A Novel Recruitment Policy to Defend against Sybils in Vehicular Crowdsourcing

Article

Accepted version

Concone F., De Vita F., Pratap A., Bruneo D., Lo Re G. and Das S. K.

In Proceedings of the 2021 IEEE International Conference on Smart Computing (SMARTCOMP)

Publisher: IEEE

A Novel Recruitment Policy to Defend against Sybils in Vehicular Crowdsourcing

Federico Concone[†], Fabrizio De Vita[‡], Ajay Pratap[⋆], Dario Bruneo[‡], Giuseppe Lo Re[†], and Sajal K. Das[≀]
[†]Department of Engineering, University of Palermo, Italy
[‡]Department of Engineering, University of Messina, Italy
[⋆]Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, India
[≀]Department of Computer Science, Missouri University of Science and Technology, Rolla, USA
{federico.concone, giuseppe.lore}@unipa.it, {fdevita, dbruneo}@unime.it, ajay.cse@iitbhu.ac.in, sdas@mst.edu

***Abstract*—Vehicular Social Networks (VSNs) is an emerging communication paradigm, derived by merging the concepts of *Online Social Networks* (OSNs) and *Vehicular Ad-hoc Networks* (VANETs). Due to the lack of robust authentication mechanisms, social-based vehicular applications are vulnerable to numerous attacks including the generation of sybil entities in the networks. We address this important issue in vehicular crowdsourcing campaigns where sybils are usually employed to increase their influence and worsen the functioning of the system. In particular, we propose a novel *User Recruitment Policy* (URP) that, after extracting the participants within the event radius of a crowdsourcing campaign, detects and filters out the sybil vehicles by using a novel sybil detection approach, called *SybilDriver*. This technique combines the advantages of VANETs and OSNs by means of an innovative concept of *proximity graph* obtained from the physical vehicular network, in conjunction with a community detection and *Random Forest* techniques adopted in the OSN domain. Detailed experimental evaluations demonstrate the effectiveness of our approach and also show that it outperforms existing state-of-the-art methods typically used in the OSNs.** [1]

***Index Terms*—Vehicular Social Network; Crowdsourcing; Sybil detection; Trust and Truthfulness; Proximity Graph.**

I. Introduction

Vehicular Social Networks (VSNs) provide an emerging mobile communication paradigm that combines the features of Vehicular Ad-hoc Networks (VANETs) and Online Social Networks (OSNs), allowing the deployment of a wide variety of applications in smart cities [1]. In particular, the inheritance of social features in the study of vehicular systems has significant potential to improve the reliability and efficiency of the vehicular environment [2]. However, bringing social network aspects into vehicular scenarios comes up with enormous challenges due to the intrinsic nature of VSNs [3].

In spite of such challenges, the potential impact of VSN applications in smart environments justifies the importance of this paradigm and significant interests in the research community. A vehicle can be considered as a dynamic sensing platform that moves around an (urban) area, collects and processes the data locally to share with other entities over the network. This phenomenon represents the concept of *vehicular crowdsourcing*, where a multitude of entities helps to solve a wide range of problems sharing real-time information among themselves or with a centralized system like the Cloud.

Although crowdsourcing applications have tremendous benefits [4], the systems supporting them are also vulnerable to various attacks, such as malicious or selfish users seeking to disrupt or worsen the functioning of the system itself [5]. A *selfish* user may generate pseudonymous identities, named *sybils*, to increase its own benefits [6]; the sybils have more significant influence on the VSN-based crowdsourcing platforms [7], thus worsening application-specific goals [8]. For instance, if an event is related to a real vehicular traffic due to a crash, the sybil vehicles can use their influence to make *honest* drivers believe that there is no traffic in the nearby. Such a malicious behavior was, for example, exploited to hack *Google's Waze* generating false traffic information and deceiving genuine drivers [9].

Unfortunately, there exist no widely deployed tools to corroborate selfish/malicious behaviors due to the lack of robust authentication mechanisms in VSNs. A possible approach to address this challenging issue is to leverage on the trustworthy participants by adopting a proper *User Recruitment Policy* (URP) that aims to identify the best set of participants to achieve application-specific goals. However, not all URPs are suitable for VSN applications. For example, consider a system that assigns a sensing task to the users in a specific urban area and uses a distance-based URP. Such a recruitment policy leads to a situation where all participants within the event radius are selected to release feedback. Then, malicious participants could generate a large number of sybil to gain a disproportionately large influence on the network. This implies that the distance-based URP alone is not enough to provide high quality services in the network.

Our Contributions: This paper describes a novel URP for VSN-based crowdsourcing applications that not only takes care of event radius but also the reduction of sybil vehicles in the network. In essence, the contributions of this paper are as follows: (i) We propose a novel sybil detection technique, called *SybilDriver*, that combines the concept of proximity graph, the Louvain community detection algorithm [10], and Random Forests (RFs) in order to detect and filter out sybil vehicles within the VSN. (ii) We use a well-known vehicular dataset for extensive performance evaluation of SybilDriver.

[1]This work was done while F. Concone, F. De Vita and A. Pratap were at the Missouri University of Science and Technology, Rolla, USA.

After demonstrating the impact of proximity graph has on the Louvain method, we show that our approach outperforms two existing sybil detection techniques employed in the OSN domain, achieving strong performance even when the number of vehicles is quite limited. (iii) To the best of our knowledge, we are the first to apply the OSN-based approaches to a real vehicular dataset. Our analysis suggests that most of the existing techniques for sybil detection on OSNs are not suitable for vehicular crowdsourcing applications because of some assumptions that are usually valid in OSNs but not in VSNs [2], thus demonstrating empirically their low detection rate when applied to the vehicular domain.

The paper is organized as follows. Section II briefly summarizes existing works on sybil detection and sybil-based URP. Section III presents the system and threat model, while Section IV describes the proposed *SybilDriver* approach. Finally, Section VI concludes the paper with directions of future works.

II. Related Work

This section reviews the literature on sybil detection in VANETs and OSNs, and sybil-based URPs.

Sybil Detection in VANETs: In the recent past, different approaches have been proposed to defend against sibyls in vehicular networks. These approaches can be divided into three main categories: resource testing-based, trusted certification-based, and physical measurement-based. The methods in the first category rely on the assumption that the malicious entities are constrained by the computation or communication capability to deal with time-consuming tasks. However, these methods may no longer be suitable for next-generation applications because malicious entities own considerable computational capabilities. More recent solutions are the trusted certification-based methods (e.g., [11], [12]) that adopt cryptographic techniques to establish trust among all entities. The main drawback of such methods is the need for a centralized authority to be trusted by all participants to generate and manage public keys, certificates and digital signatures. Moreover, they require a widespread deployment of Road-Side Units (RSUs) to disseminate these certificates. Finally, the most recent solutions leverage on measurement-based methods. In this category, the most commonly used techniques are based on Received Signal Strength Indicator (RSSI), such as signal strength distribution [13], position verification [14], similarity comparison [15], or power control [16] to discover sybils within the VANET. In spite of the above approaches to preventing sybil attacks in VANETs, there is still a need for further enhancements because those proposed solutions bypass several constraints or vulnerabilities due to strongly dependence on RSUs, trustworthiness of vehicles, trust and verification of data context [17], [18]. The emergence of VSNs has enabled these constraints to pave the way for next-generation systems that can take advantage of VANET characteristics and sibyl detection techniques adopted in the OSN domain.

Sybil Detection in OSNs: The sybil detection techniques proposed in recent years have mainly leveraged the structural properties of the social graph to discriminate sybil accounts from legitimate users. In [19], the authors proposed an efficient and effective sybil community detection algorithm, called SybilExposer, which exasperated on the properties of social graph communities to rank them according to the structural features that characterize sybil and honest regions. However, it does not deal with non-community sybil nodes as well as edges that occur between the sybil communities. Another technique based on the social graph feature is SybilRank [20], where the users are ranked based on their perceived likelihood of being sybil. Such a scheme is considered efficient due to high accuracy and lower computational complexity of $O(|V|log|V|)$, where V is the set of vertices in the graph. Several other works following the SybilRank guidelines achieved best results in terms of accuracy detection, while maintaining the same complexity. For example, SybilRadar [21] outperforms SybilRank by proposing a novel mechanism that is able to detect nodes with weak trust relationships against sybil accounts. It is also shown that SybilRadar has much better detection accuracy than other methods. As discussed in Section V, SybilExposer, SybilRadar, and many other existing methods adopting similar guidelines are not suitable for vehicular crowdsensing applications because of some assumptions that are usually valid for OSNs but not for VSNs [3]. They are due to: (i) the limited number of nodes and dynamic nature of VSNs, (ii) occurrence of social connections among members even if they do not know each other, and (iii) users communicating through intermittent and unreliable inter-vehicle connections.

Sybil-based User Recruitment Policy: Regardless of their applicability and effectiveness, the above approaches aim only at sibyl recognition without considering that such entities can be used for malicious purposes in crowdsourcing applications. Only a few works in the literature attempt to address this dual aspect. For instance, the authors in [22] proposed a cryptography-based scheme that not only protects the user's privacy, but also allows the sybil users to be recognized when subscribing to the task. Although this scheme is effective from the view point of privacy protection, it has limitations because the sybil recognition process assumes that the request to join a task can only be made by a user that is correctly registered to the system. In other words, if an entity is able to register itself to the system, then it will consider the malicious entity as genuine, thus performing malicious activities.

Similarly, in [23], the authors stress the importance of removing sibyls during crowdsourcing campaigns and design a sybil-proof online incentive mechanism to detect the sybil attack during the user recruitment. Here, the objective of the sybil attackers is to maximize the utility that is calculated as the difference between the payment and the cost. Although the described approach is effective, it does not deal with attackers aiming to influence and manipulate the network, and relies on payments, a typical behavior in crowdsourcing scenarios.

III. System and Threat Model

Our *system model* captures a particular urban area containing (potential) participants and sensing task, as depicted in Fig. 1. Each participant is registered to a crowdsourcing application

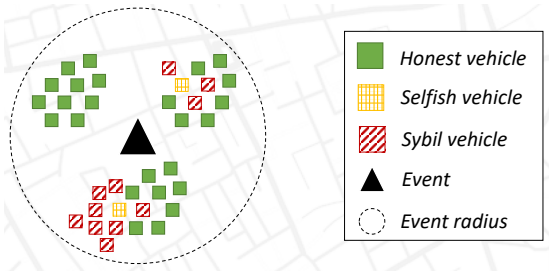


Fig. 1: System model.

equipped with a Reputation Management Engine (RME) that, based on the past interactions, takes care of assigning a trust score to each participant. Specifically, we identify three main actors in the proposed model, namely *honest vehicles*, *selfish vehicles*, and *sybil vehicles*. The first category consists of participants who physically exist and want to release real feedback about a sensing task. The second category of participants also physically exist in the model but they act as internal attackers by creating sybil vehicles. Finally, sybil vehicles are pseudonymous identities generated and employed by selfish vehicles to significantly impact the system.

In general, selfish participants aim to influence the network for deceiving the belief about a sensing task. To do so, our *threat model* assumes that the selfish vehicle is able to create an unlimited number of sybil vehicles, even if the generation of such pseudonymous identities requires computational resources. In particular, due to the constraints on vehicular network, we consider a selfish vehicle is able to inject sybil attacks only within one-hop range. In other words, a selfish vehicle knows its position and the relative positions of the neighboring vehicles in the proximity graph, due to proximity nature of equipped sensors. After the selfish vehicle knows who are its neighbors, it simulates sybil entities (i) by including them, as neighboring vehicles, in its own message shared with Fog entities, and (ii) by forging new fake messages which allow to mimic the existence of sybil entities as if they were real vehicles (refer to Sec. IV).

To deal with this malicious behavior, we are proposing an URP that leverages on a Fog computing architecture [24], in which information retrieved by the vehicular network is exploited to discover and filter-out sybil vehicles. Specifically, a sensing task created by the Cloud is submitted to the more appropriate Fog entities, and then to the n vehicles in the vehicular network satisfying the event radius. These vehicles are responsible for detecting other vehicles and share such data with the intermediate Fog entities at the upper layer. At the intermediate level, the Fog entities (Road-Side Units or RSUs) aim to construct the proximity graph by aggregating data obtained from the vehicular network. Note that the Fog entities can also exchange information with other devices at the same layer via a high capacity mesh network. Information produced at the Fog layer is sent to the Cloud responsible for resource-consuming analysis of data. First, the Cloud groups vehicles into communities, and filters those that do not fit in specific requirements. Next, the remaining communities are analyzed by employing a Smart Sybil Detection approach which returns the set of selected (trusted) participants (vehicles) that can release feedback about the published event.

IV. *SybilDriver*: The Proposed Approach

In the following sections, we describe each module involved in the proposed approach.

A. Vehicular Network

A VSN is composed of interconnected vehicles that can interact with each other using short-range communication protocols. We assume that two vehicles v_i and v_j in the network can communicate with each other only if the distance between them is smaller than or equal to a given threshold θ i.e., $d(v_i, v_j) \leq \theta$.

Once the communication is established, the vehicle v_i asks for the list of neighbors of v_j, and vice-versa, to calculate the *Jaccard similarity* index, as defined in the following:

$$J_{i,j} = \frac{|\zeta(v_i) \cap \zeta(v_j)|}{|\zeta(v_i) \cup \zeta(v_j)|}. \quad (1)$$

where $\zeta(\cdot)$ represents the set of neighbors of a vehicle. The idea behind this similarity measure is that the more the neighbors between two vehicles, the stronger will the connection be. Even if the time complexity of computing Jaccard similarity is lower than other techniques in the literature [25], assigning a weight to each communication link may require a very long time for mid-size to large-size networks. To overcome this limitation, we utilize a distributed approach where each vehicle is able to calculate its own Jaccard similarity index using information shared by other vehicles within its proximity range.

At the end of above process, each vehicle sends a message Msg to the closest Fog entity enabling the creation of the proximity graph. The message is formatted using the following JSON-like syntax:

$$\begin{aligned} Msg = \{&k_1\colon\ ID_{v_i}\,, \\ &k_2\colon\ [\{k_3\colon\ ID_{v_1}\,,\ k_4\colon\ J_{i,1}\}\,, \\ &\quad\ldots\,, \\ &\quad\{k_3\colon\ ID_{v_{|\zeta(v_i)|}}\,,\ k_4\colon\ J_{i,|\zeta(v_i)|}\}]\} \end{aligned}$$

where k_1, k_2, k_3, and k_4 respectively represent keys to access the vehicle ID, the list of neighboring vehicles, the neighbor ID, and the estimated Jaccard similarity.

B. Proximity Graph Generation

To meet the latency and scalability requirements of VSNs, the generation of the proximity graph is a task performed by the Fog entities that maintain the set of messages, $\mathbb{M} = \{Msg_1, Msg_2, \ldots, Msg_{|\mathbb{M}|}\}$, shared by the vehicles. These messages are used to generate a proximity graph $G \triangleq (V, E, W)$ by considering a certain time slot window, where V is a set of vehicles, $E \subseteq V \times V$ represents the communication links between node pairs, and W is the set of weights assigned to the links.

Algorithm 1 presents the pseudo-code of the proximity graph generation which takes $\mathbb{M}$ as the input. For each message in $\mathbb{M}$, the algorithm extracts the vehicle ID, its neighbors list, and

Algorithm 1: Proximity Graph Generation

Data: $\mathbb{M} = \{Msg_1, Msg_2, \ldots, Msg_{|\mathbb{M}|}\}$
Result: Proximity Graph, G

1 $(V, E, W) \leftarrow (emptyList(), emptyList(), emptyList())$;
2 **foreach** $msg \in \mathbb{M}$ **do**
3 $v_i \leftarrow msg.get(k_1)$;
4 $neighborsList \leftarrow msg.get(k_2)$;
5 $V.add(v_i)$;
6 **foreach** $obj \in neighborsList$ **do**
7 $v_j \leftarrow obj.get(k_3)$;
8 $msg_{v_j} \leftarrow \mathbb{M}.getMsg(v_j)$;
9 **if** $E.exist(v_i, v_j)$ **then**
10 $J_{i,j} \leftarrow obj.get(k_4)$;
11 $J_{j,i} \leftarrow W.getFromEdge(v_i, v_j)$;
12 $W.update((J_{j,i} + J_{i,j})/2)$;
13 **else**
14 $J_{i,j} \leftarrow obj.get(k_4)$;
15 **if** $v_i \in msg_{v_j}.get(k_2)$ **then**
16 $W.add(J_{i,j})$
17 **else**
18 $W.add((J_{i,j} - 1)/2)$;
19 $E.add(createEdge(v_i, v_j))$;
20 $G \leftarrow createGraph(V, E, W)$;

adds the vehicle to the vertex set V (lines 2-5). At this point, the algorithm dynamically updates the other two sets, i.e., E and W, by implementing the following logic (lines 8-19). Let v_i and v_j be two vehicles. Three different cases can occur: (i) an edge between v_i and v_j already exists (lines 9-12); (ii) no edge exists between them and $v_i \in \zeta(v_j)$ (lines 15-16); and (iii) no edge exists between v_i and v_j and $v_i \notin \zeta(v_j)$ (lines 17-18). Specifically, the first case represents a scenario where both v_i and v_j are honest vehicles or both are sybil vehicles, i.e., $v_i \in \zeta(v_j)$ and $v_j \in \zeta(v_i)$. The second one represents a scenario where a new link has to be added; while the third one is a special case where a sybil node is trying to attach itself to a honest vehicle. To manage the third case, the algorithm subtracts 1 to highlight those links between a honest vehicle and a sybil vehicle. These three cases can be summarized in the following weight function:

$$w_{i,j} = \begin{cases} \frac{J_{i,j}+J_{j,i}}{2} & \text{if } e_{i,j} \in E \quad \text{(i)} \\ J_{i,j} & \text{if } e_{i,j} \notin E \text{ and } v_i \in \zeta(v_j) \text{ (ii)} \\ \frac{J_{i,j}-1}{2} & \text{if } e_{i,j} \notin E \text{ and } v_i \notin \zeta(v_j) \text{ (iii).} \end{cases} \quad (2)$$

Upper and lower bounds of the weight are demonstrated in the following Lemma 1.

Lemma 1. *The weight of the link between any honest vehicle v_i and a sybil vehicle v_j is bounded by $-0.5 \leq w_{i,j} \leq 0$.*

Proof. By definition, the Jaccard similarity index of any two vehicles is bounded as $0 \leq J_{i,j} \leq 1$. The minimum value is obtained when there are no common neighbors, whereas the maximum value is obtained when all the neighbors are in common. If v_i and v_j are honest and sybil vehicles, respectively, then Algorithm 1 follows Case (iii) of Eq. 2. At the lower bound of Jaccard similarity index $J_{i,j}$, we can estimate $-0.5 \leq w_{i,j}$. At the upper bound of $J_{i,j}$, the weight is bounded as $w_{i,j} \leq 0$. Thus, the weight between vehicles v_i and v_j is bounded as $-0.5 \leq w_{i,j} \leq 0$. □

The proposed approach is novel in two folds: (a) Our scheme is distributed in the sense that each vehicle estimates its own Jaccard similarity index with respect to the neighboring vehicles, thus reducing the computational complexity; (b) Based on Lemma 1, the system is able to discover links between the sybil and honest vehicles by improving the performance of community detection and ranking algorithm as will be better described in Section V.

C. Community Detection and Ranking

In our model, we used the Louvain method to extract the community set $\mathbb{C} = \{C_1, C_2, \ldots, C_{|\mathbb{C}|}\}$ from the proximity graph, G. Once communities are detected, the system is ready to select and remove them from the set $\mathbb{C}$ based on the size of each community, and a *trust* score owned by the system for each vehicle within that particular community. Specifically, the trust value is calculated by the RME (mentioned in Section III) that assigns a score in a range between 0 and 1 where 0 means that the user has no trust, whereas 1 indicates that user is totally trusted. Users are initialized with a 0.5 trust score and updated over time (increased or decreased) based on the analysis performed by the RME. If the trust score of a vehicle is higher than a given threshold, τ_{trust}, then the system can trust that particular vehicle.

Based on above concepts, let $S_i \subseteq C_i$ be the set of vehicles having a trust score higher than τ_{trust}. Then, for each community, we modeled the rank, R_i, as follows.

$$R_i = \alpha \frac{|S_i|}{|C_i|} + (1 - \alpha) \frac{|C_i|}{|V|} \quad (3)$$

where $0 \leq \alpha \leq 1$, $|C_i|$ is the cardinality (i.e., the number of vehicles in C_i) of the community C_i, and $|V|$ is the number of vehicles in the proximity graph G. The idea behind this ranking index lies in preferring bigger communities with a higher number of trusted vehicles.

Algorithm 2 describes the pseudo-codes of the Community Detection and Ranking algorithm. The inputs of the algorithm are the proximity graph G and thresholds Thr, τ_{trust}, and τ_{rank}. The threshold Thr is used to verify whether the cardinality of $\mathbb{C}$ is in the order of the number of network nodes, whereas τ_{rank} is used for the pivot detection. Here, the pivot helps the systems to reduce the search space by preferring those communities having higher rank values.

Lines 1-3 initialize variables used in the algorithm. Once the communities are extracted by Louvain (line 4), the algorithm checks if the cardinality of $\mathbb{C}$ is in the order of the number of nodes which constitute the graph G (lines 5-8). In particular, if the condition is satisfied (line 5), the entire graph is considered as a single large community (line 6); otherwise the algorithm stores all communities discovered by Louvain method (line 8). Next, the algorithm iterates over each community (line 9), and assigns a rank by calling the function $setRank()$ returning R_i, defined in Eq. 3 (line 10). At the end of this procedure, the

Algorithm 2: Community Detection and Ranking

Data: $G, Thr, \tau_{trust}, \tau_{rank}$
Result: Community $Ranks$

```
1  Ranks ← emptyList();
2  Communities ← emptyList();
3  V ← G.getVertexes();
4  ℂ ← G.communityDetection();
5  if |G| − |ℂ| ≤ Thr then
6  |  Communities ← G;
7  else
8  |  Communities ← ℂ.asList();
9  foreach C_i ∈ Communities do
10 |  Ranks.add(i, setRank(|V|, C_i, τ_trust));
11 Ranks ← sortByRankInAscendingOrder(Ranks);
12 pivot ← 0;
13 foreach obj ∈ Ranks do
14 |  if obj.getNext() ! = NULL then
15 |  |  if obj.getNext().getR() − obj.getR() < τ_rank then
16 |  |  |  pivot ← pivot + 1;
17 |  |  else
18 |  |  |  break;
19 if pivot < |Ranks| then
20 |  Ranks ← Ranks.subList(pivot, |Ranks|)
```

Rank list is sorted in ascending order (line 11). At this point, the algorithm iterates over each object in the list (line 13) and if the condition is not satisfied, the pivot is set (line 16) and the for loop is interrupted (line 18). Finally, the algorithm takes the sublist from pivot's index to the end of the *Rank* list (line 20).

D. Smart Sybil Detection

The last phase of the proposed system leverages a machine learning technique to perform a more in-depth analysis for identifying sybil vehicles. The introduction of this further step is due to the fact that, after the filtering performed by Algorithm 2, the system cannot rely only on the trust score of each vehicle. In fact, if the trust score is used only as an index to determine the possibility for a vehicle to provide a feedback for a sensing task, it would be unfair to those vehicles which are new to the system.

The classification process is based on the Random Forest (RF) technique. As the algorithm is a supervised learning model, it requires data to be labeled and represented as a feature vector in order to learn how to classify unlabeled data. To this aim, in our scenario, the feature vector of each vehicle is built by considering a tuple made of the following four parameters: N_r, W_r, T, and N_t.

Given the node v_k belonging to the community C_i, N_r is the trusted neighbors ratio defined as:

$$N_r = \frac{|\zeta(v_k) \in S_i|}{|\zeta(v_k)|} \tag{4}$$

which expresses the fraction of trusted neighbors connected to a node. W_r is defined as the weights ratio and, for the generic k-th node, it is expressed as follows:

$$W_r = \frac{\sum_{j \in \zeta(v_k)} w_{k,j}}{|\zeta(v_k)|} \tag{5}$$

such a parameter provides very useful information about the nature of the neighborhood of a node (e.g., if the node has many sybils attached to it or not. Finally, T is the associated node trust value assigned by the reputation system, and N_t is the neighbors trust obtained by simply summing the trust values of the node neighbors. In this sense, the proposed RF algorithm performs a binary classification to determine if a node is to be considered as "honest" or "sybil" according to its extracted information as described above.

Theorem 1. *Computational complexity of the proposed scheme is $O(|V|^2 + Dk)$, where D is the depth of the classification tree and k is the number of trees, respectively.*

Proof. The computational complexity of the proposed scheme depends on the execution of Algorithm 1, Algorithm 2, and the classification process described so far. Algorithm 1 transforms the vehicular network into the proximity graph. This transformation may take $O(|V|^2)$ computational time in the worst-case scenario due to the fact that each vehicle explores its neighboring vehicles to connect over edges in the proximity graph. In the worst-case each vehicle could be a neighbor of the rest of vehicles and this results in $O(|V|(|V| - 1)) = O(|V|^2)$ computational complexity. Regarding Algorithm 2, the worst-case time complexity mainly depends on the execution of Louvain method and sorting ranks in ascending order. As stated by author in [10], the computational complexity of Louvain method is $O(|V|log|V|)$; while, sorting ranks in ascending order can also take $O(|V|log|V|)$ computational complexity (applying heapsort) in the worst case. Thus, the worst-case computational time complexity of Algorithm 2 is $O(|V|log|V|)$. Finally, the last phase of the proposed scheme is composed of feature extraction and classification. As a consequence, the worst case time complexity of the extracted features can be $O(|V|^2)$ because it is necessary to visit all the neighbors for each vehicle. Moreover, the classification process performed by RF depends on the depth of the classification tree (=D) and the number of trees (=k) created by the algorithm itself, i.e., $O(Dk)$ [26]. We can conclude that the overall complexity of Smart Sybil Detection phase is $O(|V|^2 + Dk)$.

On the basis of what has been discussed so far, the overall computational complexity is led by Smart Sybil Detection scheme, i.e. the computational complexity of the proposed scheme is $O(|V|^2 + Dk)$. □

V. Performance Evaluation

In order to evaluate the effectiveness of our technique, several experiments were performed. First, we describe how the experiments were conducted by means of some pre-processings and type of attacks. Then, we present the calibration of the parameters to implement the algorithms and a comparison between the results obtained while using different classifiers. Finally, the overall performances of the system we propose are tested on real data and compared with state-of-art techniques used in OSN applications.

A. Experimental Setup

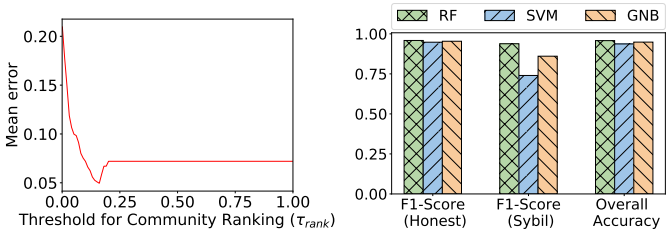
Pre-processing of chosen dataset: The effectiveness of our system is proved by using the Cologne dataset [27] that contains the mobility patterns of more than 700.000 individual car trips over a period of 24 hours, covering a region of about 400 square kilometers. This dataset is not directly applicable to our scenario and, therefore, it was necessary to pre-process the contained information before testing our method. In particular, we have generated multiple events over time, each of which has a randomly chosen location along the vehicles' routes. Considering a radius of about 60 meters for the campaign, we took all the vehicles that at a specific timestamp are inside it and set a inter vehicle distance of 5 meters so that two vehicles could physically detect each other. For each extracted VSN, we randomly selected a subset of vehicles that act as selfish, allowing them to generate a set of sibyl vehicles.

Attack strategies: We have considered [28] three types of attack. The first type of attack is very simple because the selfish node creates an edge between itself and a self-created sibyl node. Then, a more complex attack in which the selfish vehicle creates a set of sibyl nodes that can connect with honest vehicles is considered. This type of attack is typical of mobile networks where, due to the dynamism of these networks, attackers exploit the fact that nodes cannot keep connections with others for the long time. Finally, a selfish vehicle is able to launch a combination of the attacks described previously. In all the three cases, the main goal is to manage the local popularity and manipulate the sensing campaign in a negative way. During the experiments, the attacks are chosen randomly between the honest and sybil nodes and, when the attack requires the creation of sybil regions, these are generated by using the Barabasi-Albert's scale-free model [29]. To the best of our knowledge, there is no real-world evidence that the sybil regions are generated following a particular model, because the adversary can be arbitrarily malicious. However, the networks generated using a scale-free model allow to have the node degree following the power-law distribution.

Construction of dataset for the Random Forest: From the previous phase, the identity of the selfish node as well as the generated sybil nodes is known, so we were able to implicitly obtain a "labeled" VSN dataset for the RF. This training dataset contains very different VSN topologies to analyze due to the randomness of the events generated along the way as well as the selfish selection and the attacks. In particular, by extracting for each node the feature vector we described in Section IV-D, we ended up obtaining a dataset with a total number of $30,639$ labeled samples thus making the RF algorithm ready to be trained and tested.

B. Calibration and Choice of Classification Algorithm

Tuning the threshold τ_{rank}: Remark that this value is used to discover the pivot by the *Community Detection and Ranking* algorithm for the filtering process. To set this value accurately, we performed several experiments to obtain the best values aiming to minimize the mean error. Here, the mean error is



(a) (b)

Fig. 2: (a) Comparison between mean error and trust propagation threshold τ_{rank} (b) F-Score and accuracy compared for Random Forest (RF), Support Vector Machine (SVM), and Gaussian Naive Bayes (GNB) classifiers.

an average of all the errors computed, for each community $C_i \in \mathbb{C}$, as the ratio of the number of elements of minority class within the C_i and its size. With reference to Fig. 2a, our experiments have shown that when $\tau_{rank} = 0.16$, the curve reaches the minimum value possible and, consequently, we set $\tau_{rank} = 0.16$ in our experimental analysis.

Choice of classification algorithm: By analyzing the dataset constructed for the Random Forest (refer to Sec. V-A), as we would expect, the number of samples labeled as "sybil" is much lower than the number of "honest" ones which could cause wrong results during the training phase. To overcome this problem and properly train and validate the RF algorithm, a stratified 10-fold cross-validation [30] is adopted in order to guarantee the same amount of samples for each of the two classes in both train and test phases. Fig. 2b depicts a comparison in terms of accuracy and F1-score with respect to the two classes [31]. In general the RF performed better than the Support Vector Machine (SVM) and Gaussian Nayve Bayes (GNB) algorithms reaching a F1-score of 0.94 and 0.96 for the honest and sybil classes respectively and an overall accuracy of 96%. With regards to the SVM and GNB, these two algorithms performed well in terms of accuracy and F1-score for the honest class and resulted to be comparable with the RF, however, they reached very low values of F1-score for the sybil class (0.74 and 0.86 respectively) which make them not suitable for the sybil detection as they would cause a high level of false positives and false negatives. Moreover, in terms of time execution the RF results to be again the fastest algorithm. For all these reasons we selected the RF algorithm to perform the sybil detection.

C. Comparison with State-of-the-art Methods

The final set of experiments aims to make a direct comparison against some state-of-the-art methods. Each result has been obtained by repeating 1.000 times the logic introduced in Section V-A. At each iteration events are created, thus generating a network whose number of honests, selfish and sybils is not known in advance, due to the randomness of the parameters. Therefore, the final results are grouped depending on the number of nodes in the vehicular network and plotted by using the boxplots, usually used for describing the salient features of a distribution.

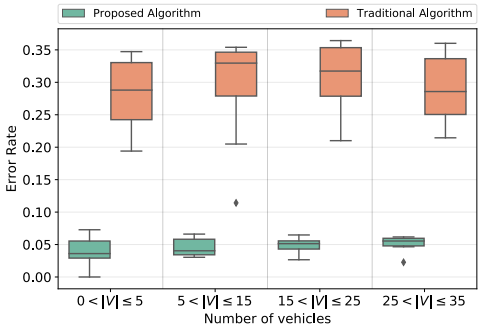


Fig. 3: Error rate comparison between the proposed community detection algorithm and the traditional Louvain method.

Discussion on the novel proximity graph: The first notable result achieved by the proposed system is improvement on the community detection because of the novel way the system assigns weights to the links (refer to Algorithm 1). Fig. 3 compares our approach with the traditional one by running the Louvain algorithm with varying number of honest and selfish vehicles. For better understanding, let us introduce the *Error rate* as the ratio of the majority class within the community cluster (i.e., honest or sybil) and the total number of vehicles inside the community. It is evident that our approach outperforms the original one in every case, thus ensuring a lower *Error rate*. Hence, we are able to obtain a better Louvain community detection as it aims to have relatively many positive links within communities, while for negative links the opposite holds.

Discussion on the sybil detection performance: To better demonstrate the potential of our approach we have conducted experiments aiming to make a direct comparison against two algorithms applied to the OSN domain SybilRadar [21] and SybilExposer [19]. These experiments are justified by the fact that our approach, similarly to these works, exploits the structural features to discover the accounts of the sibyl nodes within the OSNs. The Fig. 4a shows that SybilDriver requires more execution time than the two other techniques. This additional runtime is mainly due to the various phases of the proposed algorithm. In fact, SybilExposer is composed only by the community detection phase, while SybilRadar by the edge weighing and Random Walk-based trust propagation phases. However, even if the higher number of steps results in a higher runtime, the proposed technique requires an execution time less than seconds, an acceptable value for real-time vehicular crowdsensing applications. Regarding the sybil's detection, Fig. 4b and Fig. 4c show how our approach outperforms other methods when varying the number of vehicles inside the VSN. Indeed, by increasing the number of the vehicles, the system is always able to achieve high values of accuracy and F-Score. On the contrary, the main reason of the low detection performances of the other techniques is due to the number of nodes that, in an urban context, is much lower than in an OSN. As an example consider SybilRadar that uses a *Short Random Walk*-based Trust Propagation (SRW-TP) approach to identify sibyl vehicles. Given a graph and a trusted node (as start point), the SRW-TP selects a neighbor at random, moves to this neighbor, and propagates the trust value. This process continues until a stopping criterion is reached causing each vehicle to have a trust score. In a VSN, where the interconnected vehicles is quite limited, the probability that a sybil node will attach to multiple trust nodes is very high as well as the probability that the SRW-TP chooses the malicious node several times to propagate the trust value. This behavior causes a sibyl vehicle to have a very high chance to gain a high trust value and be misclassified as honest. Contrary to SybilRadar, SybilExposer does not use any trusted nodes in the social graph and does not depend on the SRW-TP. The idea is to use the node diversity and community diversity properties of the social graph with the aim to identify the sybil nodes as part of a sybil community. However, interconnections between nodes within a VSN are made on the fly regardless of any known relationship. This suggests that if selfish is able to create a sufficiently large community of Sibyl nodes the system will not be able to distinguish between honest and Sibyl communities, thus leading to an incorrect classification. We conclude that SybilExposer, SybilRadar, and many other works having similar characteristics, rely on assumptions (see Sec. II) that are not valid in VSN applications and inevitably lead to poor performance in terms of detection accuracy.

VI. Conclusion and Future Works

In this paper, we proposed a novel URP model aiming to reduce sybils in vehicular crowdsourcing campaigns. The core of this URP is *SybilDriver*, a sybil detection approach based on a combination of proximity graph, Louvain and RF techniques to recruit the best set of participants from VSNs by removing sybil entities and, thus, reducing the possibility of false reports. Through extensive experimental analysis, we demonstrated that the proposed model outperforms other state-of-the-art schemes usually adopted in OSN applications. The results obtained by the proposed system demonstrate that the majority of the generated sybil vehicles is filtered by the *Community Detection and Ranking*, and *Smart Sybil Detection* algorithms. Thus, even if selfish vehicles are able to generate sibyl entities and are selected by URP, our system, by filtering-out as much as possible the generated sibyls, is able to strongly reduce their influence in the vehicular network, making their attempt to manipulate the belief about a sensing task ineffective. The performance comparison also demonstrates that the response time is suitable for real-time VSN applications.

Our future work will be devoted at improving the performance of the system from different aspects. First, the current output of the system could be used to conduct a more accurate offline analysis to enhance the detection of sybil vehicles. In addition, since the proposed algorithm relies on a reputation system, we plan to extend our approach by integrating a trust management module to estimate the user's trustworthiness and discourage malicious behavior in real-time VSN applications. Finally, we plan to propose an extension of this work by including quality contributions of every recruited participant to determine whether the contribution is true or false [32].

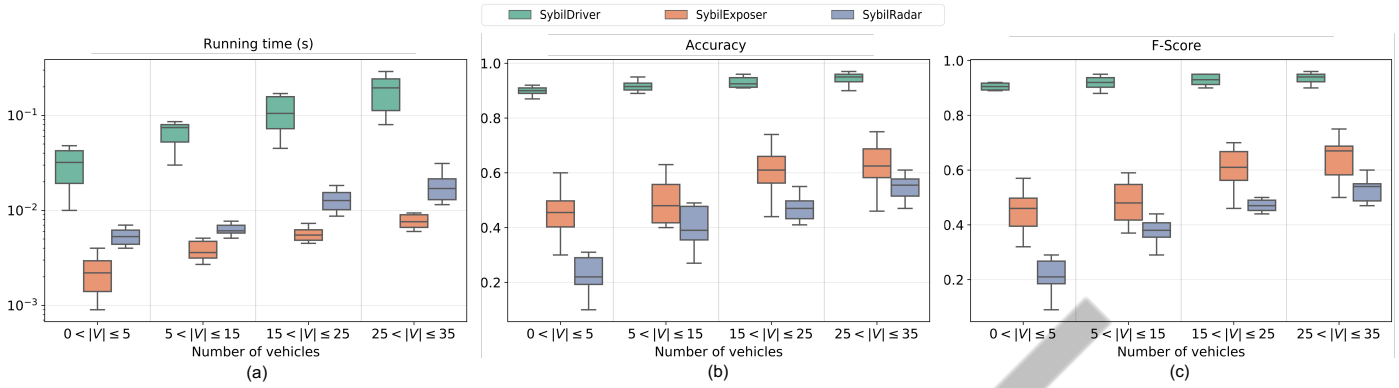


Fig. 4: Running time: (a) Accuracy and (b) F-Score comparisons for different approaches for varying number of vehicles.

ACKNOWLEDGMENTS

The work of S. K. Das was partially supported by NSF grants SaTC-2126619, CNS-1818942, DGE-1433659, OAC-1725755, and OAC-2104078.

REFERENCES

[1] D. V. Le, C. Tham, and Y. Zhu, "Quality of Information (QoI)-aware cooperative sensing in vehicular sensor networks," in *IEEE PerCom Workshops*, 2017, pp. 369–374.

[2] A. Rahim, X. Kong, F. Xia, Z. Ning, N. Ullah, J. Wang, and S. K. Das, "Vehicular social networks: A survey," *Pervasive and Mobile Computing*, vol. 43, pp. 96 – 113, 2018.

[3] A. M. Vegni and V. Loscri, "A survey on vehicular social networks," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 2397–2419, 2015.

[4] F. Concone, G. L. Re, and M. Morana, "A fog-based application for human activity recognition using personal smart devices," *ACM Trans. Internet Technol.*, vol. 19, no. 2, Mar. 2019.

[5] M. Morana, F. Concone, and G. Lo Re, "Smcp: a secure mobile crowdsensing protocol for fog-based applications," *Human-centric Computing and Information Sciences*, vol. 10, 2020.

[6] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Third international symposium on information processing in sensor networks*. IEEE, 2004, pp. 259–268.

[7] Q. Li, S. Zhu, and G. Cao, "Routing in socially selfish delay tolerant networks," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[8] C. Fiandrino, B. Kantarci, F. Anjomshoa, D. Kliazovich, P. Bouvry, and J. Matthews, "Sociability-driven user recruitment in mobile crowdsensing internet of things platforms," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[9] M. Sun, M. Li, and R. Gerdes, "Truth-aware Optimal Decision-making Framework with Driver Preferences for V2V Communications," in *IEEE Conf. on Communications and Network Security (CNS)*, 2018, pp. 1–9.

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[11] C. Iwendi, M. Uddin, J. A. Ansere, P. Nkurunziza, J. H. Anajemba, and A. K. Bashir, "On detection of sybil attack in large-scale vanets using spider-monkey technique," *IEEE Access*, vol. 6, pp. 47 258–47 267, 2018.

[12] S. M. Faisal and T. Zaidi, "Timestamp based detection of sybil attack in vanet." *IJ Network Security*, vol. 22, no. 3, pp. 397–408, 2020.

[13] B. Yu, C.-Z. Xu, and B. Xiao, "Detecting sybil attacks in vanets," *Journal of Parallel and Distributed Computing*, vol. 73, pp. 746 – 756, 2013.

[14] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in vanets," in *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. ACM, 2004, pp. 29–37.

[15] Y. Yao, B. Xiao, G. Wu, X. Liu, Z. Yu, K. Zhang, and X. Zhou, "Voiceprint: A novel sybil attack detection method based on rssi for vanets," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 591–602.

[16] Y. Yao, B. Xiao, G. Yang, Y. Hu, L. Wang, and X. Zhou, "Power control identification: A novel sybil attack detection scheme in vanets using rssi," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2588–2602, 2019.

[17] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "Vanet security challenges and solutions: A survey," *Vehicular Communications*, vol. 7, pp. 7 – 20, 2017.

[18] S. Sharma and B. Kaushik, "A survey on internet of vehicles: Applications, security issues & solutions," *Vehicular Communications*, vol. 20, p. 100182, 2019.

[19] S. Misra, A. S. M. Tayeen, and W. Xu, "SybilExposer: An effective scheme to detect Sybil communities in online social networks," in *IEEE International Conf. on Communications (ICC)*, 2016, pp. 1–6.

[20] 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*. USENIX Association, 2012, pp. 15–15.

[21] D. Mulamba, I. Ray, and I. Ray, "SybilRadar: A graph-structure based framework for sybil detection in on-line social networks," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2016, pp. 179–193.

[22] J. Shu, X. Liu, K. Yang, Y. Zhang, X. Jia, and R. H. Deng, "Sybsub: Privacy-preserving expressive task subscription with sybil detection in crowdsourcing," *IEEE Internet of Things Journal*, pp. 3003–3013, 2019.

[23] J. Lin, M. Li, D. Yang, and G. Xue, "Sybil-proof online incentive mechanisms for crowdsensing," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2438–2446.

[24] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[25] P. Srilatha and R. Manjula, "Similarity index based link prediction algorithms in social networks: A survey," *Journal of Telecommunications and Information Technology*, 2016.

[26] X. Solé, A. Ramisa, and C. Torras, "Evaluation of random forests on large-scale classification problems using a bag-of-visual-words representation." in *CCIA*, 2014, pp. 273–276.

[27] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2014.

[28] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[29] A.-L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A: Statistical Mechanics and its Applications*, vol. 281, pp. 69–77, 2000.

[30] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2009.

[31] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

[32] R. P. Barnwal, N. Ghosh, S. K. Ghosh, and S. K. Das, "Publish or drop traffic event alerts? quality-aware decision making in participatory sensing-based vehicular *CPS*," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 1, pp. 1–28, 2019.