Trust-related Attacks and their Detection: a Trust Management Model for the Social IoT

C. Marche, M. Nitti
DIEE, University of Cagliari, Italy
National Telecommunication Inter University Consortium - Research Unit of Cagliari - Italy
{claudio.marche, michele.nitti}@unica.it

***Abstract*—The integration of social networking concepts into the Internet of Things (IoT) has led to the so called Social Internet of Things paradigm, according to which the objects are capable of establishing social relationships in an autonomous way with respect to their owners. Within this scenario, "things" interact opportunistically with their peers to seek needed services. However, attacks and malfunctions in the IoT can outweigh any of its benefits if not handled adequately. In this paper, we focus on the possible types of trust attacks that can affect the IoT and propose a trust management model able to overcome all the analyzed attacks. Simulations show how the proposed model can effectively isolate almost any malicious nodes in the network at the expense of an increase in the number of transactions needed for the model to converge.**

***Index Terms*—Internet of Things, Social Internet of Things, Trustworthiness Management, Machine Learning**

I. Introduction

The Internet of Things (IoT) has become a reality with billions of devices able to send key information about the physical world and implementing simple actions, which leads to the paradigm of the anytime and anyplace connectivity for anything [1]. The massive amount of data flowing through the IoT has pushed forward the development of new applications in several domains, such as the management of industrial production plants, the logistics and transport supply chain, the e-health, the smart building, just to cite a few.

Such future IoT applications are likely developed making use of a service oriented architecture where each device can play the role of a service provider or a service requester, or both. IoT is moving towards a model where things look for other things to provide composite services for the benefit of human beings (object-object interaction). With such an interaction model, it is essential to understand how the information provided by each object can be processed automatically by any other peer in the system. This cannot clearly disregard the level of trustworthiness of the object providing information and services, which should take into account the profile and history of it. Although we experience and rely on trust during our interactions in everyday life, trust can have many definitions so that it is challenging to define it accurately. The literature on trust is also quite confusing, since it manifests itself in fairly different forms. In this paper, we adopt the following definition for trust:

Trust is the subjective probability by which an individual, the trustor, expects that another individual, the trustee, performs a given action on which its welfare depends [2].

In the IoT scenario, the requester has the role of the trustor and has to trust that the provider, which is then the trustee, will provide the required service. However, misbehaving devices may perform several types of attacks for their own gain towards other IoT nodes: they can provide false services or false recommendations, they can act alone or create a group of colluding devices to monopoly a class of services. If not handled adequately, attacks and malfunctions would outweigh any of the benefits of the IoT [3] [4]. For example, in February 2020, Simon Weckert transported 99 smartphones in a handcart and was able to generate virtual traffic jam in Google Maps[1]. In this scenario, trustworthiness management models have to solve the important issue to identify and understand which, among the nodes in the network, are trustworthy and can then lead to successful collaborations.

Several works have been proposed to address the problem of trust management in the IoT; however, all these works are usually tested considering only a subset of the possible attack patterns. Indeed, attack patterns are highly heterogeneous so that malicious nodes try to exploit the weak points of trustworthiness algorithms so as to operate unnoticed.

An approach, which is recently gaining increasing popularity and has the potential to properly address this issue, is based on the exploitation of social networking notions into the IoT, as formalized by the Social IoT (SIoT) concept [5]. According to this vision, things create relationships among them as humans do: this approach introduces the vision of social relationships among different devices, so that they are more willing to collaborate with friends w.r.t. strangers. This is expected to make the exchange of information and services among different devices easier and to perform the identification of malicious nodes by creating a society-based view about the trust level of each member of the community.

Our paper works in this direction with the goal to recognize the trustworthiness attacks and thus provides the following contributions:

- First, we analyze *all* the possible types of trust attacks described in the literature that can affect the IoT and briefly review the resiliency of existing models against the identified trust related attacks.
- Second, we propose a decentralized trust management model, based on a Machine Learning algorithm, which makes use of novel parameters, namely the goodness, the

[1] http://www.simonweckert.com/googlemapshacks.html

usefulness and the perseverance score. Thanks to these scores, the model trains and adapts itself and it is able to identify and react to all possible malicious attacks.
- Third, by using a dataset of real IoT objects, we conduct extensive experiments to show how our model reacts to *each* type of attack. Furthermore, we compare our algorithm with two well-acknowledged state-of-the-art models: the experiment results show that even if our algorithm shows the slightly worse performance when under attack by simple mechanisms, it is able to outperform the other two models when considering a network with a mix of different types of attack.

The rest of the paper is organized as follows: Section II presents the scenario of social IoT, a brief survey on trust management models and the possible types of attacks. In Section III, we define the problem, introduce the used notations and illustrate the proposed trust management model. Section IV presents the system performance against all the type of attacks analyzed in Section II while Section V draws final remarks.

II. Background

A. The Social Internet of Things

The SIoT represents the convergence of the technologies belonging to two domains: IoT and Social Networking. The result is the creation of social networks in which things are nodes that establish social links as humans do [5]. According to the SIoT model, every node is an object that is capable of establishing social relationships with other things autonomously, according to rules set by the owner. This concept is fast gaining ground thanks to the key benefits deriving from the potentials of the social networks within the IoT domain, such as: simplification in the navigability of a dynamic network of billions of objects [5]; efficiency in the dynamic discovery, selection and composition of services (and of information segments) provided by distributed objects and networks [6]; robustness in the management of the trustworthiness of objects when providing information and services [7].

When it comes to the IoT paradigm, the idea is to exploit social awareness as a means to turn communicating objects into autonomous decision-making entities. The new social dimension shall, somehow, be able to mimic interactions among users and to motivate a drift from an egoistic behavior to altruism or reciprocity. The main principle is to enable objects to autonomously establish social links with each other (by adhering to rules set by their owners) so that "friend" objects exchange data in a trustworthy manner. According to this model, a set of forms of socialization among objects is foreseen. The parental object relationship (POR) is defined among similar objects, built in the same period by the same manufacturer (the role of the family is played by the production batch). Moreover, objects can establish a co-location object relationship (CLOR) and co-work object relationship (CWOR), like humans do when they share personal (e.g., cohabitation) or public (e.g., work) experiences. A further type of relationship is defined for objects owned by the same user (mobile phones, game consoles, etc.) that is named ownership object relationship (OOR). The last relationship is established when objects come into contact, sporadically or continuously, for reasons purely related to relations among their owners (e.g., devices/sensors belonging to friends); it is named social object relationship (SOR). These relationships are created and updated on the basis of the objects' features (such as type, computational power, mobility capabilities, brand, etc.) and activities (frequency in meeting the other objects, mainly).

However, to fully exploit the benefits of a SIoT network, a trustworthiness management model, able to defend against malicious attacks, is needed, which we investigate in this paper.

B. Trustworthiness Models

This subsection provides a brief overview regarding the background of trustworthiness management in the IoT. In the last years, many researchers have tackled this problem, so that the literature is now quite rich. In this Section, we want to show the most appreciated models in the literature and do not intend to cover all the published papers. We classified them into three categories based on the metric used to compute the trust value: metrics obtained from social aspects, metrics based on the Quality of Service (QoS) and mixed approaches, i.e. papers considering both social and QoS aspects.

Among the works considering social aspects, in [8] the authors propose an adaptive decentralized trust mechanism based on social trust. Through a weighted sum, the authors combine factors that concern the cooperativeness and the social communities and demonstrate the effectiveness of the model making use of two real-world social IoT scenarios. Another trust model concerning social trust is presented in [9]. Authors propose a machine learning-based approach to formalize the trust evaluation as a classification problem. The feature vector in a social network is constructed according to social factors like the reputation and the centrality. Another social approach is used in [10]. Throughout a few SIoT trust metrics as centrality, community interest and cooperativeness, the authors illustrate a trust management scheme to facilitate an automatic trustworthy decision making based on the behavior of smart objects. Two social scenarios are described in [11] and [12]. In the first work, the authors take into account metrics such as social similarity and the importance of the service. The resulting trust management algorithm is developed using social relationships to compute the trust level of the nodes in a SIoT network. In the second one, the authors propose a centralized trust-based protocol for mobile objects. To guarantee the trust accuracy between the devices the system makes use of friendships and social contacts.

Concepts of QoS are used for example in [13]: authors present a remote attestation mechanism for the sensing layer node in the IoT. A real-time trust measurement is realized through a combination of QoS factors, such as transmission delay, historical data and feedback originated from other objects. Firstly, a node verifies the identities of the other nodes and only then measures whether the computing environment is trustworthy. In [14], the authors compute the trust scores based on the exchange of feedback, which are provided taking

into account QoS factors such as the monetary cost of the resources, the computation capabilities and the communication failures. Two other QoS approaches based on centralized architectures are described in [15] and [16]. In the first work, the authors propose a policy-based secure scheme for IoT, in which the trustworthiness of data and devices are evaluated according to the reporting history and the context in which the data are collected. In the second one, the centralized architecture is used for information sharing among health IoT devices. The proposed trust protocol considers the loss of probability of health data and the reliability of the IoT devices. Another approach concerning QoS factors is presented in [17]. Authors introduce an approach to evaluate the trust of services combining several QoS attributes (such as availability and response time) and user's ratings. The model focuses on satisfying the users' choices on web services and it is evaluated considering the influence of malicious rating.

The last group of papers makes use of both QoS and social trust metrics to compose the trust value. Among them, in [18], the authors propose a decentralized trust mechanism in a social scenario. In that model, each node computes the trustworthiness of the service providers on the basis of its own experience and on the opinion of its friends. The authors analyze the QoS factors, as computation capabilities, and social factors, such as centrality and credibility. QoS and social metrics are both considered also by Chen et al. in [19]. They adopt a distributed scheme where each node maintains its own trust assessments. The QoS factors (i.e., quality reputation and energy status) are related to the social relationships and recommendations from the other nodes. Two other mixed approaches are described in [20] and [21]. In [20], authors propose a trust evaluation model incorporating heterogeneous information from direct observation, personal experiences and global reputation. The subjective algorithm makes use of social factors, e.g. cooperativeness and community-interest, and of QoS factors, aggregated with a weighted sum mechanism and a machine learning to change the weights according to the particular context. In [21], the authors illustrate an IoT protocol that uses trust for the evaluation of nodes to make optimal routing decisions. It computes the trust of nodes by examining QoS factors, such as the number of exchanged packets, and the recommendations from the neighbors. A recent model is described in [22]. Authors propose guidelines for the design of a decentralized trust management model, which can be used for assisting humans and devices in the decision making process.

All the analyzed models are designed and tested to isolate nodes that implement a subset of the possible types of attacks. However, the heterogeneity of IoT scenarios call for models with no weak points, while existing works show a common limitation: they are not able to properly identify all the type of malicious attacks. The next subsection shows all the possible malicious behaviours that can be implemented in a network.

C. Trustworthiness Attacks

Two different behaviors can be considered in a network [23]: one is always benevolent and cooperative, while the other one is a strategic behavior corresponding to an opportunistic participant who cheats whenever it is advantageous for it to do so. The goal of a node performing maliciously is usually to provide low quality or false services in order to save its own resources; at the same time, it aims to maintain a high value of trust toward the rest of the network so that other nodes will be agreeable to provide their services when requested. This strategy, even if successful for a single node at first sight, involves a huge risk for the network because trusting the information from malicious devices could lead to serious compromises within the network and this has a direct impact on the applications that can be delivered to users [24]. A trust model has to identify this behaviour to discourage nodes from implementing it; however, such malicious nodes can perform several types of trust-related attacks, which represent the different solutions they adopt to avoid being detected. We classify trustworthiness attacks based on two dimensions: the first dimension is related to the **target** of the attack, i.e. if the malicious node aims to confuse the network by providing false services, false recommendations or both. The second dimension is connected to the **size** of the attack, i.e. if the trustworthiness attack is carried on by a single node or by a group. In the following, we briefly describe the different types of attacks known in the literature.

The largest group of attacks is composed of single nodes that indiscriminately provide both bad services and recommendations. In this group, trustworthiness attacks differ based on the mechanism they adopt in order not to be recognized:

Malicious with Everyone (ME): a malicious node acts maliciously with everyone. This is the most basic attack: a node always provides bad services and recommendations, regardless of the requester [18].

Discrimination Attack (DA): a malicious node modifies its behavior based on the service requester. This means that a node can discriminate non-friends nodes or nodes with weak social ties. As a result, some devices can consider the node as benevolent while others can label it as malevolent [25].

On-Off Attack (OOA): a node periodically changes its behavior, by alternatively being benevolent (ON) and malevolent (OFF). During the ON state, the node builds up its trust, which is then used to attack the network [26].

Whitewashing Attack (WA): a node with a bad reputation leaves the network and then registers again with a different identity. When the node re-join the network its reputation is reset to a default value [27].

Self-Promoting Attack (SPA): a malicious node provides good recommendations for itself in order to be selected as a service provider. After it is selected as a provider, it provides only bad services [28].

The other types of attacks concentrate on a single target, i.e. malicious nodes only provide bad services or bad recommendations.

Bad Mouthing Attack (BMA): this attack is addressed to ruin the reputation of other nodes; a malicious node only provides false recommendations to decrease the chance of benevolent nodes being selected as providers. Usually, this attack is part of a collusive behavior where a group of nodes

TABLE I
CLASSIFICATION OF DIFFERENT TYPES OF TRUSTWORTHINESS ATTACKS.

		Size	
		Single	*Group*
Target	*Service*	[OSA]	
	Recommendation	[ME] [DA] [OOA] [WA] [SPA]	[SA] [BSA]
		[BMA]	

TABLE II
RESILIENCY OF EXISTING MODELS AGAINST IDENTIFIED TRUST RELATED ATTACKS.

Ref	DA	OOA	BSA	WA	BMA	SA	SPA
[8]	✓	-	✓	✓	✓	-	✓
[9]	-	-	-	-	-	-	-
[10]	-	✓	-	-	-	-	-
[11]	-	-	✓	-	✓	-	✓
[12]	✓	-	✓	-	✓	-	✓
[13]	-	-	-	✓	-	✓	✓
[14]	-	-	-	-	-	-	
[15]	-	✓	✓	-	✓	-	-
[16]	-	-	✓	-	✓	-	-
[17]	-	-	-	-	✓	-	-
[18]	✓	✓	-	-	-	-	✓
[19]	✓	✓	-	-	✓	-	✓
[20]	-	-	✓	-	✓	-	✓
[21]	-	-	-	-	-	✓	-
[22]	-	-	✓	-	✓	-	-

works together to ruin the reputation of a good node but it can also be carried on by a single node [29].

Ballot Stuffing Attack (BSA): this is a type of collusive attack, where a malicious node provides good recommendations toward another malicious node to boost its reputation and increase its chances to be selected as the provider [30].

Sybil Attack (SA): a malicious node uses multiple identities to provide different types of recommendations on the same service. These multiple identities are usually fake and they are all responsible for the attack process [31].

Opportunistic Service Attack (OSA): a malicious node provides good services only when it senses that its trust reputation is dropping. In this way, the node tries to maintain an acceptable level of trust in order to still be selected as a service provider [32].

To sum it up, Table I shows a classification of trust-related attacks based on the two dimensions identified, while Table II compares the analyzed models with the attacks they are able to identify. To the best of our knowledge, all available trustworthiness models are able to isolate only a subset of the presented attacks, i.e. they are designed to recognize and isolate some specific attacks, but none of them is able to defend efficiently against all the attacks. Table II does not show the ME attack, which is used as a reference attack by all the models, and the OSA attack since a node performing it can not be completely isolated but it is only possible to reduce the number of times a node acts maliciously due to the reliability needed to build up the trust.

These attacks span from simple ones, which have a constant behaviour over time, such as ME, to more complex ones which are able to change their behaviour over time: among them, for example, there is the On-Off Attack, the Discriminatory Attack or the Opportunistic Service Attack, which have all been tested in our paper. In particular, the OSA is considered the most complex attack in the literature since it knows exactly how the trust model implemented in a system works, so it is able to accurately predict how its trust value will change based on its behaviour and then behaves accordingly.

III. PROPOSED SOLUTION

A. Notation and Problem Definition

The focus of this paper is to propose a trust management model able to identify all the trust attacks analyzed in Section II-C and isolate the nodes performing them. In our modeling, the set of nodes in the Social IoT is represented by $\mathcal{N} = \{n_1, ..., n_i, ...n_I\}$ with cardinality I, where n_i is the generic node. The resulting social network, created by the devices' relationships, can be described by an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{E} \subseteq \{\mathcal{N} \times \mathcal{N}\}$ is the set of edges, each representing a social relation between a couple of nodes. The friends of the generic node n_i are represented in our model by $\mathcal{N}_i = \{n_j \in \mathcal{N} : n_i, n_j \in \mathcal{E}\}$, that is the set of nodes that share a relation with it; moreover, we define $\mathcal{H}_{ij} = \{n_h \in \mathcal{N} : n_h \in \mathcal{N}_i \cap \mathcal{N}_j\}$ as the set of common friends between n_i and n_j.

Every node in our network can provide one or more services, so that $\mathcal{S}_j$ is the set of service that can be provided by n_j. The reference scenario is then represented by a node n_i requesting a particular service S_h: a Service Discovery component in the network is able to return to n_i a list of potential providers $\mathcal{P}_h = \{n_j \in \mathcal{N} : S_h \in \mathcal{S}_j\}$. At this point, the requester has to select one of the providers in $\mathcal{P}_h$ based on their level of trust. The trust level is usually computed based on the previous interactions among the nodes. Indeed, after every transaction l, the requester n_i assigns feedback to the selected provider n_j to evaluate the service: we can then define the set of feedback $\mathcal{F}_{ij} = \left\{f_{ij}^1, ..., f_{ij}^l, ...f_{ij}^{L_{ij}}\right\}$, where l indexes from the latest transactions ($l = 1$) to the oldest one ($l = L_{ij}$), so that L_{ij} represents the total number of transactions between the two nodes. Each feedback can be expressed using values in the continuous range $[0, 1]$, where 1 is used when the requester is fully satisfied by the service and 0 otherwise.

Figure 1 provides a simple example of a generic graph $\mathcal{N} = \{n_1, ..., n_9\}$, with each node capable of providing one or more services, as highlighted in the grey clouds; n_1 is the node that is requesting the service S_7, as highlighted in the white cloud; $\mathcal{P}_h = \{n_5, n_6\}$ is the set of nodes that can provide the requested service. In this figure, we also highlight the set $\mathcal{N}_1 = \{n_2, n_3, n_4\}$ of nodes that are friends of n_1 (in light blue color). Within note that the set $\mathcal{H}_{15} = \{n_2, n_4\}$ and the set $\mathcal{H}_{16} = \{n_4\}$ of nodes represent the common friends between n_1 and n_5 and between n_1 and n_6, respectively. For each of the provider in $\mathcal{P}_h$, the requester n_1 computes the trustworthiness levels, T_{15} and T_{16}, and then chooses the provider with the highest value, which is n_5 in our example.

The goal of any trustworthiness management model is to compute and list the trust level of all the providers. This step is

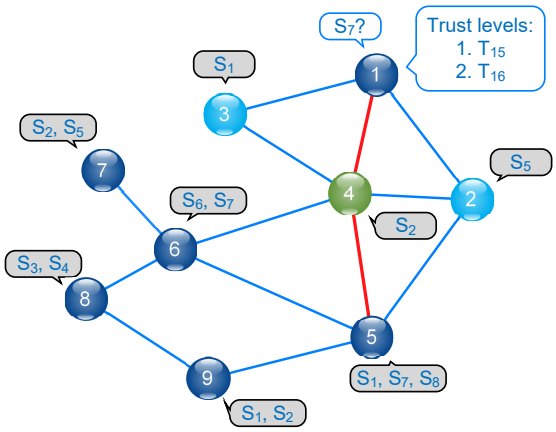


Fig. 1. Trust Management Model.

fundamental to help the requester to identify the most reliable node to whom require the service and to avoid any malicious node. In our model, we envision that each node n_i computes the trustworthiness level T_{ij} of all the possible providers n_j on its own, so that different nodes can make different choices when selecting a provider based on their past experiences.

B. Trust Management Model

According to the presented scenario, we propose a decentralized scenario, where each node calculates and stores information about the other nodes, so to have its own opinion about the network: in this way, malicious attacks that change their behaviour based on the requester, such as DA, are easily identified. Whenever a node n_i has to evaluate the trustworthiness of another node n_j, it computes the trust value as follows:

$$T_{ij} = \alpha^{L_{ij}} C_j + \beta^{L_{ij}} R_{ij} + \gamma^{L_{ij}} O_{ij} + \delta^{L_{ij}} S_{ij} \qquad (1)$$

All these addends are in the range $[0, 1]$ and the weights are selected based on the total number of transactions L_{ij} between node n_i and n_j. Moreover, the weights' sum, namely $\alpha^{L_{ij}} + \beta^{L_{ij}} + \gamma^{L_{ij}} + \delta^{L_{ij}}$, is always equal to 1, in order to normalize the trust value in the interval $[0, 1]$, and their relative value can be changed to give more impact to a particular parameter.

A generic node n_i evaluates the trustworthiness T_{ij} based on four parameters: the *Computation Capabilities* C_j of the service provider, the *Relationship Factor* R_{ij} between the two nodes, the *External Opinions* O_{ij} provided by n_i's friends and the *Dynamic Knowledge* S_{ij} acquired by the requester. The Dynamic Knowledge represents the core of our system, which has to learn how to identify malicious nodes. This ability is tied to its experience, i.e. to the past transactions of the node. Accordingly, the proposed trustworthiness model is divided into two phases: a training phase and a steady-state phase. In the training phase, the contribution of the Dynamic Knowledge is limited, because the requester is trying to learn the behavior of the provider: since the requester has to understand the behavior of each node it interacts with, the four weights are dependent by both the requester and the provider; we omit this dependency to avoid too much confusion in the presented equations. In particular, the value of $\delta^{L_{ij}}$ grows with the total number of transactions L_{ij} between the requester n_i and the provider n_j, as follows:

$$\delta^{L_{ij}} = \begin{cases} (L_{ij} - 1)/L^{tr} & \text{for } L_{ij} \leq L^{tr} \\ 1 & \text{for } L_{ij} > L^{tr} \end{cases} \qquad (2)$$

where L^{tr} represents the number of transactions needed to train the Dynamic Knowledge. The residual weight, i.e. $1 - \delta^{L_{ij}}$, is then shared among the other weights.

1) Training Phase: The goal of this phase is to let the Dynamic Knowledge factor collects enough experience. Until this happens, the trust value of the potential providers is calculated based on the elements described below.

The **Computation Capabilities** C_j is a static characteristic of an object which does not vary over time. This factor accounts for the heterogeneity of the IoT where some devices are more powerful than others so their ability to act maliciously is higher and they can lead to more uncertain transactions. To take into account this possibility, the model assigns lower values to objects with great computational capabilities w.r.t. devices with only sensing and actuation capabilities.

The **Relationship Factor** R_{ij} is a unique characteristic of the SIoT and it is related to the relationships that ties node n_i and n_j. Using [18] as a starting point, we set the greatest value for the OOR relationship and decreasing values for the other relations. If two nodes are tied by two or more relationships, e.g. they have created both an OOR and a SOR, we consider the strongest relation which then they have with the highest value. If two nodes have no direct relation, the model computes the sequence of social links between them and consider the weakest link in the path, i.e. the minimum value of all the relationship factor. To account for the uncertainty of the intermediates nodes, this value is further divided for the number of hops that separate node n_i and node n_j.

The **External Opinion** O_{ij} evaluates the recommendations provided to n_i by the friends in common with n_j, namely the nodes in $\mathcal{H}_{ij}$ and is expressed as:

$$O_{ij} = \sum_{h=1}^{|\mathcal{H}_{ij}|} T_{ih} \cdot T_{hj} \Bigg/ \sum_{h=1}^{|\mathcal{H}_{ij}|} T_{ih} \qquad (3)$$

where T_{hj} represents the opinion, i.e. the trust value, that each of the common friends n_h has for node n_j. These values are weighted with the trust values that node n_i has already computed towards its friends, so that the opinion of trustworthy nodes is considered more than the one from low trustworthy nodes. Indeed, recommendations represent an effective strategy, adopted by many trust algorithms, to easily obtain information regarding other nodes. This is especially true when a node's direct experience is still scarce. However, they are also exploited by many trustworthiness attacks, such as BMA and BSA, to confuse the network: using the external opinion only in the training phase, our model is resilient to all these types of attacks.

Moreover, at the end of each transaction, n_i assigns a feedback not only to the provider but also to the friends in

$\mathcal{H}_{ij}$, which have contributed to the computation of the external opinion. According to Eq. 4, if a node provided a positive opinion, it receives the same feedback as the provider, i.e. a positive feedback if the transaction was satisfactory, $f_{ij}^l \geq 0.5$, and a negative one otherwise, $f_{ij}^l < 0.5$. Instead, if n_h gave a negative opinion, then it receives a negative feedback if the transaction was satisfactory and a positive one otherwise.

$$f_{ih}^l = \begin{cases} f_{ij}^l & \text{if } T_{hj} \geq 0.5 \\ 1 - f_{ij}^l & \text{if } T_{hj} < 0.5 \end{cases} \tag{4}$$

Moreover, to further reduce the possibility of attacks on the recommendations, in our algorithm, a node uses them only in the training phase to accumulate experience and then it only relies on its Dynamic Knowledge.

2) Steady-State phase: After the training phase, only the Dynamic Knowledge is used to evaluate the possible providers. According to the presented scenario, certain types of malicious nodes, e.g. OOA and OSA, continuously change their behaviour. In order to address this issue, the Dynamic Knowledge must be able to continuously learn and adapt to the myriad of possible malicious behaviours. To compute its value, we make use of an incremental Support Vector Machine (iSVM), so that a node can constantly extends its knowledge after a new transaction: in particular, a SVM is a supervised learning model that analyzes a set of data, in our case the first L^{tr} transactions, to provide some sort of classification. SVM algorithms have been applied to solve a variety of applications [33]. With respect to other machine learning algorithms, the risk of over-fitting is less, it is relatively memory efficient and is effective when there is a margin of separation between classes. The accuracy of this classification is tied to the number of historical data obtained [34]: in our case, the output of the SVM represents the probability that a service provider is benevolent or not, i.e. its trust value. More details regarding the validation process of the iSVM and a comparison with other incremental machine learning algorithms will be presented in Section IV-B.

After every transaction, the Dynamic Knowledge is updated, so that it is able to learn from its past experience and can provide a more accurate evaluation. Since we make use of an incremental SVM, with each new transaction the model's knowledge is extended and updated, without the need to train the SVM from scratch. This way, each node can implement a dynamic Machine Learning algorithm even with limited resources and active learning is much faster w.r.t. a traditional approach. In order to train the SVM, past transactions are expressed in terms of scores, which have the goal to highlight different aspects of the interaction among nodes. Three scores are used as inputs for the Dynamic Knowledge, which are able to evaluate the entire history of the nodes as well as their recent behavior. In this way, the attacks with a dynamic behaviour over time, such as OOA and OSA, can be recognized. The first score is the **Goodness Score**: this score enables the SVM to evaluate nodes on a long-term period and measures how benevolent the node has been during all its transactions. The score is evaluated as the fraction of all the "good" transactions, i.e. all the transactions evaluated in a positive way by the requester:

$$G_{ij} = \frac{|\left\{ f_{ij}^l \in \mathcal{F} : f_{ij}^l > TH \right\}|}{L_{ij}} \tag{5}$$

where TH is the threshold a requester set to consider services as "good". High values of this score mean that the service requester is overall satisfied by the services obtained from the provider. This factor is also useful to identify benevolent nodes which provide services with low accuracy that a requester would like to avoid and that are then labeled with a low value of the Goodness Score.

However, the Goodness Score is not able to react to sudden changes in the behavior of a node, as it happens for dynamic attacks such as OOA and OSA. To overcome these attacks, we make use of two other scores, which evaluate the behavior of the service provider considering a small temporal window, which makes use of the last L^s transactions.

The **Usefulness Score** is used to evaluate only the recent behavior of a node, as follows:

$$U_{ij} = \sum_{l=1}^{L^s} w^l \cdot f_{ij}^l \tag{6}$$

where, in order to give more relevance to the latest transaction w.r.t. the oldest one, the weights w^l of each feedback follows a geometric distribution with parameter ρ

$$w^l = \rho(1-\rho)^{l-1} + (\xi^{res}/L^s) \tag{7}$$

to maintain the score in the range $[0, 1]$, we introduce the term ξ^{res} which account for all the residual weight of the distribution due to the transactions older than L^s. ξ^{res} is then computed as:

$$\xi^{res} = \sum_{r=L^s+1}^{L_{ij}} \rho(1-\rho)^{r-1} \tag{8}$$

The **Perseverance Score** evaluates the constancy of a node in providing good services and it is computed as:

$$\begin{cases} P_{ij}^{L_{ij}} = 0.5 & \text{if } L_{ij} = 1 \\ P_{ij}^{L_{ij}} = P_{ij}^{L_{ij}-1} + V_{ij} & \text{if } L_{ij} > 1 \end{cases} \tag{9}$$

where V_{ij} is a parameter that reward/punish a node based on its constancy in providing good/bad services, as described by:

$$V_{ij} = \begin{cases} v_{ij}u & \text{for } f_{ij}^{L_{ij}} \geq TH \\ -v_{ij}d & \text{for } f_{ij}^{L_{ij}} < TH \end{cases} \tag{10}$$

u and d represent the basic increase/decrease of the score; however, consecutive good or bad transactions can further reward/penalize a node, which is then encouraged to stay benevolent, according to the value of v_{ij}: this value is calculated as the number of consecutive transactions evaluated positively/negatively by the requester. As the other scores, also the Perseverance Score is limited in the interval $[0; 1]$; in the event the score obtained from Eq. 9 is out of these bounds, its value is set to the nearest bound.

IV. Experimental Evaluation

A. Simulation Setup

In order to test our trustworthiness model, we need a large dataset of a SIoT scenario. To this, we make use of the dataset made available by [35]; it consists of a network of 16216 devices owned by 4000 users and by the municipality of Santander (Spain), which create their own relations over 11 days. Moreover, the authors share a set of real services and applications offered and requested by the nodes, which are useful to emulate interactions among nodes. We decide to consider only a connected sub-network of around 800 nodes to increase the probability of two nodes interacting with each other. Furthermore, a model of interaction among the nodes is also needed to understand which devices are more likely to interact; trust models are usually tested considering random interactions among nodes without taking into account the behavior of objects that generate queries of services when interacting with the other peers. To this, we have adopted the query generation model presented in [36], so that at the start of each transaction, the simulator can choose the requester and select all the possible providers.

Two main behaviors are implemented in the network: one is cooperative and benevolent, so that a node always provides good services and recommendations. The other one is a malevolent behavior, where a node tries to disrupt the network by implementing one of the trust attacks presented in Section II-C. Table III shows the optimal configuration of the simulation parameters for the proposed system, and the different weights used for the model. For simplicity, we suppose that the service requester is able to perfectly rate the received service providing binary feedback: 1 for satisfactory services and 0 otherwise. Finally, Table IV presents the values for the relations created by the objects and for their computation capabilities. Between two objects that belong to the same owner and then are linked by an OOR, the relationship factor has been assigned with the highest value. CLORs have been set with only a slightly lower value since they are established between domestic objects and objects of the same workplace. SORs are relationships established between objects that are encountered occasionally (then owned by acquaintances) and for this reason a smaller value is given. Finally, the PORs are the riskiest, since they are created between objects of the same brand but that never met and depend only on the model object. If two nodes are tied by two or more relationships, the strongest relation with the highest factor is considered. Computation capabilities are divided into two classes: Class1 is assigned to objects with only sensing capabilities, that is, an object just capable of providing a measure of the environment status and to the RFID-tagged objects. Class2 is assigned to objects with great computational and communication capabilities; to this class belong objects such as smartphones, tablets, vehicle control units, displays, set top boxes, smart video cameras.

To find the optimal setting for the residual weight, i.e. $1-\delta^{L_{ij}}$, we analyze the model's response at varying the other weights, namely α, β and γ. Table V displays the transaction success rate when the system has reached the steady-state phase. As expected, the external opinion has more impact than the static characteristics, since it can help to identify malicious behaviors, however, since we are considering the startup phase, they are still useful when there is no information available.

L^{tr} is selected based on the machine learning algorithm validation. As shown in the next section, the selected value ensures a sufficient initialisation for the iSVM algorithm and an efficient prediction in the classifications. The ρ parameters guarantees a compromise in the evaluation of the feedback: a value close to 1 only considers the newest feedback, while a value close to 0 considers all the feedback as equally important. Finally, u and d are picked asymmetric in order to encourage benevolent behaviours and punish malicious nodes.

TABLE III
Simulation Parameters

Parameter	Description	Value
α	Residual weight of the Computation Capabilities	0.3
β	Residual weight of the Relationship Factor	0.3
γ	Residual weight of the External Opinion	0.4
L^{tr}	Number of transactions to train the Dynamic Knowledge	5
TH	Threshold to consider a service as "good"	0.5
L^{s}	Temporal window to compute Usefulness and Perseverance Score	10
ρ	Parameter of the geometric distribution	0.4
u	Basic increase of the Perseverance Score	0.1
d	Basic decrease of the Perseverance Score	0.2
I	Number of nodes in the network	791
	Percentage of malicious nodes	25%

TABLE IV
Parameters for Relationship Factor and Computation Capabilities

Relationship Factor

Relationship	OOR	C-LOR	SOR	POR
$\mathbf{R_{ij}}$	1	0.9	0.6	0.5

Computation Capabilities

Capabilities	Class 1	Class 2
$\mathbf{C_j}$	1	0.4

TABLE V
Parameters Settings

α = 0.1	β = 0.1	γ = 0.8	SR = 0.83
α = 0.1	β = 0.8	γ = 0.1	SR = 0.82
α = 0.8	β = 0.1	γ = 0.1	SR = 0.81
α = 0.3	β = 0.3	γ = 0.4	SR = 0.85

B. Simulation Results for ML algorithms

This Section aims to validate the performance of the incremental SVM (iSVM) algorithm and to compare it with other incremental machine learning algorithms.

In order to validate the performance of the algorithms we have used the Receiver Operating Characteristic (ROC) curve and Area Under the ROC (AUC) curve as performance metrics. The ROC represents the diagnostic ability of a binary classifier system, i.e. the true positive rate versus the false positive rate at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The measure of performance between the algorithms is provided by the AUC, which indicates how much a model is capable of distinguishing between classes: a model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0. We compare the performance of the iSVM with two well-known incremental algorithms, the incremental Logistic Regression (iLR) [37] and an incremental artificial neural network, the incremental Radial Basis Function network (iRBF) [38]. The testing network used for the validation is composed by a requester interacting with nodes, as providers, that implement each a different behaviour, from benevolent to all of the seven possible attacks. We vary the number of total transactions among the requester and all the providers to study the ability to learn of the algorithms; we consider that out of all the transactions, 70% of them are used to train the incremental models while the remaining 30% are used for the validation. Figure 2 shows the trend of the ROC curve for 4 experiments based on 160, 650, 1600 and 3200 transactions of the requester. Considering all the possible providers, this means that the number of transactions used for validation with each node is 6, 25, 60 and 120 transactions. The Figure shows how the iSVM is able to outperform the other two algorithms: except for the first set of simulations, with only 6 transactions per node used for validation, the iSVM has the best values of AUC: the system continuously learns from the processed data so that the iSVM increases its percentage of correct predictions with the growth of the dataset of transactions. Moreover, even if the accuracy of the iSVM is low when considering few transactions per node, the proposed model is able to mediate it thanks to the training phase, which makes use of other parameters to obtain higher accuracy in selecting trustworthy nodes.

C. Simulation Results for Trust Management Model

We evaluate the performance of the proposed system by analyzing the success rate, i.e. the ratio between the number of successful transactions and the total number of transactions, or by directly calculating the level of trust computed by a node.

We compare the performance of the proposed model with two well known models by the research community that, similar to our model, are designed for the same scenario, i.e. Social IoT scenario, namely the model proposed by Nitti et al. [18] and the one presented in [19] by Chen et al. Both these models make use of a subjective approach where every node has its own vision of the network and relies

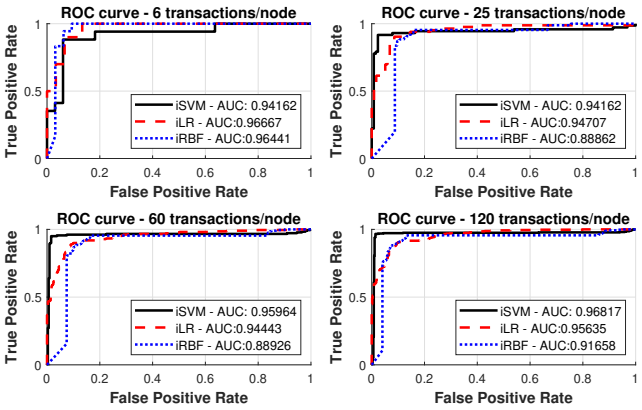


Fig. 2. ROC curves for the machine learning algorithms for 4 experiments based on 6, 25, 60 and 120 transactions per node.

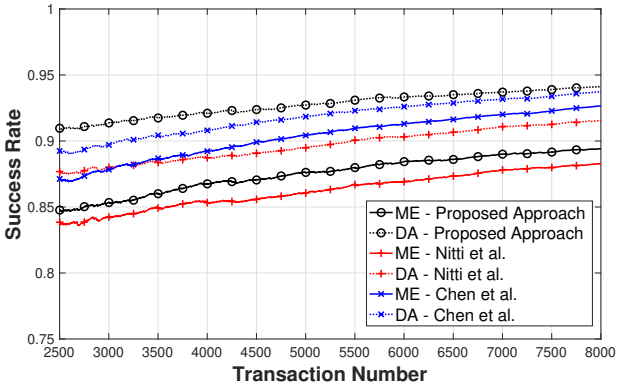


Fig. 3. Transaction success rate for two classes of trust attacks.

on the recommendations from its friends to speed up the evaluation of trust. Differences in the performance of the models can depend on the structure of the social network considered and on the types of service/information requested. To this, we did not consider our ad-hoc social network but we have adopted the Social IoT dataset described in the previous subsection, opportunistically re-scaled to a size comparable to their experiments. Moreover, we have considered the same requests for all the three models, so we are confident that the obtained results are consistent with those obtained by the authors.

These comparisons are aimed at analyzing the improvements we obtain with respect to the state of the art in the specific reference SIoT scenario. We tested all the different types of attacks, except for the SA and the SPA, which are avoided by default in our system: even if a node creates multiple identities or provides good recommendations for itself, the computed trust can not be influenced.

Figure 3 shows the transaction success rate when malicious nodes implement two trust-related attacks, ME and DA. We consider that 25% of the nodes are malicious and in the case of the DA, they only act maliciously with nodes that they meet occasionally or they have never met, i.e. with nodes they have a weak relation with, such as POR and SOR. All the models

TABLE VI
PERFORMANCE COMPARISON OF THE THREE MODELS AGAINST THE DISCRIMINATORY ATTACK.

		Proposed	Chen et al.	Nitti et al.
Requester	n_1	$T_{14} = 0.97$	$T_{14} = 0.81$	$T_{14} = 0.83$
	n_2	$T_{24} = 0.97$	$T_{24} = 0.8$	$T_{24} = 0.8$
	n_3	$T_{34} = 0.02$	$T_{34} = 0.31$	$T_{34} = 0.42$

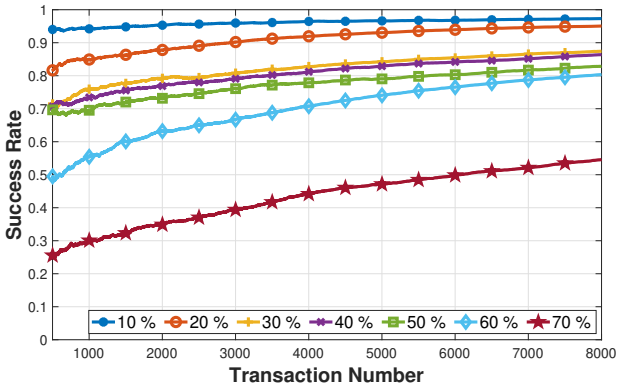


Fig. 4. Transaction success rate at increasing values of % of malicious nodes.

have a good reaction to these two attacks and are able to achieve a high success rate, ranging from 88% to 94%. This is not a surprising result, since both these attacks are usually the ones used to test trustworthiness models. All the implemented algorithms have a better performance to the Discriminatory Attack w.r.t. to the ME, even if devices implementing ME are easier to be identified since they do not behave differently according to the requester: this can be explained considering that due to the changing behavior of the DA, the total number of transactions in which a node acts as malicious are only a subset of all its transaction.

To better understand how the three models react to the DA, we set up a small network of 4 nodes fully connected, where 3 benevolent nodes, n_1, n_2 and n_3, have 15 interactions each with one malevolent DA node n_4. Only the relation $\{n_3, n_4\}$ is weak, so n_4 only behaves maliciously with n_3 and benevolent with n_1 and n_2. The results are shown in Table VI: as expected, in all the models, both n_1 and n_2 have a high trust value for n_4 while, despite n_3 is able to identify n_4 as a malevolent node in all the models, the trust value obtained is highly variable. Only our proposed model assigns a really low trust value to n_4, while the other two models compute higher values due to the strong influence of the common friends within their algorithms.

We now want to analyze the results at varying percentage of the malicious nodes. Figure 4 refers to a scenario where all the malicious nodes implement ME: it shows that even with 70% of malicious nodes the success rate is over 50% and the algorithm is still able to converge. This happens since every node has its own vision of the network based on the acquired Dynamic Knowledge, however, the accuracy decreases, since it increases the possibility that all the available service providers are malicious. We need more than 75% of malicious nodes for

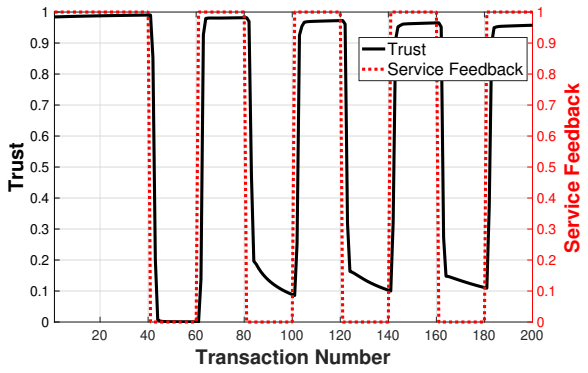


Fig. 5. Trust value of a malicious node that performs an On-Off Attack.

the success rate to drop below 0.5: we have run a similar test also for the other two algorithms: Chen's algorithm is able to resist over 80% of malicious nodes while Nitti's performance is similar to our with 75% of malicious nodes. This result is related to the subjective approach of these two models, where each node takes its own decisions.

The focus of the next set of simulations is to test how the proposed model works with the dynamic behavior of the nodes, i.e. against the OOA. We suppose that after 40 transactions, a malicious node starts to change its behavior from benevolent to malicious and vice versa every 20 transactions. Figure 5 illustrates the trust value of a node performing such attack and shows how the algorithm is able to quickly adapt to the changes in the node behavior: only 3 transactions are needed to modify the trust value of the malicious node, both when the node is exploiting its good reputation and when it is trying to build up its trust. Table VII presents a comparison with the other two models in terms of the number of transactions needed to change the trust value past 0.5 and highlighting the initial and final trust, T_i and T_f respectively, computed before and after the changing behavior. We note how our model is the fastest one to recognize the dynamic behavior so that only a node changing its behavior every 2 transactions is able to successfully being undetected. Moreover, we also observe that the final trust values T_f assigned by our model are rather confident, since they are closer to the trust limits, i.e. 0 for malicious nodes and 1 for benevolent nodes, while the other two models compute a trust value of around 0.5, thus indicating uncertainty in the evaluation of the node.

The next set of simulations focus on the reaction of the models against BSA (solid lines) and WA (dotted lines), as shown in Figure 6. In the BSA case, the requester node receives high recommendation values concerning a malicious provider from *two* common malicious friends. To tackle this attack is important to understand how each model manages the recommendations received by the common friends: in our model, such recommendations are used only in the startup phase and their weight decreases with the number of transactions as the Dynamic Knowledge acquires more experience (see Eq. 2). Chen's and Nitti's algorithms share a similar approach: the indirect opinion has always a certain relevance

TABLE VII
PERFORMANCE COMPARISON OF THE THREE MODELS AGAINST THE ON-OFF ATTACK.

	Proposed			**Chen et al.**			**Nitti et al.**		
	# trans.	T_i	T_f	# trans.	T_i	T_f	# trans.	T_i	T_f
ON $\rightarrow$ OFF	3	0.99	0.09	4	0.81	0.49	5	0.82	0.45
OFF $\rightarrow$ ON	3	0.05	0.77	5	0.41	0.5	5	0.17	0.5

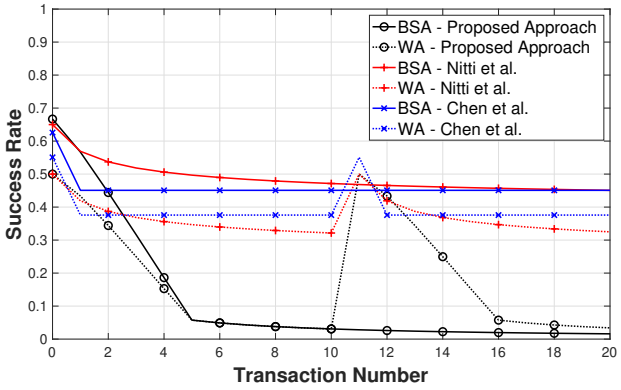


Fig. 6. Trust value of a malicious node that performs a Ballot Stuffing Attack and a Whitewashing Attack.

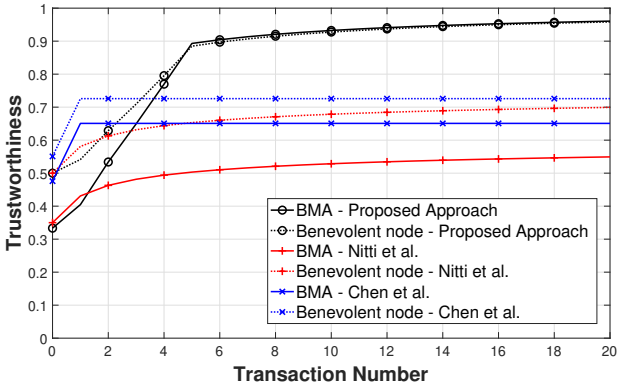


Fig. 7. Trust value of benevolent node with or without BMA.

in the trust score computation, however, its weight is different in the two models (0.15 in Chen's algorithm and 0.3 in Nitti's). From the Figure, we can see the trust value of the nodes implementing BSA and we can notice how our proposed model is almost non affected by the BSA (low trust values after only a few transactions), while the trust value computed by the other two models is definitely higher but still under the 0.5 threshold, thus marking the BSA nodes as malicious. In the case of WA, a malicious node with a bad reputation after 10 transaction leaves and re-joins the network to reset its trust to the default value. All the models are able to identify the node with few transactions and to label it again as malicious. However, Nitti's and Chen's algorithms are more robust to this attack, since the gain in the trust value of the WA node is lower w.r.t. our model.

The next set of experiments tests the BMA, where a malicious node provides false recommendations to decrease

TABLE VIII
PERCENTAGE OF POSITIVE TRANSACTIONS FOR AN OPPORTUNISTIC SERVICE ATTACK OVER 100 TRANSACTIONS

	% of Positive Transactions		
Trust Percentile	**Proposed**	**Nitti**	**Chen**
10%	100	86	82
20%	93	68	67
30%	86	61	57

the trust of benevolent nodes. We first test if this attack could lead a requester to choose a malevolent node over a benevolent one: all the models select the malevolent node *only once* and are then able to select the benevolent node. This is due to the higher importance given by the models to the direct experiences w.r.t. indirect recommendations. Moreover, the number of nodes implementing BMA does not affect this result. Then we investigate how the trust value changes in a scenario where a benevolent node is attacked by bad-mouthing nodes w.r.t. a benevolent node with no attackers. Figure 7 shows how our model is only affected by the BMA in the startup phase and it is then able to achieve the same trust values for the two benevolent nodes; the other two models present a lower trust value, which does not increase with the number of transactions, due to fixed parameters external to the requester experience, such as the centrality or the computation capabilities. Moreover, it clearly appears how BMA nodes can confuse the network, especially in Nitti's algorithm which gives a higher weight to the indirect opinion than Chen's.

The next set of simulations examines the OSA, where a node changes its behavior so that its trust value computed by the requester maintains an acceptable level. However, a node's goal is not to have a high trust value but rather to have a value higher than the other providers in order to be chosen (and then have a chance to behave maliciously). To test this attack, we consider only a service requester and a malicious service provider performing the attack. We suppose that the provider is perfectly aware of its trust reputation and act maliciously only when its trust value is among the 10%, 20% and 30% percentile of the most trustworthy nodes. Considering 100 transactions between the two nodes, Table VIII shows the percentage of positive transactions for the three models. As expected, a larger percentile enables the malicious node to perform more attacks, however the node could not be selected as a provider if there are other possible providers for the same service. If the malicious node wants to be sure to be selected and set a stringent percentile, the number of opportunities to behave maliciously reduces. However, our approach is able to compel the malicious node to perform the highest number

TABLE IX
AVERAGE OF TRUST FOR THE BENEVOLENT NODES WITH ERROR PERCENTAGE AND THE MALICIOUS NODES.

	Benevolent Node - Error percentage						Malicious Node			
	0%	**10%**	**20%**	**30%**	**40%**	**50%**	**ME/DA**	**WA**	**OOA**	**OSA**
Trust	0.99	0.86	0.79	0.72	0.57	0.28	0.02	0.05	0.12	0.88

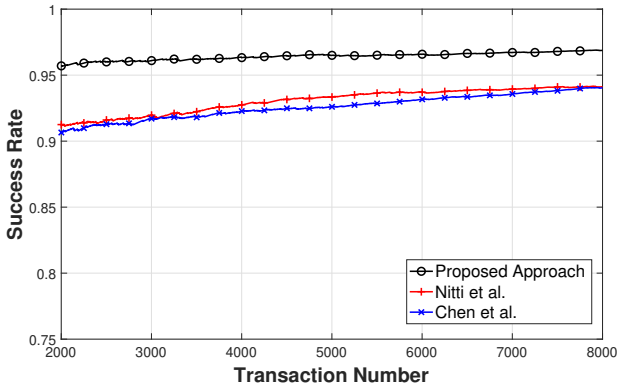


Fig. 8. Transaction success rate with all types of malicious attacks.

of positive transactions w.r.t. to the other two models, thus indicating the ability of the model to cope with this attack. In particular, if a malicious node wants to stay in the 10% percentile, it has to *always* perform benevolent.

We want now to show how the three models respond to a network with a mix of all the attacks analyzed. The result, in terms of transaction success rate, is shown in Figure 8, considering 5% of malicious nodes for each type of attack, for a total of 30% malicious nodes. Our model is able to converge faster and to outperform the other two with a success rate of over 95%. By analyzing which attacks had a higher impact, we see how simple attacks are better managed by Chen and Nitti's algorithms, however, as expected, they highly suffer smart attacks, such as OSA and OOA, which are not sufficiently tackled by them.

Finally, the last set of simulations is aimed to understand how our system reacts when benevolent nodes offer poor services due to errors related to several reasons. We then consider a requester which interacts with benevolent nodes which have a different probability to respond with an incorrect service due to some kind of error. For each value of the error percentage, we simulate 100 transactions between the nodes and mediate the results over 100 runs. Table IX shows the resulting trust values for different error rates of the benevolent nodes and compare them with the trust values of malicious nodes, without considering the attacks on the recommendation, i.e. BSA and BMA. Due to the subjective approach of our model, DA performs similar to ME, since, if it is connected to the requester by a weak link, it will always provide false services; nodes implementing WA have a slightly higher trust value, since they can reset their trust to the default value. As expected, the results show how increasing the error rate, the average trust of benevolent nodes decreases. However, even for nodes with a 50% error rate, their trust is still higher than nodes implementing OOA, which has a similar behaviour, i.e. 50% benevolent transactions and 50% malicious transactions: this is due to the Perseverance Score, which evaluates negatively the consecutive bad services of the OOA. Only a node implementing OSA is able to maintain a high level of trust. In this set of simulations, we consider that an OSA node acts maliciously only when its trust value is among the 20% percentile of the most trustworthy nodes. As seen in Table VIII, this means that the node will have more than 90% of trustworthy transactions, and thus can be considered as a node that offers bad services 10% of the time.

V. CONCLUSIONS

In this paper, we have analyzed the possible types of attacks that can be implemented by nodes to disrupt an IoT system. We then have proposed a trust management model based on a Machine Learning algorithm for a Social IoT scenario. The proposed solution is also applicable to general IoT scenarios, however, information regarding the type of friendship between two nodes is able to reduce the uncertainty in the selection of a trustworthy provider and provide better performance. The proposed model has been tested against all the different types of attacks, except for the SA and the SPA, which are avoided by default. Experiments have shown that our model is able to overcome all the possible attacks. Furthermore, we compare our algorithm with two well-acknowledged state-of-the-art models: simulations show that even if our algorithm show slightly worse performance when under attack by simple mechanisms, such as ME, it is able to outperform the other two models when considering a network with a mix of different types of attack.

ACKNOWLEDGEMENTS

This work was supported by Italian Ministry of University and Research (MIUR), within the PON R&I 2014-2020 framework (Project AIM (Attrazione e Mobilità Internazionale).

REFERENCES

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[2] D. Gambetta *et al.*, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 13, pp. 213–237, 2000.

[3] K. Li, L. Tian, W. Li, G. Luo, and Z. Cai, "Incorporating social interaction into three-party game towards privacy protection in iot," *Computer Networks*, vol. 150, pp. 90–101, 2019.

[4] W. Meng, K.-K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards bayesian-based trust management for insider attacks in healthcare software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 761–773, 2018.

[5] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[6] H. Xia, C.-q. Hu, F. Xiao, X.-g. Cheng, and Z.-k. Pan, "An efficient social-like semantic-aware service discovery mechanism for large-scale internet of things," *Computer Networks*, vol. 152, pp. 210–220, 2019.

[7] M. A. Azad, C. Perera, and M. Barhamgi, "Privacy-preserving crowd-sensed trust aggregation in the user-centeric internet of people networks," *ACM Transactions on Cyber-Physical Systems*, 2020.

[8] R. Chen, F. Bao, and J. Guo, "Trust-based service management for social internet of things systems," *IEEE transactions on dependable and secure computing*, vol. 13, no. 6, pp. 684–696, 2015.

[9] K. Zhao and L. Pan, "A machine learning based trust evaluation framework for online social networks," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2014, pp. 69–74.

[10] A. M. Kowshalya and M. Valarmathi, "Trust management for reliable decision making among social objects in the social internet of things," *IET Networks*, vol. 6, no. 4, pp. 75–80, 2017.

[11] B. Jafarian, N. Yazdani, and M. S. Haghighi, "Discriminative-aware trust management for social internet of things," *Computer Networks*, p. 107254, 2020.

[12] R. Chen, J. Guo, D.-C. Wang, J. J. Tsai, H. Al-Hamadi, and I. You, "Trust-based service management for mobile cloud iot systems," *IEEE transactions on network and service management*, vol. 16, no. 1, pp. 246–263, 2018.

[13] B. Gong, Y. Zhang, and Y. Wang, "A remote attestation mechanism for the sensing layer nodes of the internet of things," *Future Generation Computer Systems*, vol. 78, pp. 867–886, 2018.

[14] P. De Meo, F. Messina, M. N. Postorino, D. Rosaci, and G. M. Sarné, "A reputation framework to share resources into iot-based environments," in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2017, pp. 513–518.

[15] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 716–723, 2017.

[16] H. Al-Hamadi and R. Chen, "Trust-based decision making for health iot systems," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1408–1419, 2017.

[17] B. Li, L. Liao, H. Leung, and R. Song, "Phat: A preference and honesty aware trust model for web services," *IEEE Transactions on network and service management*, vol. 11, no. 3, pp. 363–375, 2014.

[18] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 5, pp. 1253–1266, 2014.

[19] Z. Chen, R. Ling, C.-M. Huang, and X. Zhu, "A scheme of access service recommendation for the social internet of things," *International Journal of Communication Systems*, vol. 29, no. 4, pp. 694–706, 2016.

[20] N. B. Truong, H. Lee, B. Askwith, and G. M. Lee, "Toward a trust evaluation mechanism in the social internet of things," *Sensors*, vol. 17, no. 6, p. 1346, 2017.

[21] D. Airehrour, J. A. Gutierrez, and S. K. Ray, "Sectrust-rpl: A secure trust-aware rpl routing protocol for internet of things," *Future Generation Computer Systems*, vol. 93, pp. 860–876, 2019.

[22] X. Fan, L. Liu, R. Zhang, Q. Jing, and J. Bi, "Decentralized trust management: Risk analysis and trust aggregation," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–33, 2020.

[23] A. Altaf, H. Abbas, F. Iqbal, and A. Derhab, "Trust models of internet of smart things: A survey, open issues, and future directions," *Journal of Network and Computer Applications*, vol. 137, pp. 93–111, 2019.

[24] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov, "Decentralized self-enforcing trust management system for social internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2690–2703, 2020.

[25] D. Wang, T. Muller, Y. Liu, and J. Zhang, "Towards robust and effective trust management for security: A survey," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2014, pp. 511–518.

[26] J. Caminha, A. Perkusich, and M. Perkusich, "A smart trust management method to detect on-off attacks in the internet of things," *Security and Communication Networks*, vol. 2018, 2018.

[27] A. Jøsang and J. Golbeck, "Challenges for robust trust and reputation systems," in *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*, vol. 5, no. 9. Citeseer, 2009.

[28] M. Rashmi and C. V. Raj, "A review on trust models of social internet of things," in *Emerging Research in Electronics, Computer Science and Technology*. Springer, 2019, pp. 203–209.

[29] J. Guo, R. Chen, and J. J. Tsai, "A survey of trust computation models for service management in internet of things systems," *Computer Communications*, vol. 97, pp. 1–14, 2017.

[30] R. Chen and J. Guo, "Dynamic hierarchical trust management of mobile groups and its application to misbehaving node detection," in *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*. IEEE, 2014, pp. 49–56.

[31] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for vanets: a statistical approach to rogue node detection," *IEEE transactions on vehicular technology*, vol. 65, no. 8, pp. 6703–6714, 2015.

[32] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, 2014.

[33] Y. Ma and G. Guo, *Support vector machines applications*. Springer, 2014, vol. 649.

[34] J. Xu, C. Xu, B. Zou, Y. Y. Tang, J. Peng, and X. You, "New incremental learning algorithm with support vector machines," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2230–2241, 2018.

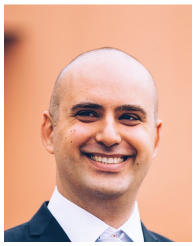
[35] C. Marche, L. Atzori, and M. Nitti, "A dataset for performance analysis of the social internet of things," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–5.

[36] C. Marche, L. Atzori, V. Pilloni, and M. Nitti, "How to exploit the social internet of things: Query generation model and device profiles' dataset," *Computer Networks*, p. 107248, 2020.

[37] S. Lee and C.-H. Jun, "Fast incremental learning of logistic model tree using least angle regression," *Expert Systems with Applications*, vol. 97, pp. 137–145, 2018.

[38] P. Reiner and B. M. Wilamowski, "Efficient incremental construction of rbf networks using quasi-gradient method," *Neurocomputing*, vol. 150, pp. 349–356, 2015.

Claudio Marche received the M.Sc. degree in telecommunication engineering with full marks in 2018 from the University of Cagliari. Since graduation, he has been working as Researcher in the Department of Electrical and Electronic Engineering at the University of Cagliari, in the MCLab research group. He is currently a Ph.D. student in Electronic and Computer Engineering at the University of Cagliari. His current research interests include Internet of Things (IoT), Social Internet of Things (SIoT) and Trustworthiness for IoT.



Michele Nitti is an Assistant Professor at the University of Cagliari, Italy since 2015. In 2013, he has been a visited student at the Department of Management, Technology and Economics at ETH Zurich, Switzerland. He served as a technical program chair for various international conferences (IEEE BMSB 2017, IEEE IoT V&T Summit 2020) and workshops (IEEE ICCCS 2019, IEEE GIoTS 2020). Currently, he is a member of the editorial board for the IEEE IoT Journal, Elsevier Computer Networks and MDPI IoT and co-founder of an academic spin-off (GreenShare s.r.l.) which works in the mobility sector. His main research interests are in architecture and services for the Internet of Things (IoT), particularly in the creation of a network infrastructure to allow the objects to organize themselves according to a social structure (Social Internet of Things - SIoT).