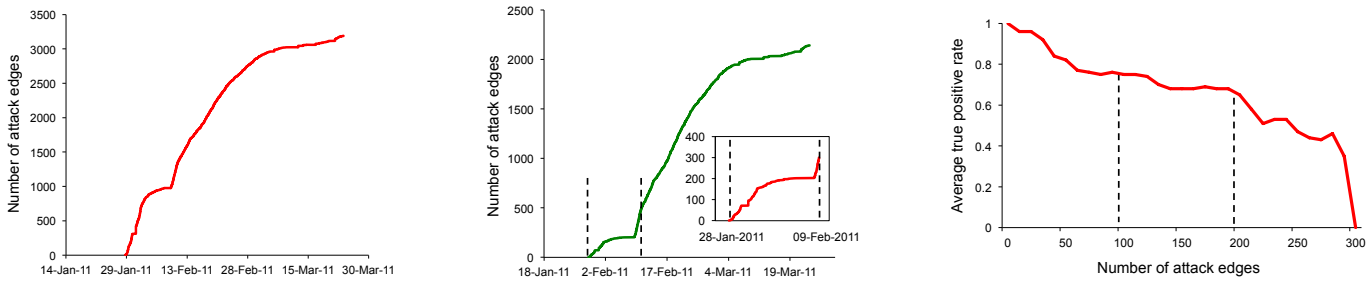
where π_0 is the initial landing probability distribution, we can compute the landing probability $\pi_k(u_i)$ for every $u_i \in V_N$ at step k by

$$\pi_k(u_i) = \sum_{(u_j,u_i)\in E_N} \frac{\pi_{k-1}(u_j)}{\deg(u_j)}. \tag{30}$$

For the case of $k \geq T(\epsilon)$, it follows that $||\pi_k - \pi||_{\text{TV}} \leq \epsilon$, and Equation 4 can be used to approximate π_k [37].

Based on this observation, we present a GSD algorithm $\Delta_L = (\delta_L, \tau, \psi)$ whose detector δ_L is scaler and assigns a rank $r_i = 1 - (\pi_k(u_i)/\deg(u_i))$ to every node $u_i \in V_N$. The rank value r_i is the degree-normalized landing probability of u_i in a random walk that starts from any node $u_j \in K_N$ with equal probability, but terminates after a small number of steps $k = O(\log n)$ regardless to the mixing time of G_N, as described in Algorithm 4.

The detector δ_L runs in $O(n \log n)$ time, and assuming the adversary α_r establishes attack edges with honest nodes at random, δ_L guarantees that the total number of Sybil identities that are ranked lower than honest identities is $O(g \log n)$ [8], [34]. Thus, local community detection is more efficient and Sybil-resilient when compared to global community detection. The network graph, however, may have many small periphery communities that do not form one big community or cluster [24], and thus, are not fast mixing [25]. To tackle this problem, we can use the non-Sybil-resilient global community



(a) (b) (c)

Fig. 6: Large-scale infiltration in Facebook over (a) the complete sampled graph, (b) its Largest Connected Component (LCC), and (c) the corresponding average true positive rate of the GSD algorithm $\Delta_L = (\delta_L, \tau, \psi)$. The LCC consisted of 155,622 nodes and 751,199 edges, with an average clustering coefficient of 0.5405 and a diameter of 23. Overall, there were 2,143 attack edges connecting the LCC to the Sybil region, which consisted of 100 Sybil identities. Each Sybil identity had at least one attack edge.

detection to split the graph into a set of non-empty, node-disjoint subgraphs, as described in Equation 8. This could be achieved efficiently by locally maximizing the gain in modularity around every node in the graph, just as in the well-known Louvain method [38]. After global community detection, each identified community is considered a new but smaller network graph where local community detection can be applied to detect Sybils around a set of known honest identities. For the likely case when the smaller network graph does not contain any of the original known honest identities, the system owner can construct one by manually verifying a small random sample of the nodes [39], which has the advantage of making the system more resilient to an adversary who tries to established attack edges with known honest nodes, as compared to a deterministic method.

C. How often should one run GSD algorithms?

Given a social adversary α_s, we expect both Δ_G and Δ_L to be ineffective, as the number of attack edges can grow arbitrarily large. In fact, for Δ_L the $O(g \log n)$ bound means that $O(n)$ identities can get misclassified if $g = \Omega(n/\log n)$, which is explained by the absence of a sparse cut between the Sybil and the honest regions. The adversary, however, is expected to spend a relatively long time before the edges are established, as this involves carrying out a non-trivial social engineering attack [5]. This leads us to the direction of *graph dynamics*, where one considers the growth of communities in the network graph over time, as demonstrated in Figure 5.

To explore how often the system owner is required to run a GSD algorithm, we used a dataset of a recent real-world Sybil activity in Facebook [4]. The dataset describes the results of a research experiment that evaluated the feasibility of running a large-scale infiltration campaign by ~ 100 automated fake accounts in Facebook. The data were collected over the period from January 28 through March 23 in 2011, during which a total of $3,190$ attack edges were established between Sybil and honest identities. Figure 6(a) shows how fast these edges were established. The rapid increase after two weeks is explained by the *triadic closure principle* [40]: After establishing the first 300 attack edges, the Sybils were able to have mutual connections with honest identities, which improved the subsequent success rate of the infiltration.

Let us consider the Largest Connected Component (LCC) in the BFS-sampled graph.[2] We focus on the first two weeks of the infiltration, January 28 through February 9, which we show in Figure 6(b). During the two weeks, a total of 300 attack edge were established between honest identities in the LCC and 100 Sybils, where the latter formed a *clique* or a completely-connected graph. After applying Louvain method [38], the honest identities that had at least one attack edge with the Sybils were assigned to a single dense subgraph in the LCC, which represented a community structure consisting of $10,053$ nodes and $41,654$ edges, with an average clustering coefficient of 0.3271 and a diameter of 15. We marked this dense subgraph as the honest region, and then for every new attack edge we ran the local community detector Δ_L on the resulting network graph for 10 times and calculated its average true positive rate, where in each run we picked a node in the honest region at random and marked it as the known honest node. As shown in Figure 6(c), as the number of attack edges increased, the average true positive rate of Δ_L decreased all the way down to zero. This gives us an indicator that for GSD algorithms to be effective, they need to be run regularly; at least on a weekly basis in this case. To be computationally efficient, on the other hand, GSD algorithms have to be adjusted so that they perform incremental processing of the network graph as it evolves, in a way similar to real-time graph mining [41].

V. Conclusion and Future Work

We presented a unified framework to reason about and evaluate Graph-based Sybil Detection (GSD) algorithms. Using the framework, along with a dataset of a real-world Sybil activity in Facebook, we showed that GSD algorithms should be designed to find local community structure around a set of known honest identities, while incrementally tracking the changes in the graph as nodes and edges are added and removed. Moving from graph statistics to graph dynamics promises better detection performance, as one expects to achieve an early detection of Sybil identities in social and information systems.

We are currently investigating different ways to design an incremental GSD algorithm that scales to graphs consisting of

[2] As the Sybils initially targeted nodes at random, the complete BFS-sample representing the honest region consisted of multiple connected components.

hundreds of millions of nodes, while providing near real-time detection of Sybil identities with formal security guarantees.

ACKNOWLEDGMENTS

We thank Tony Cheng for his help on SyPy. This research is partially supported through funding from the NSERC Internetworked Systems Security Network (ISSNet) and GRAND NCE. The first author is thankful to the University of British Columbia for a generous doctoral fellowship.

REFERENCES

[1] Facebook Inc., "Platform statistics," http://newsroom.fb.com/Platform, January 2013.

[2] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "Key challenges in defending against malicious socialbots," in *Proceedings of the 5th USENIX Workshop on Large-scale Exploits and Emergent Threats*, ser. LEET'12, Berkeley, CA, USA, 2012.

[3] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 251–260.

[4] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 93–102.

[5] ——, "Design and analysis of a social botnet," *Computer Networks*, 2012.

[6] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: defending against sybil attacks via social networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 267–278, August 2006.

[7] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against sybil attacks," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 3–17.

[8] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 15–15.

[9] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 363–374.

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

[11] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post, "Exploring the design space of social network-based sybil defenses," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, January 2012, pp. 1–8.

[12] H. Yu, "Sybil defenses via social networks: a tutorial and survey," *SIGACT News*, vol. 42, pp. 80–101, October 2011.

[13] T. A. Snijders, "The statistical evaluation of social network dynamics," *Sociological methodology*, vol. 31, no. 1, pp. 361–395, 2001.

[14] U. Brandes and T. Erlebach, *Network analysis: methodological foundations*. Springer, 2005, vol. 3418.

[15] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2003, pp. 137–146.

[16] A. Sinclair, "Improved bounds for mixing rates of markov chains and multicommodity flow," *LATIN'92*, pp. 474–487, 1992.

[17] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. Amer Mathematical Society, 2009.

[18] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," *Physics reports*, vol. 424, no. 4, pp. 175–308, 2006.

[19] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University, 1994.

[20] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[21] R. Rotta and A. Noack, "Multilevel local search algorithms for modularity clustering," *J. Exp. Algorithmics*, vol. 16, pp. 2.3:2.1–2.3:2.27, Jul. 2011.

[22] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 2, pp. 172–188, 2008.

[23] J. Šíma and S. E. Schaeffer, "On the np-completeness of some graph cluster measures," in *SOFSEM 2006: Theory and Practice of Computer Science*. Springer, 2006, pp. 530–537.

[24] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

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

[26] M. R. Yazan Boshmaf, Konstantin Beznosov, "Graph-based sybil detection in social and information system," *LERSSE Online Digital Library*, 2013.

[27] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[28] A. Jøsang and J. Golbeck, "Challenges for robust of trust and reputation systems," Sep. 2009.

[29] G. Urdaneta, G. Pierre, and M. van Steen, "A survey of DHT security techniques," *ACM Computing Surveys*, vol. 43, no. 2, Jan. 2011.

[30] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 259–268.

[31] C. Parker, "An analysis of performance measures for binary classifiers," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 2011, pp. 517–526.

[32] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Machine Learning*, vol. 31, pp. 1–38, 2004.

[33] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[34] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 475–486.

[35] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[36] ——, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[37] G. H. Golub and H. A. Van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 35–65, 2000.

[38] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, 2008.

[39] R. Andersen and K. J. Lang, "Communities from seed sets," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 223–232.

[40] A. Rapoport, "Spread of information through a population with socio-structural bias: I. assumption of transitivity," *Bulletin of Mathematical Biology*, vol. 15, pp. 523–533, 1953, 10.1007/BF02476440.

[41] Z. Cai, D. Logothetis, and G. Siganos, "Facilitating real-time graph mining," in *Proceedings of the fourth international workshop on Cloud data management*. ACM, 2012, pp. 1–8.