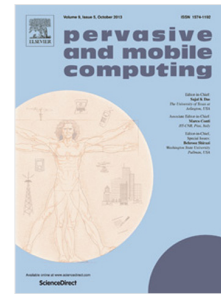


## Journal Pre-proof

A fog-assisted system to defend against Sybils in vehicular crowdsourcing

Federico Concone, Fabrizio De Vita, Ajay Pratap, Dario Bruneo,  
Giuseppe Lo Re, Sajal K. Das



PII: S1574-1192(22)00050-5  
DOI: <https://doi.org/10.1016/j.pmcj.2022.101612>  
Reference: PMCJ 101612

To appear in: *Pervasive and Mobile Computing*

Received date : 14 November 2021  
Revised date : 15 March 2022  
Accepted date : 7 May 2022

Please cite this article as: F. Concone, F. De Vita, A. Pratap et al., A fog-assisted system to defend against Sybils in vehicular crowdsourcing, *Pervasive and Mobile Computing* (2022), doi: <https://doi.org/10.1016/j.pmcj.2022.101612>.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2022 Elsevier B.V. All rights reserved.

# A Fog-assisted System to Defend against Sybils in Vehicular Crowdsourcing

Federico Concone<sup>a,\*</sup>, Fabrizio De Vita<sup>b</sup>, Ajay Pratap<sup>c</sup>, Dario Bruneo<sup>b</sup>, Giuseppe Lo Re<sup>a</sup>, and Sajal K. Das<sup>d</sup>

<sup>a</sup>Department of Engineering, University of Palermo, Italy

<sup>b</sup>Department of Engineering, University of Messina, Italy

<sup>c</sup>Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, India

<sup>d</sup>Department of Computer Science, Missouri University of Science and Technology, Rolla, USA

## Abstract

Technological advancements in vehicular transportation systems have led to the growth of novel paradigms, in which vehicles and infrastructures collaborate to infer high-level knowledge about phenomena of interest. *Vehicular Social Network* (VSN) is one such paradigm in which vehicular network entities are considered as part of an Online Social Network (OSN), paving the way for new services derived from social context. Although vehicular crowdsourcing has tremendous benefits, its deployment in real systems requires to solve important challenges including defense against Sybil attacks. This paper proposes a novel fog-assisted system that uses *SybilDriver* to minimize the presence of Sybil entities in VSN-based crowdsourcing applications. The proposed system exploits the characteristics of Vehicular Ad-hoc NETWORKs (VANETs) and OSNs to effectively recognize Sybils, and the adoption of fog computing helps reduce the overall network overhead by processing data closer to the vehicles. We perform detailed experiments on real-world publicly available datasets primarily to assess the effectiveness of *SybilDriver* against different Sybil attack strategies. Our experimental results show that *SybilDriver* detects Sybils with higher performance than state-of-the-art techniques under different settings. Furthermore, an evaluation of the fog architecture in terms of message complexity demonstrates low impact on the network overhead.

**Keywords:** Vehicular Social Network; Crowdsourcing; Sybil detection; Truthfulness; Fog Computing.

## 1. Introduction

Technological advancements in recent years have led to the development of *smart cities* that exploit data collected from heterogeneous sources to enhance the citizens' quality of life. In a smart city, modern vehicles play a fundamental role beyond being a means of transport. Precisely, a vehicle acts as a dynamic

\*Corresponding author. The bulk of the paper was developed while F. Concone, F. De Vita and A. Pratap were visiting the Missouri University of Science and Technology.

Email addresses: federico.concone@unipa.it (Federico Concone), fdevita@unime.it (Fabrizio De Vita), ajay.cse@iitbhu.ac.in (Ajay Pratap), dbruneo@unime.it (Dario Bruneo), giuseppe.lore@unipa.it (Giuseppe Lo Re), sdas@mst.edu (and Sajal K. Das)

5 sensing platform roaming in an urban area, collecting and processing data locally to share with other entities in the network through wireless communication platforms [1]. This is the concept behind Vehicular-Ad-hoc NETWORKS (VANETs) in which vehicles and infrastructures collaborate to exchange messages and obtain high-level information about a specific phenomenon. However, a VANET only considers local connections between vehicles within a certain communication range, without considering that these connections form a  
 10 social network of virtually connected entities to share a common purpose or benefit.

Vehicular Social Networks (VSNs) offer an emerging paradigm that combines the features of VANETs and Online Social Networks (OSNs), paving the way to the deployment of important applications in smart cities [2]. For example, Apple launched the Carplay system for vehicles in March 2014, with which drivers can participate in social activities easily and safely; Google joined the development of VSNs by releasing  
 15 Android Auto in June 2014 [3]. Such tools contribute to the technology that makes VSNs a perfect and ubiquitous platform to allow users (drivers and passengers) to not only share local data but also benefit from assistance through the use of social contexts. This paradigm leads to the concept of *vehicular crowdsourcing*, where a multitude of entities helps solve a wide range of problems by sharing real-time information among themselves, or with other entities in the network. However, the deployment of VSN-based crowdsourcing  
 20 applications in real systems pose important challenges, including entities that seek to disrupt or worsen the functioning of the system itself. A *malicious* entity may generate pseudonymous identities, called *Sybil*s, to increase their own benefits [4] while worsening the application-specific goals [5]. For instance, if an event is related to real vehicular traffic due to a crash, *Sybil*s can use their influence to make *honest* drivers believe that there is no traffic nearby.

25 Such a malicious behavior was indeed exploited to hack *Google's Waze* system by generating false traffic information and deceiving the genuine drivers [6]. Unfortunately, there exists no widely deployed tools to corroborate this type of behaviors due to the lack of robust authentication mechanisms in VSNs, and in general VANETs. A possible approach to address this challenge is to leverage on trustworthy participants by adopting a proper *recruitment policy* allowing the selection of the best set of participants, who will improve  
 30 the quality of service provided by the system. A naïve approach is to apply a distance-based recruitment policy and select all participants within the event radius. However, this approach is not effective because malicious participants could generate a large number of *Sybil*s with fake GPS positions, and successfully worsen the quality of service by providing false reports. Thus, the distance-based (or any other naïve approach) alone is not effective to provide high-quality reliable services in vehicular crowdsourcing.

35 This paper presents an extension of *SybilDriver*, a new technique that we recently proposed in [1] to detect *Sybil* entities generated within the radius of a specific vehicular crowdsourcing event. The underlying

idea of SybilDriver is to exploit the features of VANETs and OSNs to discover and filter out Sybils during the recruitment phase. Starting from VANETs, the SybilDriver exploits the concept of a proximity graph via *proximity interactions* among the vehicles present in the same zone of interest. Then, it exploits two inter-related elements often employed in the social network domain [7, 8], namely the *community detection algorithm* and a *similarity index* between the nodes. Community detection is the process by which nodes with similar characteristics are grouped together based on some criteria ((e.g., common interests, neighborhood, or interaction patterns). For similarity index, we adopt the *Jaccard index* that models the connections between the nodes based on their neighborhood – the more the neighbors between two nodes, the stronger is the connection. Finally, a machine learning (ML) technique is used to further improve the accuracy of Sybils detection in the VSNs.

**Contributions.** This work is an extension of our recently published paper in the 7th IEEE International Conference on Smart Computing (SMARTCOMP 2021) [1]. Compared to the conference version, we make the following major contributions:

1. We introduce a fog architecture, in which different entities in different layers cooperate to identify Sybils in vehicular crowdsourcing applications. Data processing is distributed among vehicles and fog entities with a goal to reduce the overall network overhead and guarantee real-time detection, whereas a remote cloud infrastructure maintains an overall consistent view of the participants' reputation scores as well as the general model of the ML algorithm.
2. We define a new module for SybilDriver that allows monitoring of the *Quality of Information (QoI)* for published events. Such a metric is modeled using the truthfulness score of an event that is defined considering several aspects ranging from SybilDriver performance to the degree of belief and uncertainty associated with the event itself.
3. We evaluate the performance of SybilDriver by conducting new experiments on two popular datasets and considering a broader set of attack strategies. The experimental study demonstrates the capability of our approach to detecting Sybils and exhibits good performance compared to other state-of-the-art techniques under different settings. We also analyze the QoI metric and the message complexity between various entities of the fog-based architecture, in terms of *message preparation time*, *message size*, and *transmission delay*. Finally, we provide a comprehensive analysis of the effectiveness of SybilDriver against individual attack strategies, and highlight their weaknesses and strengths.

Further differences with [1] include lemmas, theorems, system and threat models, an illustrative example, and discussing additional recent works on Sybil detection in VANETs, OSNs, and Crowdsourcing.

Table 1: Summary of related works presented in Section 2.

Domain	Category	References	This paper
VANET	<i>Resource Testing</i>	[4, 11]	✓
	<i>Trusted Certification</i>	[12, 13, 14]	
	<i>Physical Measurement</i>	[15, 16, 17, 18, 19]	
OSN	<i>Behavior-based</i>	[20, 21]	✓
	<i>Graph-based</i>	[22, 23, 24, 25]	
	<i>Deep Learning</i>	[26, 27]	
Crowdsourcing	<i>Incentive mechanisms</i>	[28, 29]	✓
	<i>Other</i>	[30, 31]	

**Organization.** The remainder of the paper is organized as follows. Section 2 summarizes related work on Sybil detection in different research areas. Section 3 describes the system model and the threat model in terms of the attacker goal, knowledge, and capabilities. Section 4 introduces the fog architecture of the proposed system, and Section 5 presents the main modules of SybilDriver, from the proximity graph generation to the QoI model. After providing an illustrative example in Section 6, we present detailed experimental evaluation in Section 7 to demonstrate the effectiveness of the proposed approach. Section 8 concludes the paper with directions of future research.

## 2. Related Work

This section reviews the existing literature about Sybil detection with a particular focus on VANET, OSN, and Crowdsourcing scenarios (see Table 1). The literature suggests that there are no universally accepted techniques, and efforts need to be done to address this challenge. Defense in VANETs is still a work-in-progress [9], while techniques in OSNs do not meet the vehicular constraints [10]. These limitations may be solved by exploiting information gathered from the physical world, i.e., VANETs, and combining it with the effective Sybil detection techniques adopted in OSNs. This represents the logic behind SybilDriver that takes advantage of the features of the different domains, as shown in Table 1.

### 2.1. Defend against Sybils in VANETs

The approaches in vehicular networks can be divided into three main categories: *Resource Testing*, *Trusted Certification*, and *Physical Measurement*.

*Resource Testing* is the most commonly adopted solution to face Sybil attacks. The basic principle is that the malicious entities are constrained by the computation or communication capabilities to deal with time-

consuming tasks. A verification agent checks the resources associated with an entity in the network and, if there are any discrepancies, it is reported as compromised. For instance, the authors in [4] assumed physical  
 90 devices have only one radio incapable of transmitting and receiving messages on more than one channel at any given time. In [11], the authors present an admission control mechanism that challenges nodes looking to join the network through computational puzzles. Nodes completing the puzzles are provided proof of the vetted identity. Resource testing-based techniques are considered a minimal defense against Sybil attacks – the goal is to reduce risk rather than eliminate it completely. They may no longer be suitable for next-  
 95 generation applications because malicious entities own considerable computational capabilities.

*Trusted Certification*-based methods exploit cryptographic techniques to establish trust among all entities [12, 13]. One of most interesting works is discussed in [14]. In this scheme, a trusted authority is used to distribute group keys to all the entities in the network. Then, a short group signature (generated with the keys) is employed to preserve member privacy at lower levels, while allowing mid-level nodes to detect  
 100 Sybil attacks. However, its computational complexity is a limitation for real-time applicability. In general, the main drawback of these kinds of methods is the need for a trusted centralized authority to manage public keys, certificates, and digital signatures. Moreover, they require a widespread deployment of Road-Side Units to disseminate these certificates.

*Measurement*-based methods rely on signal strength distribution [16], position verification [17], similarity  
 105 comparison [18], or power control [19] to discover Sybils in VANETs. In this category, Received Signal Strength Indicator (RSSI) based techniques are the most adopted. An example is described in [15], which presents a robust and lightweight solution for Sybil attack problem based on RSSI readings of messages.

## 2.2. Defend against Sybils in OSNs

The existing detection schemes in OSNs can be classified in three main categories, namely *behavior-*  
 110 *based*, *graph-based*, and *DNN-based* approaches.

*Behavior*-based schemes assume that Sybil accounts show distinct behavior and interaction features compared to benign accounts. The authors of [20] introduce a weighted-strong-social (WSS) graph to model the accounts' following and forwarding behavior patterns of benign accounts and Sybils. In [21], behavior features are used to measure the connection-strength between nodes. Then, the authors combine  
 115 the betweenness-centrality graph property and these behavior features to detect Sybil nodes. Despite the effectiveness of these approaches, the monitoring of behavioral patterns of nodes is not suitable for the vehicular scenario, specifically for the volatile graph structure.

*Graph*-based approaches assume that Sybils exhibit link patterns that differ from benign ones, e.g. high separation of honest regions from Sybil regions, and limited edges between benign regions and Sybil re-

gions. Most of the existing approaches are Random Walk-based (RW) and aim to propagate trust values among nodes in the social graph. SybilRank [22] is one of the first and most representative approaches of this category due to its high accuracy and low computational complexity. Other works followed the SybilRank guidelines and achieved better detection performance, while maintaining the same general complexity. For example, SybilRadar [23] outperforms SybilRank by proposing a novel mechanism that is able to detect nodes with weak trust relationships against Sybil accounts. Random-walk based schemes usually are highly dependent upon the proper choice of *known* trusted nodes. Motivated by these issues, in [24], the authors proposed an efficient and effective Sybil community detection algorithm, called SybilExposer. This technique exasperates the structural features of the social graph and identifies Sybils in attack scenarios where they are strongly integrated with the honest communities. SybilEdge [25] determines whether a new account is Sybil by aggregating over its choices of friend request targets and these targets' respective responses. The approach is quite effective, however it cannot be applied in the VANET scenario due to time constraints of dynamic vehicular networks.

Existing literature [26, 27] proposed the use of Deep Learning to achieve higher performance in several applications. In [26], the authors exploit Deep Neural Networks (DNNs) to detect social bots, intended as Sybil entities. In particular, they design a novel DNN model to integrate the strength of feature engineering, and behavior modeling in a unified manner. This model consumes high time complexity, thus conflicting with the limitations of vehicular networks. DNNs have also been used to extract relevant features about malicious accounts. It is the case of DEC [27], a machine learning-based framework that uses DNNs to extract more than 20,000 features for each account. These features are used to train supervised machine learning models that classify accounts across many different kinds of abuses, from spam activities to Sybils generation.

### 2.3. Defend against Sybils in Crowdsourcing

A common approach to face Sybils in crowdsourcing applications is to deploy appropriate *Incentive*-based mechanisms. In [28], the authors propose Informant, a protocol based on an economic incentive policy for which a reward is calculated through an auction. However, it was demonstrated that Sybil attacks may undermine the auction-based mechanisms. Motivated by these premises, in [29], the authors designed a Sybil-proof online incentive mechanism to detect Sybil attacks in scenarios with high dimension of flexibility, such as time and mobility. Depending on users' flexibility in performing their tasks, the authors have investigated both single-minded and multi-minded cases. 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

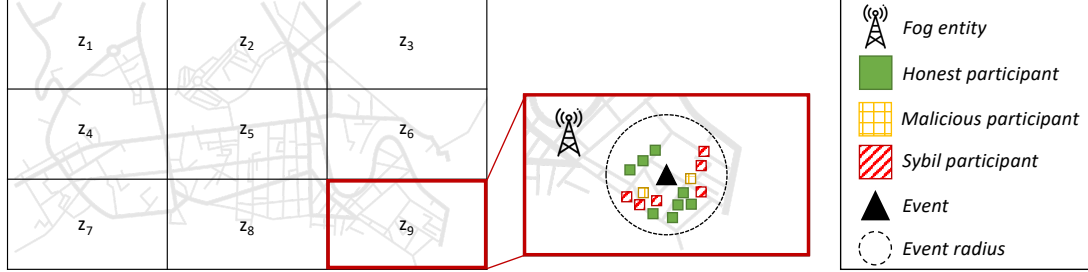


Figure 1: The proposed *system model* captures a particular urban area composed of a set of regions, each of which contains (potential) participants, sensing tasks, and fog entities.

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.

In recent years, the research community is designing advanced methods to discriminate between benign and Sybil entities. In this sense, one of the most interesting works is [30], named Social Turing Test. The authors have explored the possibility of involving humans in the Sybil detection task. Using data from two famous social platforms, the authors tested the efficacy of humans, both expert annotators and workers hired online, at detecting Sybil accounts simply from the information on their profiles. The main drawback of this approach consists is that expert workers cannot be dispensed with, as the average worker does not perform well individually. Another interesting work is discussed in [31]. The authors focused on the privacy leaks and attacks during task subscription, and propose a privacy-preserving task subscription scheme with Sybil detection, called SybSub. The main limitation of SybSub consists in assuming that a request to join a task can only be made by registered users in the system. In other words, if an entity is able to register itself to the system, then the malicious entity is misclassified as genuine.

### 3. System and Threat Models

In this section, we formally introduce the entities involved within the proposed model. Then, we proceed to the description of the threat model.

#### 3.1. System Model

The considered system model is shown in Fig. 1. We examine an urban area composed of  $Z$  sensing zones, each containing a fog entity,  $f^z$ , and a set of vehicles,  $V^z = \{v_1^z, v_2^z, \dots, v_{|V|}^z\}$ , responsible for contributing to a specific event. An event represents a physical variable of the environment the application is managing. Usually, a vehicular crowdsourcing application reports a set of distinct events but, for clarity of exposition, from here onward, we assume that a specific area is associated with only one event, indicated as



$\varepsilon^z$  (the extension to the general case is trivial). Examples of reported events could include gas availability at a gas station, a person injured in a roadside, or other information related to the trip and road conditions [32].

175 Once  $\varepsilon^z$  is generated, a sensing task is submitted, through the fog entity, to all the vehicles  $V^z$  whose goal is to simply report feedback asserting if the event is “true” or “false”. Moreover, to access the scope of the sensing process at a reasonable granularity, we distribute the duration of sensing tasks,  $\Delta$ , into fixed periodic intervals,  $\delta$ , called *response periods*, where  $\delta \in [1, \Delta]$  denotes the  $\delta$ -th response period.

Finally, we assume that the crowdsourcing platform is equipped with a Reputation Management Engine  
180 (RME) that assigns trust scores to each participant based on the past interactions. We identify three main actors in the proposed model, namely *Honest Participants*, *Malicious Participants*, and *Sybil participants*. The first category consists of entities who physically exist and want to release real feedback about a sensing task. The second category of participants also physically exists in the model, but they act as internal attackers by creating Sybil vehicles. Finally, Sybil vehicles are pseudonymous identities generated and employed by  
185 malicious participants to significantly impact the system. In general, malicious participants aim to influence the network by deceiving the belief about a sensing task. For a better readability, from now on we will use to the terms “nodes”, “participant”, and “vehicle” interchangeably.

### 3.2. Threat Model

In this section, we describe the *threat model* of the adversary by considering the three key aspects: *i)*  
190 the *attacker’s knowledge* of the scenario where he wants to perform an attack; *ii)* the *attacker’s capability* or, in other words, the actions that the attacker can enforce; *iii)* the intent of the attacker, also known as the *attacker’s goal*.

**Attacker’s knowledge:** The attacker has access to the crowdsourcing platform and knows exactly where the events are located in the urban area, and when the events occur. However, the platform does not allow the  
195 attacker to realize whether an event was generated by real users or not, i.e., the attacker does not know the truthfulness (or falsity) of the event. Further, the malicious participant knows his position and the relative positions of the neighboring vehicles in the proximity graph, due to the proximity nature of equipped sensors. Thus, the malicious participant injects Sybil attacks (i) if it is physically within the event radius and (ii) within the one-hop range.

200 **Attacker’s capability:** After the malicious participant knows who are his neighbors, it simulates Sybil participants by forging fake messages that allow mimicking the existence of Sybil entities, as if they are real vehicles. The malicious participant can create an unlimited number of Sybil vehicles, even if the generation of such pseudonymous identities requires computational resources. In this sense, the research community has already demonstrated how easy it is for an attacker to generate Sybil entities for diverse vehicular

crowdsourcing applications. For instance, launching attacks against crowdsourced maps like Waze, or Uber, requires (i) to automate input to mobile devices that run the app, (ii) control the device GPS and simulate device movements (e.g., car driving), and (iii) obtain access to multiple devices [33].

**Attacker's goal:** Malicious participants aim to worsen the services provided by the crowdsourcing application to the end-users. For example, an attacker might generate different Sybil participants to fool drivers into believing that there is no traffic in a certain urban area when actually the traffic is particularly congested. This causes a disservice for drivers using the VSN-based application.

#### 4. System Architecture

The fog-based architecture (see Fig. 2) exploits information retrieved by the vehicular network to identify Sybil participants, thus allowing for the enhancement of the service quality provided to the end-users.

First of all, a sensing task created by the Cloud is submitted to the appropriate fog entities, and then to the vehicles within the event radius. These vehicles are responsible for detecting other vehicles and sharing such data with the intermediate fog entities at the upper layer.

The fog entities aim to run the SybilDriver algorithm [1] that (i) generates the *Proximity Graph* by aggregating data obtained from the vehicular network, and then (ii) runs the *Smart Sybil Detection* which returns the set of selected (trusted) participants. Moreover, a fog entity models the truthfulness of interest event as well as the related QoI. Note that the fog entities can also exchange information with other ones at the same layer via a high-capacity mesh network. In a real-world scenario, this characteristic may allow to reduce the network overhead by permitting, for example, the mutual sharing of trust scores of the participants involved in the nearby sensing regions, without the necessity to contact the Cloud. Finally, all the produced data are sent to the Cloud, an entity responsible for resource-consuming analysis of data, such as the managing RME, or the updating machine learning models employed by the system.

To clarify the involvement of entities' communication in the proposed architecture, message flow running within a general sensing zone  $z \in Z$  is further discussed in the following section.

##### 4.1. Message Flow

Let  $v_m^z$  and  $v_n^z$  (with  $m \neq n$ ) be a pair of vehicles, and  $f^z$  a fog entity, all belonging to the sensing region  $z \in Z$ . An overview of the entire data flow through the layers of the proposed architecture, from the vehicles to the fog, and the Cloud, is depicted in Fig. 3.

After an initial phase dedicated to the discovery of other vehicles via proximity sensors, the vehicle  $v_m^z$  creates a message  $M_1$  to ask  $v_n^z$  to send the list of its neighbors. Then,  $v_n^z$  returns the required information through the message  $M_2$ . It is worthy of noting that there are as many  $M_1$  and  $M_2$  as all the pairs of

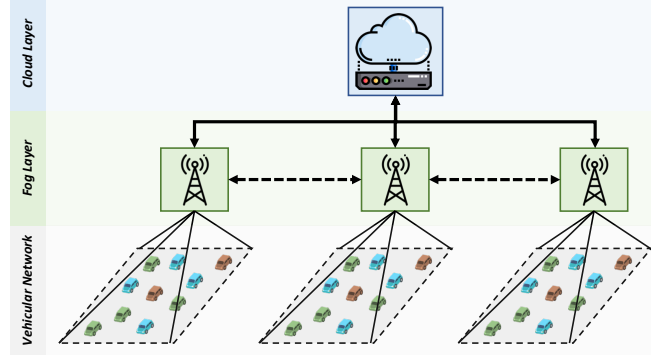


Figure 2: Proposed architecture.

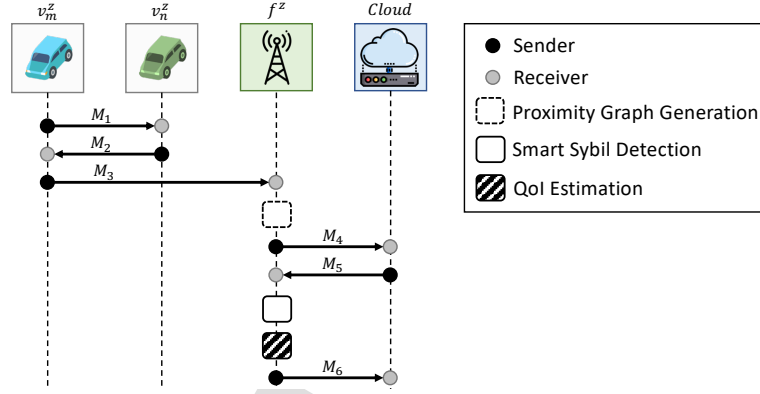


Figure 3: Information exchanged between vehicles, fog, and Cloud.

communicating vehicles in the network at a specific time  $\delta$ . For example, the vehicle  $v_n^z$  sends a message  $M_1$  to a vehicle  $v_m^z$  which, in turn returns a message  $M_2$ . The set of messages  $M_2$  received by  $v_m^z$  is processed to deliver a message  $M_3$  to the corresponding fog entity. Moreover, fog entities receive a set of messages  $M_3$  to be elaborated, one for each vehicle involved in the sensing process. Then, the fog entity runs the

240 *Proximity Graph Generation* and contacts the Cloud with  $M_4$  to retrieve the reputation scores of the analyzed vehicles. These values are returned with  $M_5$  allowing the *Smart Sybil Detection* for determining the *honest participants*, the *Sybil participants*, and the final QoI. At the end of the entire process, the Cloud analyses the information contained in  $M_6$  (e.g., honest and Sybil participants) to evaluate if it is necessary to update machine learning models or the reputation values assigned to the participants.

## 245 5. Modules of SybilDriver

The physical vehicular network represents the starting point of SybilDriver and provides the basic information for the construction of the proximity graph. Generally, this network is composed of interconnected

Table 2: List of notations

Description	Symbol
Set of sensing zones of the urban environment	$Z$
Fog entity located in the sensing zone $z \in Z$	$f^z$
Event in the sensing zone $z \in Z$	$\varepsilon^z$
Duration of the sensing task	$\Delta$
Response period $\delta \in \Delta$	$\delta$
Communication threshold	$\theta$
Euclidean distance between $v_m^z$ and $v_n^z$	$d(v_m^z, v_n^z)$
Set of neighbors of $v_m^z \in V^z$	$\zeta(v_m^z)$
Jaccard similarity between vehicles $v_m^z$ and $v_n^z$	$J_{m,n}$
Set of messages shared by vehicles	$M^z$
Set of vertices in graph $G^z$	$V^z$
Set of edges in $G^z$	$E^z$
Set of weights in $G^z$	$W^z$
Set of communities extracted from $G^z$	$\mathbb{C}^z$
$i$ -th community	$C_i^z$
Set of trusted vehicles in the $i$ -th community	$S_i^z$
Community rank	$R_i^z$
Trust threshold	$\tau_{trust}$
Pivot threshold for Community Ranking	$\tau_{rank}$
Cardinality threshold	$Thr$

vehicles that can interact with each other using short-range communication protocols. For a better readability, Table 2 reports the notation we used in the discussion of the proposed approach.

We assume that two vehicles  $v_m^z$  and  $v_n^z$  (located in  $z \in Z$ ), can communicate only if the distance between them is smaller than or equal to a given threshold  $\theta$  i.e.,  $d(v_m^z, v_n^z) \leq \theta$ . Once the communication is established,  $v_m^z$  sends  $M_1$  to request the neighboring list of  $v_n^z$ , which is provided by means of the message  $M_2$ . Then, the *Jaccard similarity* index is calculated as follows:

$$J_{m,n} = \frac{|\zeta(v_m^z) \cap \zeta(v_n^z)|}{|\zeta(v_m^z) \cup \zeta(v_n^z)|}. \quad (1)$$

where  $\zeta(\cdot)$  represents the vehicle's neighbors. The idea behind this similarity measure is that the more the neighbors between two vehicles, stronger the connection between them. This is the only one of many similarity measures generally employed in the social networking domain [34]. However, it perfectly fits our application scenario where the neighboring set of vehicles is the only detail passes to the system.

Even if the time complexity of computing Jaccard similarity is lower than other techniques (e.g., [35]),

255 assigning a weight to each communication link may require a very long time for mid-size to large-size networks. To overcome this limitation, SybilDriver adopts a *distributed approach* where each vehicle calculates its own Jaccard index using data shared by the vehicles within its proximity range. At the end, each vehicle sends a message  $M_3$  to  $f^z$  adopting a JSON-like syntax:

$$M_3 = \{k_1: ID_{v_m^z}, k_2: [\{k_3: ID_{v_1^z}, k_4: J_{m,1}\}, \dots, \{k_3: ID_{v_{|\zeta(v_m^z)|}^z}, k_4: J_{m,|\zeta(v_m^z)|}\}]\}$$

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

### 5.1. Proximity Graph Generation

To meet the latency and scalability requirements of vehicular applications, the proximity graph generation is a task performed by the fog entity  $f^z$ , which collects and creates a set  $\mathbb{M}^z$  composed of  $M_3$  message type. These messages are used to generate a proximity graph  $G^z \triangleq (V^z, E^z, W^z)$  for each response period  $\delta$ , where  $V^z$  and  $W^z$  represent sets of vehicles and weights assigned to the links, respectively. Moreover,  $E^z \subseteq V^z \times V^z$  is the communication links between pairs.

Algorithm 1 presents the pseudo-code of the proximity graph generation which takes  $\mathbb{M}^z$  as the input. For each message in  $\mathbb{M}^z$ , the algorithm extracts vehicle ID, its neighbors' list, and adds the vehicle to the vertex set  $V^z$  (lines 2-5). At this point, the algorithm dynamically updates the other two sets, i.e.,  $E^z$  and  $W^z$ , by implementing the following logic (lines 8-19). Let  $v_m^z$  and  $v_n^z$  be two vehicles in the sensing zone  $z$ . Three different cases can occur: (i) an edge between  $v_m^z$  and  $v_n^z$  already exists (lines 9-12); (ii) no edge exists between them and  $v_n^z \in \zeta(v_m^z)$  (lines 15-16); and (iii) no edge exists between  $v_m^z$  and  $v_n^z$  and  $v_n^z \notin \zeta(v_m^z)$  (lines 17-18). Specifically, the first case represents a scenario where either both  $v_m^z$  and  $v_n^z$  are honest or Sybil vehicles, i.e.,  $v_m^z \in \zeta(v_n^z)$  and  $v_n^z \in \zeta(v_m^z)$ . 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 vehicle is trying to connect itself to an honest vehicle. To manage the third case, the algorithm subtracts 1 to highlight those links between an honest vehicle and a Sybil vehicle. These three cases can be summarized in the following weight function:

$$w_{m,n}^z = \begin{cases} \frac{J_{m,n} + J_{n,m}}{2} & \text{if } e_{m,n}^z \in E^z & \text{(i)} \\ J_{m,n} & \text{if } e_{m,n}^z \notin E^z \text{ and } v_n^z \in \zeta(v_m^z) & \text{(ii)} \\ \frac{J_{m,n} - 1}{2} & \text{if } e_{m,n}^z \notin E^z \text{ and } v_n^z \notin \zeta(v_m^z) & \text{(iii).} \end{cases} \quad (2)$$

The upper and lower bounds of the weight are demonstrated in the following theorem.

**Theorem 1.** *The weight of a link between any honest vehicle  $v_m^z$  and a Sybil vehicle  $v_n^z$  is bounded by*

270  $-0.5 \leq w_{i,j}^z \leq 0.$

**Algorithm 1:** Proximity Graph Generation**Data:** The set of messages  $\mathbb{M}^z$ **Result:** Proximity Graph,  $G^z$ 1  $(V^z, E^z, W^z) \leftarrow (emptyList(), emptyList(), emptyList());$ 2 **foreach**  $msg \in \mathbb{M}^z$  **do**3      $v_m^z \leftarrow msg.get(k_1);$ 4      $neighborsList \leftarrow msg.get(k_2);$ 5      $V^z.add(v_m^z);$ 6     **foreach**  $obj \in neighborsList$  **do**7          $v_n^z \leftarrow obj.get(k_3);$ 8          $tmp \leftarrow \mathbb{M}^z.getMsg(v_n^z);$ 9         **if**  $E^z.exist(v_m^z, v_n^z)$  **then**10              $J_{m,n} \leftarrow obj.get(k_4);$ 11              $J_{n,m} \leftarrow W.getFromEdge(v_m^z, v_n^z);$ 12              $W.update((J_{n,m} + J_{m,n})/2);$ 13         **else**14              $J_{m,n} \leftarrow obj.get(k_4);$ 15             **if**  $v_m^z \in tmp.get(k_2)$  **then**16                  $W.add(J_{m,n})$ 17             **else**18                  $W.add((J_{m,n} - 1)/2);$ 19              $E.add(createEdge(v_m^z, v_n^z));$ 20  $G^z \leftarrow createGraph(V^z, E^z, W^z);$ 

*Proof.* By definition, the Jaccard similarity index of any two vehicles is bounded as  $0 \leq J_{m,n} \leq 1$ . The minimum value is obtained when there are no common neighbors, whereas the maximum one is obtained when all the neighbors are common. If  $v_m^z$  and  $v_n^z$  are honest and Sybil vehicles, respectively, then Algorithm 1 follows Case (iii) of Eq. (2). At the lower bound of the Jaccard similarity index  $J_{m,n}$ , we can estimate  $-0.5 \leq w_{m,n}^z$ . At the upper bound of  $J_{m,n}$ , the weight is bounded as  $w_{m,n}^z \leq 0$ . Thus, the weight between vehicles  $v_m^z$  and  $v_n^z$  is bounded as  $-0.5 \leq w_{m,n}^z \leq 0$ .  $\square$

The proposed approach is novel in two ways. (a) Our scheme is distributed in the sense that each vehicle estimates its own Jaccard similarity index to the neighboring vehicles, thus reducing the computational complexity; (b) Based on Theorem 1, the system can discover links between Sybil and honest vehicles by improving the performance of the community detection and ranking algorithm as discussed in Section 7.

**Lemma 1.** The computational time complexity of Algorithm 1 is  $\Theta(|V^z|^2)$ .

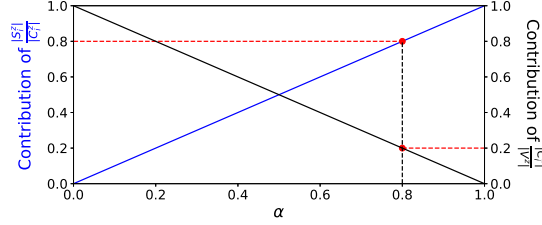


Figure 4: The existing relationship between the parameters composing the Ranking index against  $\alpha$ .

*Proof.* Algorithm 1 creates the proximity graph  $G^z$ , using as input the messages contained in the set  $\mathbb{M}^z$ . We remark that a vehicle generates a  $msg \in \mathbb{M}^z$  that includes the list of its neighbors, thus requiring a time proportional to the number of connections for the *for statement* at line (6). This means that, in the worst-case scenario, a vehicle is connected with the other  $(|V^z| - 1)$  vehicles in the network. Considering that the *for statement* at line (6) has to be repeated exactly  $|V^z|$  times (see line (2)), it can be observed that the cost function for the Algorithm 1 is  $T(V^z) = |V^z|(|V^z| - 1) = |V^z|^2 - |V^z|$ , thus resulting in  $\Theta(|V^z|^2)$ .  $\square$

## 5.2. Smart Sybil Detection

The proximity graph created by the previous module is the baseline for the Smart Sybil detection phase, which targets to accomplish two main tasks: (i) *community detection and ranking*, necessary to perform a pre-filtering of the participants; (ii) *effective recognition* through a machine learning technique. Both steps require parameters that, at this point, the fog does not yet possess, i.e., the reputation scores assigned to participants based on their previous interactions. The value calculated by the Reputation Management Engine (RME) assigns a score in a range between 0 and 1, where 0 means the participant has no trust, whereas 1 indicates a trusted user. Participants are initialized with a 0.5 reputation score. The score is then updated over time (increased or decreased) based on the analysis performed by the RME for biased free analysis. A vehicle is considered *trusted* by the system if its reputation score is higher than the threshold  $\tau_{trust}$ .

Once  $f^z$  owns all the required information, it runs *Community Detection and Ranking* and *Effective Recognition* tasks as described below.

### 5.2.1. Community Detection and Ranking

SybilDriver uses Louvain method to extract the community set  $\mathbb{C}^z = \{C_1^z, C_2^z, \dots, C_{|\mathbb{C}|}^z\}$  from the proximity graph  $G^z$ . Then, SybilDriver filters the communities depending on (i) their size, and (ii) a *community rank* modeled from the vehicles within that community. Let  $S_i^z \subseteq C_i^z$  be the set of vehicles having a reputation

score higher than  $\tau_{trust}$ , then rank  $R_i^z$  for each community can be modeled as follows:

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

where  $0 \leq \alpha \leq 1$ ,  $|C_i^z|$  is the cardinality of the community  $C_i^z$ , and  $|V^z|$  is the number of vehicles in the proximity graph  $G^z$ . In order to better understand the physical meaning of Eq. (3), we can refer to Fig. 4, in which the contribution of  $\frac{|S_i^z|}{|C_i^z|}$  and  $\frac{|C_i^z|}{|V^z|}$  members against  $\alpha$  parameter is shown. In particular, it is shown that  
 305 when  $\alpha = 0.8$ ,  $R_i^z$  is more dependent on the first term of Eq. (3), and less on the second term. In general,  $\alpha$  can be tuned according to the application-specific goals. In our simulation, the value of  $\alpha$  has been set to prefer bigger communities with a higher number of trusted participants.

Algorithm 2 describes pseudo-codes of *Community Detection and Ranking* algorithm. The inputs are the proximity graph  $G^z$ , the thresholds  $Thr$ ,  $\tau_{trust}$ , and  $\tau_{rank}$ . Precisely,  $Thr$  is used to verify whether the  
 310 cardinality of  $C^z$  is in the order of the number of network nodes, whereas  $\tau_{rank}$  is used for the pivot detection. Here, the pivot allows to reduce the search spaces by preferring communities with higher ranks.

Lines 1-3 initialize different variables. Once the communities are extracted by Louvain (line 4), the algorithm checks if the cardinality of  $C^z$  is in the order of the number of nodes that constitute the graph  $G^z$  (lines 5-8). In particular, if the condition is satisfied (line 5), the entire graph is considered as a single large  
 315 community (line 6); otherwise the algorithm stores all communities discovered by the Louvain method as per line 8 of the algorithm. Next, the algorithm iterates over each community (line 9), and assigns a rank by calling the function *setRank()* returning  $R_i^z$ , defined in Eq. (3) (line 10). At the end of this procedure, the ranked 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 incremented by 1 (line 16) and the for loop  
 320 is interrupted (line 18). Finally, the algorithm extracts from the ranked list a sub list of communities using the *subList()* function where the pivot is passed as the starting index, and  $|Ranks|$  (i.e., the cardinality of the ranked list) as the ending one (line 20).

**Lemma 2.** *The computational time complexity of Algorithm 2 is  $O(|V^z| \log |V^z|)$ .*

*Proof.* Algorithm 2 strongly depends on four main operations, while the remaining ones can be assumed to  
 325 have unitary time cost. The first time-consuming task is the *Louvain-based community detection* (line 4), which is characterized by  $O(|V^z| \log |V^z|)$ , as discussed in [36]. The second most expensive operation is the *for statement* (line 9) that requires a time proportional to  $|V^z|$  – all the vehicles, for each community, have to be parsed for the calculation of the rank. This process should be repeated  $|C^z|$  times, means all vehicles for all the communities must be analyzed – in the worst-case scenario complexity should be  $O(|V^z|)$ . Next,



**Algorithm 2:** Community Detection and Ranking

---

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

```

1 Ranks  $\leftarrow$  emptyList();
2 Communities  $\leftarrow$  emptyList();
3  $V \leftarrow G^z.getVertexes();$ 
4  $\mathbb{C}^z \leftarrow G^z.communityDetection();$ 
5 if  $|G^z| - |\mathbb{C}^z| \leq Thr$  then
6   | Communities  $\leftarrow G^z$ ;
7 else
8   | Communities  $\leftarrow \mathbb{C}^z.asList();$ 
9 foreach  $C_i^z \in$  Communities do
10  | Ranks.add( $i, setRank(|V^z|, C_i^z, \tau_{trust})$ );
11 Ranks.sort('Ascending');
12 pivot  $\leftarrow 0$ ;
13 foreach obj  $\in$  Ranks do
14   | if obj.getNext()  $\neq NULL$  then
15     | if obj.getNext().getR()  $- obj.getR() < \tau_{rank}$  then
16       | | pivot  $\leftarrow$  pivot + 1;
17     | else
18       | | break;
19 if pivot  $< |Ranks|$  then
20   | Ranks  $\leftarrow$  Ranks.subList(pivot, |Ranks|)
```

---

330 the *sorting operation* (line 11) may negatively impact the system because, in general, the time complexity of a sorting algorithm depends on the number of elements. Considering that in the worst-case scenario we have one community for each vehicle, the algorithm we are adopting (i.e. Heap sort) is characterized by  $O(|V^z| \log |V^z|)$ . Finally, the fourth most expensive task is the *for statement* (line 13) that depends on the number of communities identified previously. In the worst-case scenario we have one community for each vehicle, thus resulting in  $O(|V^z|)$ . Given the above cases, the overall time complexity of the proposed

335 Algorithm 2 can be written as  $2O(|V^z| \log |V^z|) + 2O(|V^z|)$  that finally results in  $O(|V^z| \log |V^z|)$  computational time complexity.  $\square$

### 5.3. Effective Recognition

In the proposed system, machine learning is employed to perform a more in-depth analysis for identifying Sybil participants. This further step is required because, after the filtering performed by Algorithm 2, SybilDriver cannot rely only on the reputation score of each vehicle. If the reputation score is the only index used to determine the possibility of a vehicle to provide feedback for a sensing task, it would be unfair to newly joined vehicles.

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 to classify unlabeled data. Thus, the feature vector of each vehicle is built by considering a tuple of four parameters –  $N_r$ ,  $W_r$ ,  $T$ , and  $N_t$  as discussed in the following.

Given a node  $v_m^z \in C_i^z$ , the trusted neighbors' ratio  $N_r$  is defined as:

$$N_r = \frac{|\zeta(v_m^z) \cap S_i^z|}{|\zeta(v_m^z)|}, \quad (4)$$

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

$$W_r = \frac{\sum_{j \in \zeta(v_m^z)} w_{m,j}^z}{|\zeta(v_m^z)|}. \quad (5)$$

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 neighboring nodes. 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 2.** *The computational complexity of the proposed scheme is  $O(|V^z|^2)$ .*

*Proof.* The computational time complexity of the proposed scheme depends on three-time complexities: (i) proximity graph construction, (ii) community detection and raking, and (iii) feature extraction and effective recognition. Lemmas 1 and 2 state the time complexity of cases (i) and (ii) are  $O(|V^z|^2)$  and  $O(|V^z| \log |V^z|)$ , respectively. In case (iii), the worst-case for feature extraction can be written as  $O(|V^z|^2)$ , because it is necessary to visit all the neighbors for each vehicle in the network. Furthermore, 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.,  $O(Dk)$  [37]. However, we can see  $O(Dk) \ll O(|V^z|^2)$ . Thus, we conclude the overall computational complexity is led by the linear combination of all the three cases, i.e.,  $O(|V^z|^2)$ .  $\square$

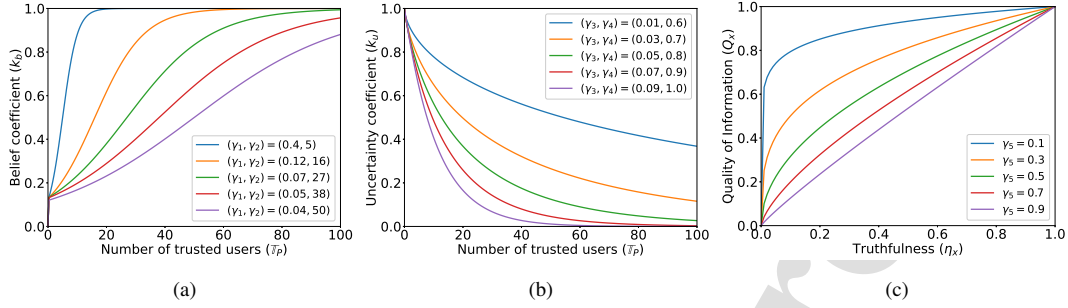


Figure 5: (a) Effect of trusted users over belief weight. (b) Impact of trusted users over uncertainty weight. (c) Effect of truthfulness over QoI index.

#### 5.4. Truthfulness and QoI models

The last task of the proposed system is to compute the truthfulness of the  $\varepsilon^z$ , the related QoI provided to the end-users, during the time span  $\Delta$ . These metrics are strictly dependent on three parameters returned by the *Smart Sybil Detection* algorithm, i.e., the set of selected (trusted) participants  $\mathbb{T}_P$ , the set of discarded participants  $\mathbb{D}_P$ , and  $\beta$  that is a factor indicating the effectiveness of the detection performance [38]. Intuitively, the better the recognition rate ( $\beta \approx 1$ ), the higher the truthfulness and QoI values will be – the set of participants is homogeneous and composed of only honest participants. Conversely, the lower the recognition performance ( $\beta \approx 0$ ), the worse the two metrics will be – the set of participants is heterogeneous and composed of honest, sybil and malicious entities. In addition to this aspect, the truthfulness and the QoI are also related to both a degree of uncertainty and a degree of belief [39, 40], indicated as  $k_u$  and  $k_b$  respectively. The truthfulness of a specific event is then described by the combination of two coefficients  $k_b$  and  $k_u$ , that refer to the event  $\varepsilon^z$  in the considered crowdsourcing application:

$$\eta_{\varepsilon^z} = \beta k_b + (1 - \beta) k_u, \quad (6)$$

where  $0 < \{\beta, k_b, k_u\} < 1$ . We have modeled the coefficients  $k_b$  and  $k_u$  using a parametric version of the Sigmoid function [41] and the exponential relaxation function [42], respectively. Thus, the coefficient  $k_b$  can be written as:

$$k_b = \begin{cases} 0, & \text{if } |\mathbb{T}_P| = 0, \\ \frac{1}{1 + e^{-\gamma_1(|\mathbb{T}_P| - \gamma_2)}}, & \text{otherwise.} \end{cases} \quad (7)$$

For low values of  $|\mathbb{T}_P|$ , the coefficient  $k_b$  decreases, which in turn, contributes to a smaller expected truthfulness. In contrast,  $k_b$  gradually increases with the increase of  $|\mathbb{T}_P|$ . The Sigmoid function is depicted in Fig. 5a when varying  $\gamma_1$  and  $\gamma_2$ . These values are the stretching exponents that determine how quickly  $k_b$

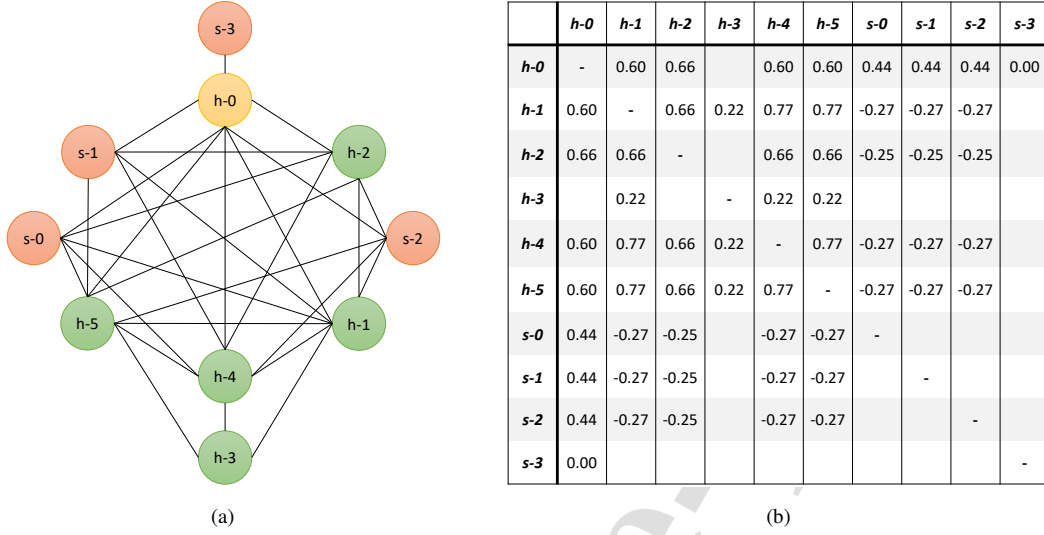


Figure 6: Illustrative example: (a) Network topology. (b) Matrix of weights generated by Algorithm 1 for the network topology.

settles to the maximum value, i.e., 1. As regards  $k_u$ , it is reasonable to assume that the degree of uncertainty is higher when there are only a few participants and lower in other cases. In other words, for smaller values of  $|\mathbb{T}_P|$ , SybilDriver should exhibit a high degree of uncertainty that decreases with a higher value of  $|\mathbb{T}_P|$ . Such behavior can be modeled using the exponential function as follows:

$$k_u = e^{-(\gamma_3 |\mathbb{T}_P|)^{\gamma_4}} \quad (8)$$

where  $0 < \{\gamma_3, \gamma_4\} < 1$  are the stretching exponents that determine how quickly  $k_u$  settles to the minimum value. Fig. 5b shows the trend of the curve when varying  $\gamma_3$  and  $\gamma_4$  values – with an increase in the number of trusted participants, the degree of uncertainty decreases. Based on the obtained truthfulness value (Eq. (6)), the proposed model estimates the QoI index,  $Q_{\varepsilon^z}$ , for an event  $\varepsilon^z$  as follows [39]:

$$Q_{\varepsilon^z} = (\eta_{\varepsilon^z})^{\gamma_5} \quad (9)$$

where  $0 < \gamma_5 < 1$  denotes the discontinuing factor. Fig. 5c shows the growth of the QoI index for different values of  $\gamma_5$  parameter. Evidently, with an increase of  $\gamma_5$ , the function shapes a linear pattern. Fine-tuning this parameter enables the vehicular crowdsourcing platform to achieve different levels of QoI and adapt contextual requirements of the application.

## 6. An Illustrative Example

This section illustrates an example with reference to the topology given in Fig. 6a. Honest nodes are colored in green (from  $h-1$  to  $h-5$ ), Sybils in red (from  $s-0$  to  $s-3$ ), and selfish in yellow (labeled  $h-0$ ).

1	===== Community Detection =====	15	Is $R_1 - R_2 \geq \tau_{rank}$ ?
2	[0] h-1, h-2, h-3, h-4, h-5	16	Yes, drop [2] and keep [1,0]
3	[1] h-0, s-0, s-1, s-2	17	===== Sybil Detection =====
4	[2] s-3	18	Performing Random Forest for communities [1, 0]
5	===== Community Ranking =====	19	[1] {h-0: 1, s-0: 1, s-1: 1, s-2: 1}
6	Calculating ranking index for each community	20	[0] {h-1: 0, h-2: 0, h-3: 0, h-4: 0, h-5: 0}
7	[0] $ V =10,  C_0 =5,  S_0 =4 \quad R_0=0.62$	21	Discard the untrusted participants
8	[1] $ V =10,  C_1 =4,  S_1 =0 \quad R_1=0.24$	22	{h-0, s-0, s-1, s-2, s-3}
9	[2] $ V =10,  C_2 =1,  S_2 =0 \quad R_2=0.06$	23	Return the trusted participants
10	Sorting the communities in ascending order	24	{h-1, h-2, h-3, h-4, h-5, s-2}
11	[2] 0.06	25	===== Truthfulness and QoI =====
12	[1] 0.24	26	Detection Performance
13	[0] 0.62	27	Truthfulness = 0.279
14	Calculating the pivot:	28	Quality of Information = 0.317

Figure 7: A step-by-step execution of the proposed detection technique for the topology of Fig. 6a.

**Proximity Graph Generation:** In a distributed manner, each node in the vehicular network calculates its own *Jaccard similarity* index (Eq. (1)) with respect to the neighbors. This value is sent to the system that runs Algorithm 1 and outputs the matrix of weights in Fig. 6b. All the edges between Sybil and honest vehicles have negative weights, thus verifying empirically Lemma 1. In this step, the algorithm needed  $t_1 = 0.043 \times 10^{-2}$  seconds time for the topology shown in Fig. 6a.

**Community Detection and Ranking:** Communities are detected and ranked after generation of the proximity graph. Lines 2-4 in Fig. 7 show the Louvain technique outputs three communities, labeled as [0], [1], and [2]. The Sybil vehicle *s-3*, having no neighbors other than vehicle *h-0*, was placed in an isolated community, while all the remaining vehicles formed two more significant communities: community [0] includes only honest vehicles, while community [1] contains Sybil vehicles together to the selfish node. Then, salient parameters are extracted from each community (lines 7-9) and the *rank* index is calculated according to Eq. (3). By setting  $\tau_{rank} = 0.16$  (refer to [1]), the isolated community is dropped and the remaining ones are kept for the subsequent phases (lines 15-16). At this step, the time required to complete the task is  $t_2 = 0.12 \times 10^{-2}$  seconds.

**Effective Recognition:** The filtering conducted in the previous phase is not sufficient enough to reduce the number of Sybils in the network. This highlights the importance of the classification procedure, in our case performed by Random Forest (refer to [1]). After extracting the features and classifying vehicles, two separate sets of participants are returned, i.e., *trusted* and *untrusted*. At this step, the time required to complete the classification is  $t_3 = 8.69 \times 10^{-2}$ .

**Truthfulness and QoI:** The *trusted* and *untrusted* sets are finally used to evaluate the truthfulness and QoI of the analyzed event (lines 26-28), where the outcome is obtained considering 0.07, 27, 0.05, 0.8, and 0.5 as  $\gamma_1, \gamma_2, \gamma_3, \gamma_4$  and  $\gamma_5$ , respectively (see Fig. 5). The time to calculate these values is negligible.

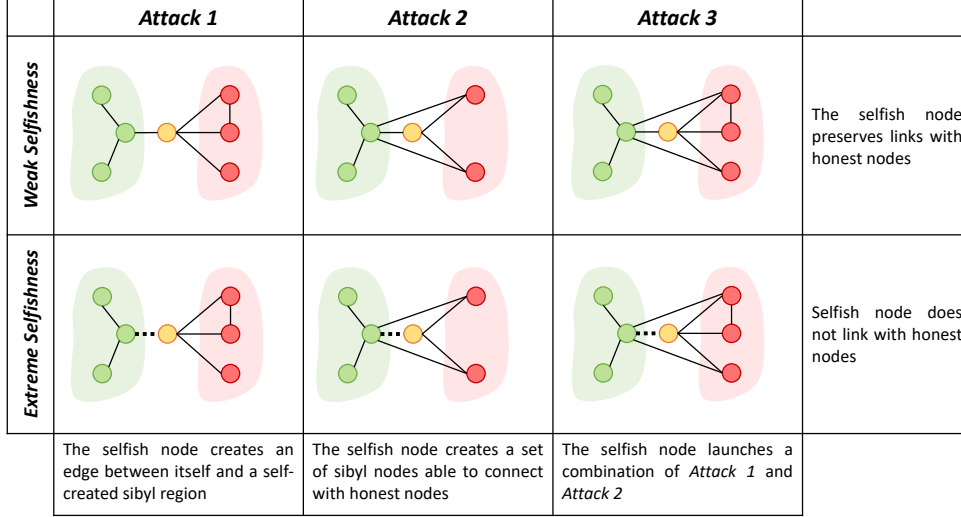


Figure 8: Overview of attacks used for the experimental evaluation. The green area represents the Honest region, while the red one is the Sybil region. The difference between *Weak Selfishness* and *Extreme Selfishness* is the way the selfish node connects with the Honest region, here indicated with the dotted line.

From the above discussion, we conclude the total running time for the network of 10 vehicles as  $t_{TOT} = (0.043 + 0.12 + 8.69) \times 10^{-2} = 8.84 \times 10^{-2}$  seconds, a reasonable time for a real-time scenario.

## 7. Performance Evaluation

In order to evaluate the performance of the proposed system, an extensive evaluation was conducted. Different from our previous work [1], we assess the system performance on two popular datasets with a larger set of attack strategies. First, we discuss the setup that has been adopted in the experimental assessment. Then, the overall system performances are tested on very popular datasets and compared with state-of-art techniques in terms of running time, accuracy, F-Score, and QoI metrics. Furthermore, we provide a more in-depth study about the effectiveness of the recognition procedure against the different attack strategies, while highlighting the drawbacks of the proposed approach. Finally, the complexity of the messages exchanged by all the entities is analyzed with the aim to prove the low impact of the fog architecture.

### 7.1. Experimental Setup

The experiments discussed in the following sections are performed on two datasets in order to assess the generality of SybilDriver. The Cologne dataset [43] was already adopted in our previous work [1] and contains the mobility patterns of more than 700,000 individual car trips over a period of 24 hours. The Seoul dataset [44] was introduced in this work and contains the vehicular mobility traces of more than 2,000 vehicles over an area of  $2.5km \times 1.5km$ . Since these datasets do not contain any information about

events, they were randomly generated with a life-span of 30 min, and included in the map. Therefore, all vehicles that fall in the area of interest of the event, whose radius has been set to 60m, are considered for the final detection [1]. To make the analysis as detailed as possible, we present the results when the density of vehicles in the zone  $z$  is *Low*, *Medium*, and *High*, i.e., when the number of vehicles is in the range (0, 10], (10, 20], and (20, 30], respectively.

Fig. 8 provides a brief description of attacks we have considered in the performance evaluation. As it can be noted, the attack strategies have been divided according to the type of attack launched and the strategy adopted. Looking at the columns, we analyzed three types of attacks typical of mobile networks, namely *Attack 1*, *Attack 2*, and *Attack 3*, where attackers exploit the fact that nodes cannot keep connections with others for a long time [45]. In *Attack 1*, the selfish node creates a Sybil region totally separate from the honest one and in which Sybil nodes connect with other Sybil nodes. In *Attack 2*, the selfish node is able to create a Sybil region in which exists connections not only among Sybil identities but also with the honest nodes. Here, it is important to remark that Sybil nodes can create false connections only with those nodes that really belong to the neighborhood of the selfish node, i.e., within its one-hop range. In other words, the *Attack 2* allows to mimic the normal user's social structures from the perspective of the social graph. Moreover, *Attack 3* is constructed by the selfish node through a combination of the other two attacks. This is the most sophisticated of the three in that the social structure of the vehicular network is always changing, as are the connections between vehicles, thus complicating detection by the system.

In the addressed social scenario, the way the selfish node interacts with the honest region is of crucial importance to achieve its personal goals. In fact, the concept of selfishness denotes that selfish nodes aim only at getting personal revenue by manipulating the network. Generally, mobile nodes have two patterns to exhibit selfishness [46]. The first one is that the malicious node decides to increase its influence in the network by creating ad-hoc Sybil regions while preserving links with honest nodes. The second is that malicious nodes do not contribute information. In the case of this work, this attack translates into lying about the presence of honest vehicles and thus providing corrupted information about the network structure. Therefore, we distinguished the attacks into two categories *Weak Selfishness* and *Extreme Selfishness*.

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 [47]. 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. Finally, before moving on to the next sections, we remark that the choice of  $\tau_{rank} = 0.16$  and

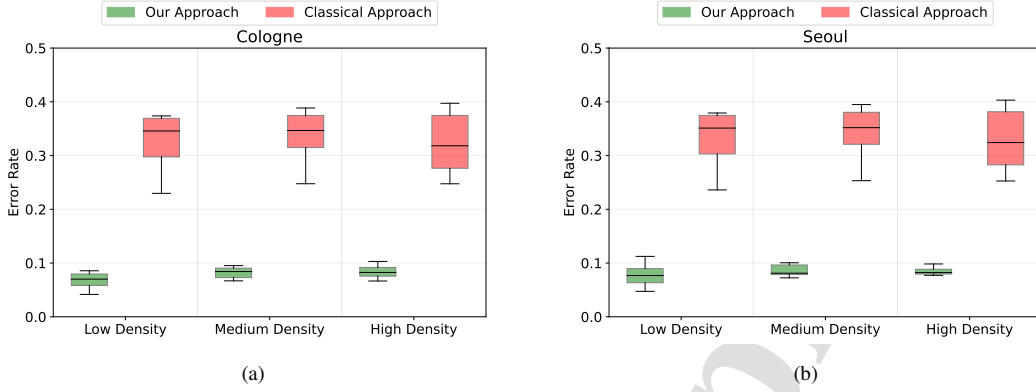


Figure 9: Error Rate comparison between Algorithm 1 and the *classical* Louvain method: (a) Cologne Dataset; (b) Seoul Dataset.

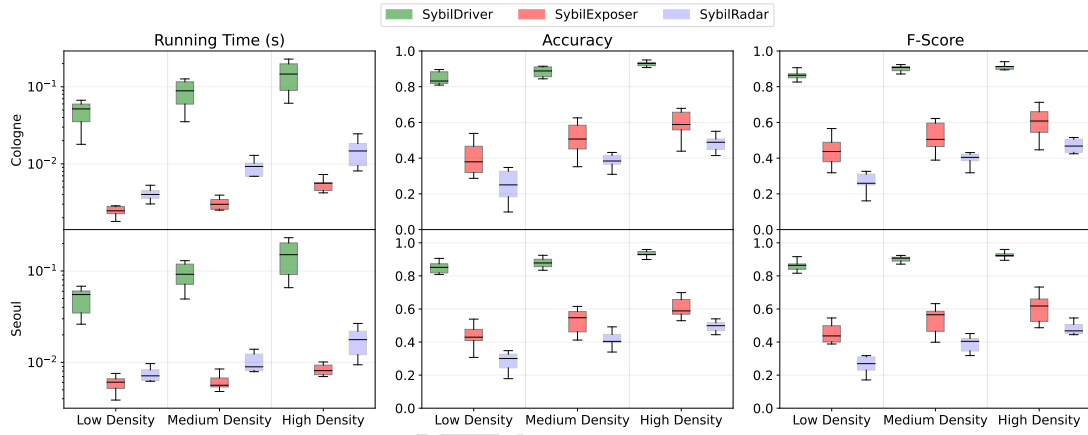


Figure 10: Running time, Accuracy and F-Score comparisons for SybilDriver, SybilExposer, and SybilRadar. The first row shows results obtained from Cologne Dataset, while the second one summarizes the results achieved when Seoul Dataset is used.

the Random Forest classifier is extensively discussed in [1], where a series of preliminary experiments were conducted with the aim to find the best configuration for the recognition.

## 7.2. Sybil Detection Performance

A first discussion regarding the performance achieved by the *Community Detection and Ranking algorithm* is provided in Sec. 5. The experimental evaluation on the new set of attacks confirmed one of the strengths of the proposed technique, i.e., the way Louvain succeeds in creating homogeneous communities thanks to the novel way the weights are assigned to the links [1].

Let the *Error Rate* be defined as the ratio of the majority class within the cluster to the number of nodes within the cluster, then Fig. 9 shows that our approach outperforms the “classical” one in every case. The



450 *Error Rate* is not affected by the number of vehicles in  $z$  because it stays very low even when the density increases. Minor differences can be noted between the outcomes achieved with respect to the Cologne and Seoul datasets – our technique creates clusters composed of similar nodes (e.g., honest or Sybil) in the considered scenarios. This finding is of extreme importance because if the communities are homogeneous, then the probability of filtering out malicious communities and maintaining reliable ones increases, thus  
 455 improving the analysis performed by the RF.

The remaining experiments focused on the performance of the detection algorithm against other techniques in the literature, namely SybilRadar [23] and SybilExposer [24]. Similarly to these works, SybilDriver exploits social graph characteristics to discover the Sybil nodes within the network, allowing us to perform a direct comparison with them.

460 Fig. 10 shows the outcomes achieved in the two analyzed datasets. In general terms, SybilDriver achieves better performance at the cost of an increased running time. This additional time is due to the higher number of steps SybilDriver executes compared to SybilExposer and SybilRadar: the first one performs only the community detection task, while the second one executes the edge weighing step and a RW-based trust propagation. Nevertheless, even if the higher number of steps results in a higher running  
 465 time, the proposed technique requires an execution time of less than seconds, an acceptable value for crowd-sensing applications. Regarding the detection of the Sybils, it is possible to note that the performance of the techniques are very similar as the datasets and vehicle density vary. According to our analysis, the main reason behind the lower accuracy and F-Score achieved by the other techniques is due to the number of nodes in the vehicular network. To better understand this issue, let's take a closer look at SybilRadar. This  
 470 technique, as well as many other in the literature (refer to Section 2), is RW-based and aims to propagate trust values among nodes to identify Sybils. Given a graph and a trusted node, the RW selects a neighbor at random, moves to this neighbor, and propagates the trust value. In a vehicular context, where the number of interconnected vehicles is limited, the probability that the trust propagation chooses a malicious node multiple times to propagate the trusted value is very high; then, the misclassification error would be very high,  
 475 as confirmed by our experimental analysis. Differently to SybilRadar, SybilExposer uses the node diversity and community diversity properties to identify the Sybil nodes as part of a Sybil community. However, interconnections between nodes within a vehicular network are made on the fly regardless of any known relationship. Then, if a malicious node is able to create a sufficiently large community of Sybil nodes, then the system will not be able to distinguish between honest and Sybil communities, thus leading to incorrect clas-  
 480 sification. This constraint affects also our technique, which however benefits from the *Effective Recognition* to improve overall performance. More details about this aspect are provided in the next section.

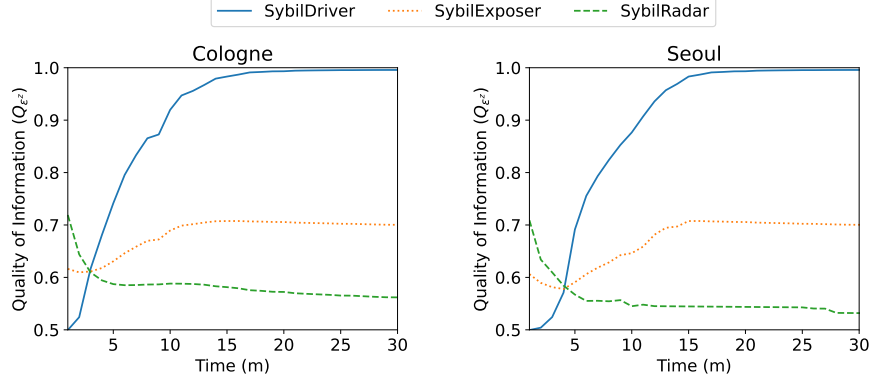


Figure 11: Evolution of QoI ( $Q_{\varepsilon}$ ) over response periods  $\delta$  of one minute, where  $\delta \in [1, 30]$ . On the left, results obtained from Cologne dataset, whilst on the right the outcomes from Seoul dataset.

The very last set of experiments, in this section, is devoted to measuring the QoI for both datasets. To conduct this type of analysis, we distributed a sensing task of duration  $\Delta = 30$  min, dividing it into sensing periods,  $\delta$ , of one minute each, i.e.,  $\delta \in [1, 30]$ . For the calculation of the metrics, we considered  $\beta$  parameter to be exactly the accuracy of the three techniques discussed previously, this is because the ground truth for the dataset is available [1]. In other words, we adopted:

$$\beta = \frac{TP + TN}{TP + TN + FP + FN}, \text{ and } (1 - \beta) = \frac{FP + FN}{TP + TN + FP + FN}, \quad (10)$$

where  $TP$ ,  $TN$ ,  $FP$ , and  $FN$  represent the number of **True Positives**, **True Negatives**, **False Positives**, **False Negatives**, respectively. We used the same values for  $\gamma_1$ ,  $\gamma_2$ ,  $\gamma_3$ ,  $\gamma_4$ , and  $\gamma_5$  as in the illustrative example.

Fig. 11 shows the QoI values obtained from both datasets. It can be seen that SybilDriver outperforms the other two techniques as time increases. This trend is due to the high discriminatory ability of our technique compared to the others, which has a greater influence of the quantity  $(1 - \beta)$  being characterized by a higher number of FPs and FNs. In the considered scenario, the system requires only 15 min to reach convergence, i.e., a sufficient number of participants have joined the sensing process. At this  $\delta$ , SybilDriver reaches optimal values (above 0.9) against the lower 0.7 and 0.6 obtained by SybilExposer and SybilRadar, respectively. The outcomes also demonstrate that the convergence depends on the frequency of the sensing periods – the more the  $\Delta$  is divided into intervals close to each other, the faster the system converges, and vice versa. In general, such a characteristic makes the system to be adapted in any referenced application scenario.

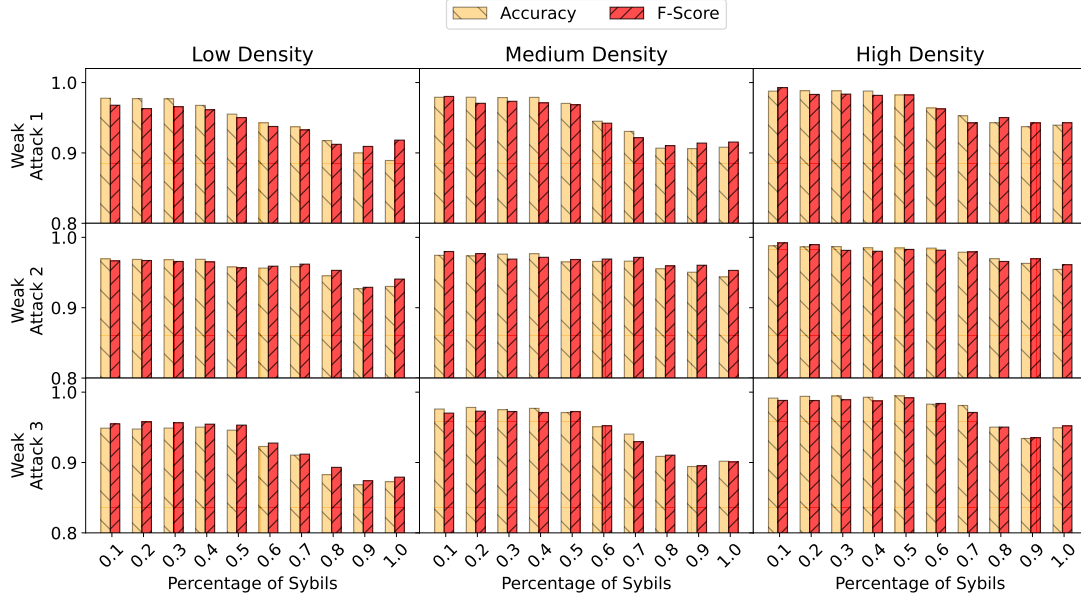


Figure 12: Accuracy and F-Score of the proposed Sybil detection technique against *Weak Selfishness* attacks while varying the percentage of Sybils in each scenario.

### 7.3. A More in-depth Performance Evaluation

The aim of this section is to test the effectiveness of our Sybil detection technique against each single attack strategy, so that weaknesses and strengths can be highlighted. To perform this task, we conducted a series of experiments by varying the percentage of Sybils with respect to the number of honest nodes—for example, if there are 5 honest vehicles in the network and the percentage of Sybils is 1.0 (i.e., 100%), then selfish node generates a Sybil region of 5 vehicles, thus equaling the number of the honest ones. Since SybilDriver achieves similar performance on the examined datasets, all experiments described in this section refer to the Cologne dataset only.

The outcomes of the *Weak Selfishness*-attacks can be observed in Fig. 12 and, as could be expected, the worst case occurs in the *Low Density* scenario. Looking at the most sophisticated attack, i.e., Attack 3, accuracy exhibits a decreasing trend as the percentage of Sybil increases, starting from an average value of about 94% and arriving at 86%. Similar arguments can be made about F-Score. The obtained result is highly dependent on the Attack 1 pattern, since, in this scenario, the system does not have enough information to discern who is actually stating the truth about the network structure. In particular, if the selfish creates (ad-hoc) a Sybil region that is totally separate from the honest ones and that equals their cardinality, then the effectiveness of the technique will be negatively affected. All performance increases if *Medium* and

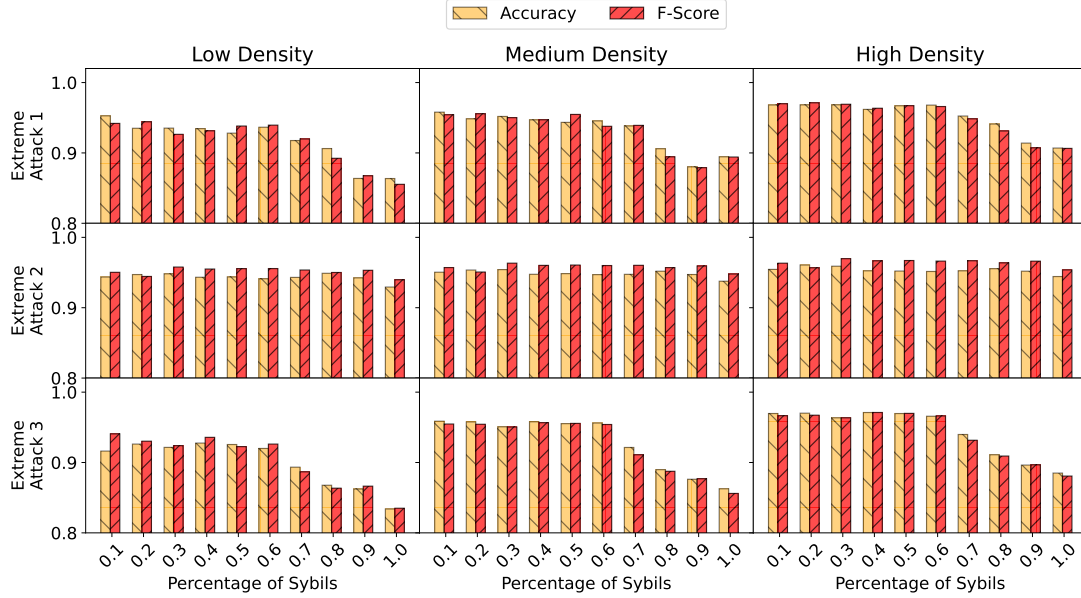


Figure 13: Accuracy and F-Score of the proposed Sybil detection technique against *Extreme Selfishness* attacks while varying the percentage of Sybils in each scenario.

510 *High* densities scenarios are considered. In general, as the number of vehicles in the network increases, the probability of finding honest nodes with a high reputation also increases, thus allowing the system to overcome the uncertainty that characterizes the *Low Density* scenario.

Similar experiments were conducted for *Extreme Selfishness*-attacks and obtained results are depicted in Fig. 13. Again, a decreasing trend of both accuracy and F-Score can be appreciated. However, the outcomes are characterized by slightly lower values than those of weak selfishness attacks in Fig. 12. Decrease in values is specifically due to the dynamic nature of the selfish which may decide whether to declare the presence of nearby honest vehicles, or not. An unexpected trend is the one shown by the Extreme Attack 2 whose performance are comparable with the weak one. This aspect clearly reveals that attacks in which the selfish tries to blend into the crowd are ineffective against the proposed approach.

520 The common factor in the attack strategies discussed so far is that vehicular networks always contain, at least, one honest vehicle. This characteristic is satisfied in most urban contexts, such as highways, cities, and is also a fundamental driving for the proposed technique to identify Sybil vehicles (see Figs. 12 and 13). However, it should not be ignored that there are other contexts, (much less frequent [48]), in which it is not always guaranteed that above condition holds. For this reason, we perform a very last set of experiments aimed at verifying the performance of the recognition technique in a scenario where no honest vehicles are

525

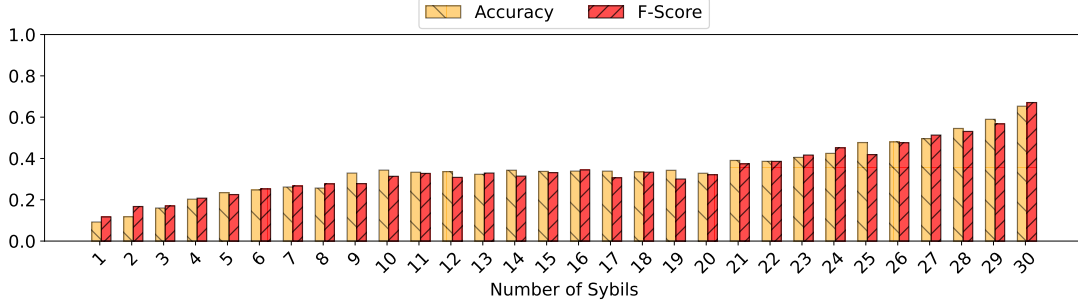


Figure 14: Accuracy and F-Score in the (extreme) scenario where no honest participants are present.

present. In this case, the attack strategy of the selfish consists of creating as many Sybil vehicles as possible in order to influence the belief of the system about a certain event in the shortest possible time. The results of this study are summarized in Fig. 14, where the average values of accuracy and F-Score are shown by varying the number of Sybil vehicles generated by the selfish nodes. As we expected, the system exhibits weak performance in recognizing Sybils because it does not have enough information about the structure of the network. It can be observed that as the number of Sybils increases, the performance of the system gradually improves until acceptable values are obtained for a number of Sybils greater than 25.

These experiments clearly show that in specific application scenarios, where the presence of the honest vehicles is not assured, the detection algorithm is less effective. However, they also highlight that a selfish cannot create as many Sybils as it wants, since the more the Sybils are in the network, the more is the effectiveness of the proposed technique. This scenario definitely represents a constraint for the Sybil detection algorithm, but not for the whole system. Indeed, this borderline case could be managed through the truthfulness and QoI models and, in particular, by setting their parameters in an appropriate way.

#### 7.4. Data Transmission

Different entities involved in the proposed system may transmit messages (containing information related to neighbors, trust scores, etc.) in different sizes. Therefore, in this section, we discuss how data processing and transmission impact the performance of the entire infrastructure. In the proposed approach, data are JSON-formatted to reduce memory occupancy and speed up the transmission as compared to XML [49]. Moreover, as discussed in Section 4.1, and shown in Fig. 3, six different types of messages are exchanged between participants, fog entities, and Cloud.

Fig. 15 depicts a comparison between messages  $M_1$  and  $M_2$  (see Fig. 3) in terms of preparation time, size, and the transmission delay necessary for sending them. In Fig. 15 (a) we can notice a constant value for the preparation time of  $M_1$  which does not depend on the number of neighbors in the graph. On the other hand,

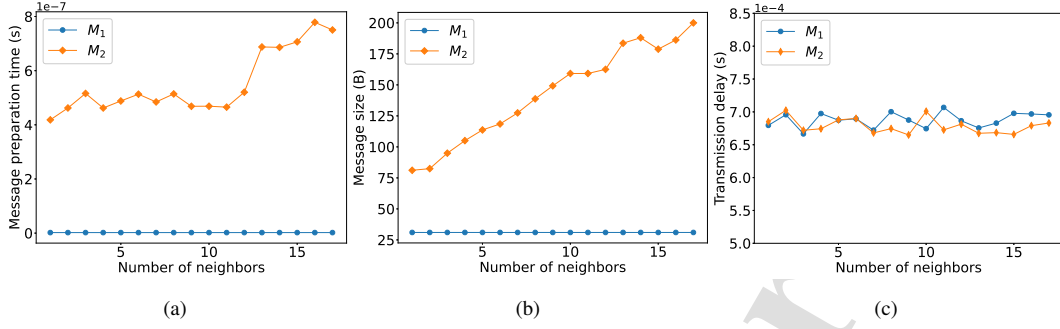


Figure 15: Analysis conducted on messages  $M_1$  and  $M_2$  in terms of (a) preparation time in seconds, (b) size in bytes, (c) transmission time in seconds.

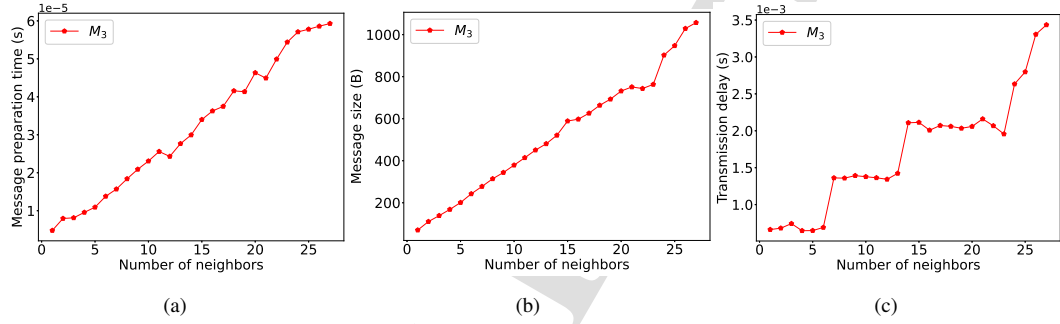


Figure 16: Analysis of message  $M_3$  in terms of: (a) preparation time in seconds, (b) size in bytes, (c) transmission time in seconds.

we observe a totally opposite behavior for the message  $M_2$  whose preparation time increases proportionally with the number of neighbors due to the structure of the message. For the same reasons, these trends are also obtained for the message size (shown in Fig. 15 (b)); precisely, the message size does not change because the only entities that interact are the *sender* vehicle and the *receiver* one for the message  $M_1$ . The case of the message  $M_2$  is different because its internal structure strongly depends on the graph topology which, in turn, causes the message size to increase in accordance with the number of sender neighbors. With respect to Fig. 15 (c), the transmission delays are in the same range for both messages. Such a result derives from the experimental setup we chose to measure the performance of our system. Specifically, considering an urban scenario, we assumed a data rate of  $3MBps$  with a packet size of  $256B$  using the IEEE 802.11p WAVE protocol. Given this setting and that both the messages never reached (in our experimentation) a size larger than  $200B$  (see Fig. 15 (b)), the protocol sends, in any case, one package of  $256B$ . This explains why the two messages have comparable performance in terms of transmission delay.

Fig. 16 depicts the analysis of  $M_3$  when varying the number of neighbors. Likewise  $M_2$ , the message

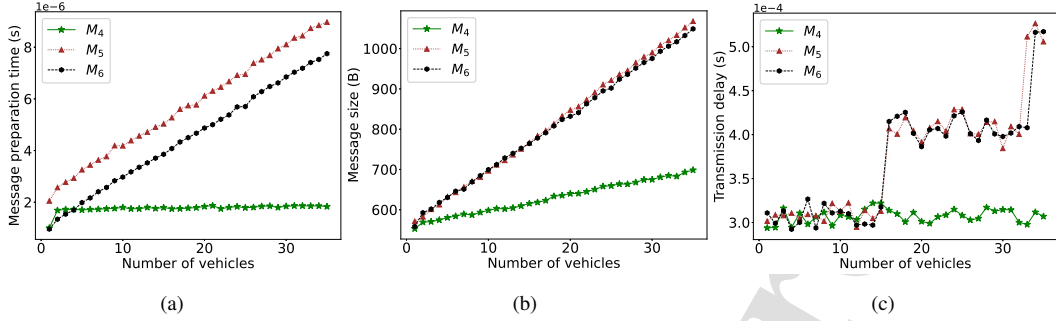


Figure 17: Analysis of messages  $M_4$ ,  $M_5$ ,  $M_6$ : (a) preparation time in seconds, (b) size in bytes, (c) transmission time in seconds.

structure depends on the number of neighbors inside the graph – we can observe a proportional increment of the preparation time (Fig. 16 (a)), message size (Fig. 16 (b)), and transmission delay (Fig. 16 (c)) when augmenting the neighbors. However, since the message  $M_3$  contains not only the neighbors of the sender node but also the neighbors of its neighbors (necessary for the the Jaccard computation), the order of magnitude necessary to prepare the message is around  $10^{-5}$ . Of course, this has an impact on the message dimension whose size ranges between  $50B$  and  $1KB$ ; this, in particular, can be also observed in the transmission delay plot having the shape of a “stair” graph due to the number packets that need to be sent which oscillated between 1 and 4.

Fig. 17 depicts the analysis for the messages  $M_4$ ,  $M_5$ , and  $M_6$ . Since the two entities involved in the communication are the fog and the Cloud, we performed the experiments considering a Wi-Fi connection with  $20MBps$  of data rate and, the same packet size we adopted in the previous assessments. Unlike the previous cases, in this scenario, all the messages’ structures depend on the number of vehicles inside the proximity graph. We observe an increasing trend in the message preparation time (Fig. 17 (a)). The messages  $M_5$  and  $M_6$  exhibit a comparable pattern due to their very similar JSON file structure. A similar trend is also observed for the message  $M_4$  whose dimension is much smaller, thus requiring a lower preparation time which still increases accordingly with the graph dimension.

These trends are even more evident in Fig. 17 (b) where the number of vehicles impacts the message sizes, ranging from about  $600B$  to  $1.1KB$  for  $M_5$  and  $M_6$ , and from  $600B$  to  $700B$  for  $M_4$ . As a result, Fig. 17 (c) shows an area (from 0 to roughly 12 vehicles) where all the three messages have the same transmission time since the message size can be contained in 3 packets. Once this threshold is exceeded, the graphs diverge with  $M_4$  which remains almost constant if we do not consider the oscillations due to the communication channel. The messages  $M_5$  and  $M_6$  follow almost the same pattern (like in Fig. 17 (b)) requiring 4 and 5 packets as the number of vehicles increases.

## 585 8. Conclusion and Future Work

In this paper, we proposed a novel fog-assisted system aiming to reduce the number of Sybils in vehicular crowdsourcing applications, if any. The main core is *SybilDriver*, a Sybil detection approach that leverages the features of VANETs and OSNs to take advantage of both domains. In particular, a combination of proximity graph, Jaccard similarity index, Louvain community detection, and Random Forest is exploited to recruit the best set of participants, i.e., a set of honest participants, and reduce the possibility of false reports. Through extensive experimental analysis, we demonstrated that *SybilDriver* outperforms state-of-the-art schemes in the OSN domain, in terms of detection performance and QoI provided to end-users. The obtained results proved the majority of the generated Sybil vehicles are filtered by the *Community Detection and Ranking*, and *Smart Sybil Detection* algorithms. Moreover, the adoption of the fog architecture helps 595 reduce the network overhead, making the system more suitable for real-world applications.

Our future work will be devoted at improving the proposed approach from different aspects. First of all, we plan to enrich the information available to the system by including features typical of social networks, such as friendships or past interactions between communicating nodes. However, this integration should be evaluated appropriately because, if on the one hand, it should allow the system to improve the estimation about the quality of connections between entities, on the other hand, it could create phenomena that are 600 more difficult to manage [50]; for example, an attacker could exploit this aspect in order to be considered honest, despite being malicious [51].

This situation and many others could be handled through the adoption of a proper *incentive* mechanism, which we intend to be reward-based. The design and definition of this mechanism is very complex as it could lead to an extension of attack strategies by malicious participants. They could engage in more complex behaviors to manage, e.g., increase their reputation to not only influence the network but also to maximize the reward. Then, the reputation module has to be designed in conjunction with the incentive mechanism in order to robustly estimate node trustworthiness and deter malicious behavior in VSN applications. 605

Finally, we aim to address the limitation of our approach, that is, scenarios where no honest vehicles are present. One possible solution to this limitation would be for leveraging expert systems, generally employed to assist in the monitoring, detection, and diagnosis of abnormal and unexpected conditions [52]. Then, abnormal conditions could be modeled and predicted with high confidence by combining expert knowledge with real-time data provided by the vehicular network. 610

**Acknowledgments:** We would like to thank the Associate Editor and anonymous reviewers for insightful comments that helped us improve the quality of the manuscript significantly. This work was partially supported by the National Science Foundation grants DGE-1433659, CNS-1818942, SATC-2030624, and 615



OAC-2104078. The work of Ajay Pratap is partially supported by the Science and Engineering Research Board (SERB), Government of India under grant SRG/2020/000318.

## References

- 620 [1] F. Concone, F. De Vita, A. Pratap, D. Bruneo, G. Lo Re, S. K. Das, A novel recruitment policy to defend against sybils in vehicular crowdsourcing, in: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 105–112. doi:10.1109/SMARTCOMP52413.2021.00035.
- [2] D. V. Le, C. Tham, Y. Zhu, Quality of Information (QoI)-aware cooperative sensing in vehicular sensor networks, in: IEEE PerCom Workshops, 2017, pp. 369–374. doi:10.1109/PERCOMW.2017.7917590.
- 625 [3] Z. Ning, F. Xia, N. Ullah, X. Kong, X. Hu, Vehicular social networks: Enabling smart mobility, IEEE Communications Magazine 55 (2017) 16–55.
- [4] J. Newsome, E. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis and defenses, in: Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004, 2004, pp. 259–268. doi:10.1109/IPSN.2004.239019.
- 630 [5] C. Fiandrino, B. Kantarci, F. Anjomshoa, D. Kliazovich, P. Bouvry, 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.
- [6] M. Sun, M. Li, R. Gerdes, Truth-aware Optimal Decision-making Framework with Driver Preferences for V2V Communications, in: IEEE Conf. on Communications and Network Security (CNS), 2018, 635 pp. 1–9.
- [7] M. S. Bhuvaneswari, K. Muneeswaran, User community detection from web server log using between user similarity metric, International Journal of Computational Intelligence Systems 14 (2020) 266–281.
- [8] F. Dabaghi Zarandi, M. Kuchaki Rafsanjani, Community detection in complex networks using structural similarity, Physica A: Statistical Mechanics and its Applications 503 (2018) 882–891. 640
- [9] H. Hasrouny, A. E. Samhat, C. Bassil, A. Laouiti, Vanet security challenges and solutions: A survey, Vehicular Communications 7 (2017) 7 – 20.

- [10] A. M. Vegni, V. Loscri, A survey on vehicular social networks, *IEEE Communications Surveys & Tutorials* 17 (2015) 2397–2419.
- 645 [11] H. Rowaihy, W. Enck, P. McDaniel, T. La Porta, Limiting sybil attacks in structured p2p networks, in: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 2596–2600. doi:10.1109/INFCOM.2007.328.
- [12] C. Iwendi, M. Uddin, J. A. Ansere, P. Nkurunziza, J. H. Anajemba, A. K. Bashir, On detection of sybil attack in large-scale vanets using spider-monkey technique, *IEEE Access* 6 (2018) 47258–47267.
- 650 [13] S. M. Faisal, T. Zaidi, Timestamp based detection of sybil attack in vanet., *IJ Network Security* 22 (2020) 397–408.
- [14] L. E. Funderburg, I.-Y. Lee, A privacy-preserving key management scheme with support for sybil attack detection in vanets, *Sensors* 21 (2021).
- [15] M. Demirbas, Y. Song, An rssi-based scheme for sybil attack detection in wireless sensor networks, in: *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, 2006, pp. 5 pp.–570. doi:10.1109/WOWMOM.2006.27.
- 655 [16] B. Yu, C.-Z. Xu, B. Xiao, Detecting sybil attacks in vanets, *Journal of Parallel and Distributed Computing* 73 (2013) 746 – 756.
- [17] P. Golle, D. Greene, J. Staddon, Detecting and correcting malicious data in vanets, in: *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, VANET '04*, Association for Computing Machinery, New York, NY, USA, 2004, p. 29–37. URL: <https://doi.org/10.1145/1023875.1023881>. doi:10.1145/1023875.1023881.
- 660 [18] Y. Yao, B. Xiao, G. Wu, X. Liu, Z. Yu, K. Zhang, 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.
- 665 [19] Y. Yao, B. Xiao, G. Yang, Y. Hu, L. Wang, X. Zhou, Power control identification: A novel sybil attack detection scheme in vanets using rssi, *IEEE Journal on Selected Areas in Communications* 37 (2019) 2588–2602.
- [20] X. Li, Q. Lin, J. Mao, Hybrid graph-based sybil detection with user behavior patterns, *Procedia*

- 670 Computer Science 187 (2021) 607–612. 2020 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI2020.
- [21] G. Jethava, U. P. Rao, User behavior-based and graph-based hybrid approach for detection of sybil attack in online social networks, *Computers & Electrical Engineering* 99 (2022) 107753.
- [22] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale  
675 social online services, in: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI'12*, USENIX Association, USA, 2012, p. 15.
- [23] D. Mulamba, I. Ray, 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.
- 680 [24] S. Misra, A. S. Md Tayeen, W. Xu, Sybilexposer: An effective scheme to detect sybil communities in online social networks, in: *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6. doi:10.1109/ICC.2016.7511603.
- [25] A. Breuer, R. Eilat, U. Weinsberg, Friend or Faux: Graph-Based Early Detection of Fake Accounts on Social Networks, *Association for Computing Machinery*, New York, NY, USA, 2020, p. 1287–1297.  
685 URL: <https://doi.org/10.1145/3366423.3380204>.
- [26] M. Fazil, A. K. Sah, M. Abulaish, Deepssbd: A deep neural network model with attention mechanism for socialbot detection, *IEEE Transactions on Information Forensics and Security* 16 (2021) 4211–4223.
- [27] T. Xu, G. Goossen, H. K. Cevahir, S. Khodeir, Y. Jin, F. Li, S. Shan, S. Patel, D. Freeman, P. Pearce,  
690 Deep entity classification: Abusive account detection for online social networks, in: *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, 2021, pp. 4097–4114. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/xu-teng>.
- [28] N. B. Margolin, B. N. Levine, Informant: Detecting sybils using incentives, in: *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security, FC'07/USEC'07*, Springer-Verlag, Berlin, Heidelberg, 2007, p. 192–207.  
695
- [29] J. Lin, M. Li, D. Yang, G. Xue, Sybil-proof online incentive mechanisms for crowdsensing, in: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2438–2446.

- [30] G. A. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, B. Y. Zhao, Social turing tests: Crowdsourcing sybil detection, in: NDSS Symposium 2013, Internet Society, 2013.
- 700 [31] J. Shu, X. Liu, K. Yang, Y. Zhang, X. Jia, R. H. Deng, Sybsub: Privacy-preserving expressive task subscription with sybil detection in crowdsourcing, *IEEE Internet of Things Journal* (2019) 3003–3013.
- [32] V. Agate, F. Concone, P. Ferraro, A resilient smart architecture for road surface condition monitoring, in: M. Ben Ahmed, A. A. Boudhir, İ. R. Karas, V. Jain, S. Mellouli (Eds.), *Innovations in Smart Cities Applications Volume 5*, Springer International Publishing, Cham, 2022, pp. 199–209.
- 705 [33] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, B. Y. Zhao, Ghost riders: Sybil attacks on crowd-sourced mobile mapping services, *IEEE/ACM Transactions on Networking* 26 (2018) 1123–1136.
- [34] M. A. Hasan, M. J. Zaki, *A Survey of Link Prediction in Social Networks*, Springer US, Boston, MA, 2011, pp. 243–275. URL: [https://doi.org/10.1007/978-1-4419-8462-3\\_9](https://doi.org/10.1007/978-1-4419-8462-3_9). doi:10.1007/
- 710 978-1-4419-8462-3\_9.
- [35] P. Srilatha, R. Manjula, Similarity index based link prediction algorithms in social networks: A survey, *Journal of Telecommunications and Information Technology* (2016).
- [36] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks 2008 (2008) P10008.
- 715 [37] X. Solé, A. Ramisa, C. Torras, Evaluation of random forests on large-scale classification problems using a bag-of-visual-words representation., in: *CCIA*, 2014, pp. 273–276.
- [38] H. Jiang, B. Kim, M. Y. Guan, M. Gupta, To trust or not to trust a classifier, in: *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, Curran Associates Inc., Red Hook, NY, USA, 2018, p. 5546–5557.
- 720 [39] S. Bhattacharjee, N. Ghosh, V. K. Shah, S. K. Das, QnQ: Quality and Quantity based Unified Approach for Secure and Trustworthy Mobile Crowdsensing, *IEEE Transactions on Mobile Computing* 19 (2020) 200–216.
- [40] R. P. Barnwal, N. Ghosh, S. K. Ghosh, S. K. Das, Publish or drop traffic event alerts? quality-aware decision making in participatory sensing-based vehicular cps, *ACM Trans. Cyber-Phys. Syst.* 4 (2020).

- [41] F. Concone, A. De Paola, G. L. Re, M. Morana, Twitter analysis for real-time malware discovery, in: 2017 AEIT International Annual Conference, 2017, pp. 1–6. doi:10.23919/AEIT.2017.8240551.
- [42] R. S. Anderssen, S. A. Husain, R. Loy, The kohlrusch function: properties and applications, *Anziam journal* 45 (2003) 800–816.
- [43] S. Uppoor, O. Trullols-Cruces, M. Fiore, J. M. Barcelo-Ordinas, Generation and analysis of a large-scale urban vehicular mobility dataset, *IEEE Transactions on Mobile Computing* 13 (2014).
- [44] F. H. Kumbhar, Vehicular mobility trace at seoul, south korea, 2020. URL: <https://dx.doi.org/10.21227/kjzd-rk05>. doi:10.21227/kjzd-rk05.
- [45] K. Zhang, X. Liang, R. Lu, X. Shen, Sybil attacks and their defenses in the internet of things, *IEEE Internet of Things Journal* 1 (2014) 372–383.
- [46] Q. Xu, Z. Su, B. Han, D. Fang, Z. Xu, X. Gan, Analytical model with a novel selfishness division of mobile nodes to participate cooperation, *Peer-to-Peer Networking and Applications* 9 (2016) 712–720.
- [47] A.-L. Barabási, R. Albert, H. Jeong, Scale-free characteristics of random networks: the topology of the world-wide web, *Physica A: Statistical Mechanics and its Applications* 281 (2000) 69–77.
- [48] A. Bordonaro, F. Concone, A. De Paola, G. Lo Re, S. K. Das, Modeling efficient and effective communications in vanet through population protocols, in: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 305–310. doi:10.1109/SMARTCOMP52413.2021.00064.
- [49] F. Concone, G. Lo Re, M. Morana, A fog-based application for human activity recognition using personal smart devices, *ACM Trans. Internet Technol.* 19 (2019).
- [50] J. Ren, Y. Zhang, K. Zhang, X. Shen, Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions, *IEEE Communications Magazine* 53 (2015) 98–105.
- [51] A. J. Bidgoly, B. T. Ladani, Modelling and quantitative verification of reputation systems against malicious attackers, *The Computer Journal* 58 (2015) 2567–2582.
- [52] M. Latah, Detection of malicious social bots: A survey and a refined taxonomy, *Expert Systems with Applications* 151 (2020) 113383.

**Declaration of interests**

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

--