A Simulation Software for the Evaluation of Vulnerabilities in Reputation Management Systems

V. Agate, A. De Paola, G. Lo Re, M. Morana

ACM Transactions on Computer Systems (TOCS)
DOI: 10.1145/3458510

Publisher: ACM

A Simulation Software for the Evaluation of Vulnerabilities in Reputation Management Systems

VINCENZO AGATE, ALESSANDRA DE PAOLA, GIUSEPPE LO RE, and MARCO MORANA,
University of Palermo, Italy

Multi-agent distributed systems are characterized by autonomous entities that interact with each other to provide, and/or request, different kind of services. In several contexts, especially when a reward is offered according to the quality of service, individual agents (or coordinated groups) may act in a selfish way. In order to prevent such behaviours, distributed Reputation Management Systems (RMSs) provide every agent with the capability of computing the reputation of the others according to direct past interactions, as well as indirect opinions reported by their neighborhood. This last point introduces a weakness on gossiped information that makes RMSs vulnerable to malicious agents intent on disseminating false reputation values. Given the variety of application scenarios in which RMSs can be adopted, as well as the multitude of behaviours that agents can implement, designers need RMS evaluation tools that allows to predict the robustness of the system to security attacks, before its actual deployment. To this aim, we present a simulation software for the vulnerability evaluation of RMSs, and illustrate three case studies in which this tool was effectively used to model and assess state-of-the-art RMSs.

CCS Concepts: • **Networks** → **Network simulations**; • **Computing methodologies** → **Distributed algorithms**; • **Security and privacy** → *Distributed systems security*;

Additional Key Words and Phrases: Agent-based simulation, Multi-agent systems, Distributed Reputation Management Systems

ACM Reference Format:
Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. 2021. A Simulation Software for the Evaluation of Vulnerabilities in Reputation Management Systems. *ACM Trans. Comput. Syst.* , (June 2021), 29 pages. https://doi.org/

1 INTRODUCTION

Many distributed applications rely on software agents that repeatedly interact in order to achieve complex goals. In trading and e-commerce frameworks [52], for instance, agents allow to automate virtual business processes between different users and/or institutions; agents support peer-to-peer applications [39] by autonomously exchanging resources and services, so enhancing systems' scalability [56]; intelligent software agents can be exploited by collaborative intrusion detection systems to perform distributed security monitoring [55]; finally, social networks [7] and crowdsensing systems [59] can be easily modeled as multi-agent systems in which different entities are connected to each others.

The previous enumeration, although not exhaustive, makes it clear how the reliability of these applications depends on the correct behaviour of the agents. Unfortunately, the distributed nature of these systems, and the consequent lack of a central point of control, make it difficult to discover agents that adopt selfish behaviours in order to maximize their own utility. This aspect is even

This article is based on the work presented in the 22nd IEEE/ACM DS-RT Symposium [3].

Authors' address: Vincenzo Agate, vincenzo.agate@unipa.it; Alessandra De Paola, alessandra.depaola@unipa.it; Giuseppe Lo Re, giuseppe.lore@unipa.it; Marco Morana, marco.morana@unipa.it,
University of Palermo, Palermo, 90128, Italy.

2021. 0734-2071/2021/6-ART $15.00
https://doi.org/

more critical when the behaviour of a few agents affects the utility perceived by cooperative agents, so impairing the whole community. For this reason, several distributed applications rely on Reputation Management Systems (RMSs) to estimate the future behaviour of unknown agents before establishing actual interactions [22].

In distributed RMSs, every agent is able to estimate the reputation of the others. Generally, this is achieved by exploiting two different types of information, namely i) the direct experience resulting from previous interactions between the agent and its neighborhood, and ii) the indirect feedbacks reported by the neighbors about their interactions with other agents in the network.

Such a distributed approach avoids single points of failure and represents a scalable solution avoiding a potential performance bottleneck. Nevertheless, designing and evaluating a distributed RMS is far more complex than the same in a centralized scenario. For instance, since the reputation aggregation model is tightly dependent on the distributed protocol used to spread information over the agent network, it is often hard to preventively assess how these two components simultaneously influence each other.

Some distributed RMSs are based on a sound mathematical formulation that allows to theoretically evaluate them [30]; however, mathematical analysis is usually intended to assess ad-hoc case studies and do not provide a general formalism to evaluate a wide range of RMSs [20].

Other works have proposed solutions aimed to evaluate a RMS by simulating the interactions between the agents. This category of tools can be extremely useful as long as it allows to model all aspects that affect the behaviour of the RMS. However, even though several simulators have successfully addressed individual challenges, to the best of our knowledge there are no comprehensive frameworks which are able to deal with all the characteristics of a distributed RMS, regardless of the application scenario.

The simulation platform proposed in this paper allows researchers, on the one hand, to model a distributed environment where several agents interact, and on the other one to define the specific features of the RMS, the behavior of each agent, as well as the set of security attacks to be simulated. High-level interfaces allow disregarding some low level details (e.g., implementing the agent communications, driving the simulations), so letting the designer focus on more relevant tasks, such as defining new reputation algorithms, or selecting the specific features to produce the desired robustness. Moreover, an automatic assessment module permits to compute quantitative metrics that summarize the level of vulnerability of the RMS under analysis. These results can be immediately used to show the effects of different design choices on the RMS's performance.

With respect to other works presented in the literature, the simulation platform proposed here is characterized by the following desirable characteristics, partially identified in [33]:

- the inclusion of an abstract RMS model, that can be easily implemented to represent a specific system, and that maintains the independence from specific application scenarios;
- the inclusion of formal models for most common security attacks against RMSs;
- the capability of evaluating the effects of the decentralized nature of the RMS under evaluation, by comparing the achieved reputation estimation with reputation values obtainable by a centralized RMS which knows the actual outcome of all transactions;
- the availability of a set of well-defined metrics, capable of evaluating the RMS's accuracy and its vulnerability to security attacks, as well as the utility perceived by the agents while varying the policies of the designed RMS;
- a high flexibility in defining the set of security attacks, the aspects of the RMS to be analyzed, and the simulation scenarios;
- the possibility of performing large-scale simulations.

This paper extends our previous work published in [3], by providing a solid formulation of the RMS components, a formal model of different security attacks, and a wider set of evaluation metrics.

The remainder of the paper is organized as follows. The first part of Section 2 introduces the general RMS model we defined in order to represent many of the RMSs presented in the literature; then, the Section describes the most common attacks on the security of such systems. Section 3 provides a description of the agent-based simulation framework, focusing on the models adopted to define the service exchange policies, the reputation algorithms, and the agents' behaviours. Section 4 presents distinct sets of evaluation metrics aimed to measure the RMS's *accuracy* and *vulnerability*, as well as the agents' *utility*. Section 5 describes how the framework was used to model three RMSs, while Section 6 presents the evaluation of their performances when dealing with security attacks. Section 7 reviews the common approaches adopoted in the literature to evaluate the robustness of RMSs. Conclusions follow in Section 8.

2 REPUTATION MANAGEMENT SYSTEMS

Most of the reputation management systems presented in the literature exhibit similar characteristics: they exploit feedback coming from the agents (about other agents) in order to estimate the reputation of every participant of the network. RMSs may be useful in several scenarios where reputation values can be exploited to support agent's decision processes, regardless of whether agents are humans or software applications.

In the following, we present a general RMS model that can be adopted to represents many of the RMSs presented in the literature, and discuss relevant security attacks which can influence their performance.

2.1 RMS Model

In a generic model, each agent can act both as service consumer and/or provider. In the latter case, the agents' behaviour is unknown in advance to other agents; thus, its estimation is one of the main goal of the RMS.

Let $V(t)$ be the set of agents v_i taking part to the RMS, at time t. Without loss of generality, we can assume that the behaviour of the provider v_i can be derived from its level of cooperativeness, C_i, which represents the quantity and quality of the provided resources with a range of values that depends on the specific distributed application. In a distributed peer-to-peer system, for instance, a value $C_i = 0.8$ in the range $[0, 1]$ would indicate that the provider v_i satisfies on average 80% of the received file requests. It is worth noticing that, in a realistic scenario, the agents' behaviour may vary over time, thus the level of cooperativeness as to be indicated as $C_i(t)$.

The reputation of an agent, that should approximate its level of cooperativeness, is computed by the RMS according to available information, the most important of which are *feedbacks.* In general terms, a feedback $f_{ij}(t)$, generated by the consumer agent v_i about the provider agent v_j, is a tuple of values related to the outcome of the interaction between the provider and the consumer. These may include, for instance, a boolean value indicating whether the service request has been accepted, a quantitative value related to the quality of the obtained service, as well as other information as described in [20].

The RMS's architecture determines how feedbacks are exchanged between agents [28]. In centralized systems, a central server aggregates feedbacks to build a global and unique reputation value, i.e., $R_j^c(t)$, associated with the provider agent v_j. The centralized information fusion function F_{fusion}^c can be expressed by the following equation:

$$R_j^c(t) = F_{fusion}^c\left(\mathbf{f_{*j}}\right), \tag{1}$$

where, $\mathbf{f_{*j}} = \left\{f_{ij}(t'), \forall t' \in [0;t], \forall v_i \in V(t) : v_i \neq v_j\right\}$ is the set of feedbacks $f_{ij}(t')$ sent by each consumer $v_i \in V(t)$ about the provider $v_j \in V(t)$, with $v_i \neq v_j$, for every timestep $t' \in [0;t]$.

In a distributed RMS, agents other than generating feedbacks after each interaction, are involved in the whole process of reputation evaluation, without any central coordination activity. In these systems, information flows among agents according to the topology of an overlay *reputation network*, whose nodes correspond to the agents and each edge $< v_i, v_j >$ indicates that agents v_i and v_j know each other. It is worth noticing that the reputation network, represented by the set of edges $E(t)$, may vary over time. A single agent v_i can exploit direct feedbacks in order to compute a *local reputation* value $l_{ij}(t)$ of the agent v_j:

$$l_{ij}(t) = F_{local}\left(\mathbf{f_{ij}}\right), \tag{2}$$

where F_{local} is the *local reputation* function that considers as input the whole history of direct feedbacks about agent v_j, i.e., $\mathbf{f_{ij}} = \left\{f_{ij}(t'), \forall t' \in [0;t]\right\}$, collected up to the current timestep.

Furthermore, v_i exploits both direct feedbacks and messages received from other agents in order to build its own estimate of the reputation of v_j, i.e., $r_{ij}(t)$. Message exchanges occur according to a distributed *gossip protocol* which determines the information sharing between agents.

According to the *gossip protocol* adopted by the distributed RMS, at each time step t, the agent v_i may receive a message $\mathcal{M}_{k\to i}(t)$ from each neighbor v_k in the reputation network. Such a message is generated through the function F_{gossip}, which can consider either the whole history of direct feedbacks, or all received gossip information, or both of them. This can be specified by the following equation:

$$\mathcal{M}_{i\to k}(t) = F_{gossip}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right), \tag{3}$$

where $\mathcal{M}_i = \{\mathcal{M}_{k\to i}(t'), \forall t' \in [0;t], \forall v_k \in V(t') : \exists(v_i, v_k) \in E(t')\}$.

Messages collected from other agents, together with direct feedbacks, allow v_i to estimate the reputation of v_j through the *information fusion* function:

$$r_{ij}(t) = F_{fusion}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right). \tag{4}$$

It is worth noticing that many RMSs presented in the literature do not consider the whole history of feedbacks and messages at each time step, since this would require too much memory and computational effort. A common approach, for instance, is the adoption of a Markov model, according to which new reputation value depends on its previous value together with latest local reputation and gossip messages:

$$r_{ij}(t) = F_{fusion}\left(r_{ij}(t-1), l_{ij}(t), \mathcal{M}_i(t)\right), \tag{5}$$

where $\mathcal{M}_i(t) = \{\mathcal{M}_{k\to i}(t), \forall v_k \in V(t) : \exists(v_i, v_k) \in E(t)\}$, is the subset of messages received during the current timestep t. Other approaches, not based on the Markov assumption, are also described in the literature, e.g., the one discussed in [41]. In these cases, the information fusion process can be modeled through the most general equation 4.

Reputation values estimated through equations 4 or 5 can be exploited as basis of an *incentive mechanism* whose goal is to drive agents to act cooperatively. Incentives can be defined by assigning a penalty to agents with bad behaviour, or a reward to ones with good behaviour [43]. The effectiveness of the incentive mechanisms depends on the assumption that agents are individually rational, that is they choose the behaviour that maximizes their utility, to the best of their knowledge.

Although the definition of the utility function strictly depends on the application scenario, here we describe some of the most frequently adopted solutions:

- **Utility as image.** In some application scenarios, such as blog hosting sites or content-centric frameworks, the main goal of an agent is to exhibit a high reputation value in order to acquire high visibility in the community [43]. In such a case, the utility perceived by the agent v_i can be modeled as:

$$u_i(t) = F_{utility\_rep}(\mathbf{r_{*i}}(t)), \tag{6}$$

  where $\mathbf{r_{*i}}(t)$ is a vector containing the opinions held by other agents about the agent v_i at time t, i.e., $\mathbf{r_{*i}}(t) = \left\{r_{ji}(t), \forall v_j \in V(t)\right\}$, and $F_{utility\_rep}(\cdot)$ is a non-negative, concave, and increasing function. In such a scenario, RMSs may not include and explicit incentive mechanism, since a reputation loss would be the penalty for bad behaviors; as a consequence, rational agents will behave so as to maximize their own reputation.
- **Utility as amount of provided services.** When distributed systems are based on service-exchange models where some agents act as providers and other as consumers, the main goal of providers is generally to maximize their reward, that is to provide as much services as possible. Thus, the agent's utility can be modeled as:

$$u_i(t) = F_{utility\_prv}(\mathbf{prv_i}(\mathbf{t})), \tag{7}$$

  where $\mathbf{prv_i}(\mathbf{t})$ is the amount of services provided by agent v_i till timestep t, and $F_{utility\_prv}(\cdot)$ is a non-negative, concave, and increasing function. In this case, agents' utilities do not directly depend on their reputation; nevertheless, obtaining a good reputation so as to have more chances to be chosen as service provider is the implicit incentive that drives the agents to be cooperative.
- **Utility as balance between provided and received services.** When agents at the same time act both as providers and consumers, such as in peer-to-peer applications, the utility function could be more complex since it should take into account the benefit of obtaining services, but also the possible cost of providing them [47]:

$$u_i(t) = F_{utility\_p2p}(\mathbf{prv_i}(\mathbf{t}), \mathbf{rcv_i}(\mathbf{t})), \tag{8}$$

  where $\mathbf{prv_i}(\mathbf{t})$ and $\mathbf{rcv_i}(\mathbf{t})$ are, respectively, the number of services provided and received by agent v_i until the timestep t. Here, the incentive mechanism has to encourage agents to provide more services; thus, it generally imposes to provide few services to agents with low reputation values.

The previous classification underlines that in many of current distributed applications, it is necessary the introduction of opportune incentive mechanisms in order to create a strict dependence between the reputation of an agent and what it perceives as its own utility. In general terms, the incentive mechanism can be seen as a mapping function, deterministic or not, that drives each agent in the selection of the action to perform.

When the utility is defined according to equation 7, the decision function allows a consumer to select a provider; for instance, it could suggest to choose the provider with the highest reputation, or a random one with a probability that is proportional to its reputation. In scenarios in which the utility function is modeled according to equation 8, the decision function is exploited by providers to decided whether provide or not a service, for instance, the agent can establish to provide resources proportionally to the reputation of its peers.

Therefore, the decision function evaluated by the agent v_i accepts as input its estimation of the reputation of other agents, i.e., $\mathbf{r_{i*}}(t) = \left\{r_{ij}(t), \forall v_j \in V(t)\right\}$, and its own level of cooperativeness, i.e., $C_i(t)$:

$$a_i(t) = F_{action}(\mathbf{r_{i*}}(t), C_i(t)), \tag{9}$$

The decision function can be also combined with a *detection algorithm* capable of identifying malicious agents according to a given policy. In its simplest form, the detection algorithm could consider malicious the agents with low reputation values. The most common approach is instead to consider a wider range of decisions that directly depend on the reputation values.

2.2 Security Attacks on RMSs

In this subsection, we introduce the most common attacks capable of undermining the correct functioning of distributed RMSs. Since the proper operation of these systems depends on the choice of the agents to cooperate in the reputation estimation, fake feedbacks represent one of the most serious threat.

Several type of security attacks that exploit the distributed nature of RMSs and their dependence on agent feedbacks are reported in the literature [23, 51]. Malicious agents performing these attacks are *insiders*, i.e., authorized agents of the system that legitimately participate in reputation evaluation. According to this assumption, in this paper we do not consider security issues related to attacks performed by outsiders, such as threats to authentication, integrity and confidentiality. Other approaches, such as the one proposed in [27], can be adopted in order to guarantee a correct membership management.

A classification of security attacks performed by insiders can be made on the basis of their goals, e.g., *promoting* and *slandering*, or according to the technique adopted to complete the attack, such as *oscillating* behaviours or the creation of more (*sybil*) identities. In *promoting* and *slandering* attacks, malicious agents spread fake information over the network in order to alter the evaluation of other agents' reputation. This kind of attacks may be performed by an individual agent or by a *coalition* of multiple agents. In *oscillation* attacks, instead, malicious agents modify their degree of cooperativeness in order to elude a truthful evaluation of their reputation. A *sybil* attack, conversely, aims to deceive the RMS detection algorithm by creating new identities that can be exploited to whitewash a past bad reputation or to strengthen a coalition attack.

2.2.1 Promoting Attack.

Promoting attacks [36] aim to artificially increase the reputation of a target agent in order to allow it to obtain illegitimate benefits, or to hide its selfish behaviour. For instance, *promoting* attacks are very common in e-commerce applications, due to their capability of providing an unjustified advantage over competitors.

Such type of attack is performed by disseminating fake positive information through the gossip protocol. Generally, the promoting agent, v_j, is different from the target agent, v_i, which actually takes advantage of the attack.

Ideally, attackers should send to their neighborhoods those messages that maximize the reputation of the target agent. However, malicious agents cannot achieve the best optimization, since they do not know the policy and parameters used by the neighborhood agent to perform the reputation fusion (see equation 4). As a consequence, typically, the attackers diffuse into the neighborhood most favorable information pro target agent.

The RMS component able to cope with *promoting attacks* is the information fusion function, F_{fusion}, which determines the balance among past history, direct experience, and gossiped information. In order to increase the RMS resistance to fake information, F_{fusion} could also weight information obtained, e.g., on the basis of the reputation of gossiper agents.

2.2.2 Slandering Attack.

Slandering attacks [8] aim to decrease the reputation of a *victim* agent by exploiting the gossip protocol to disseminate fake information over the reputation network. Just as *promoting*, even this type of attack is very common in e-commerce scenarios, where malicious agents try to obtain indirect advantages by spoiling their competitors.

Unlike *promoting* attacks, here slanders should send to their neighbors those messages capable of minimizing the target's reputation. However, since such an optimization would require knowing the reputation estimated by neighbors, the attack is actually performed by flooding the worst information regarding the victim agent.

Even in this case, the information fusion function is the main component that can let the system resist through an opportune balancing of direct experience and whispered information.

2.2.3 Oscillation Attack. This kind of attacks [50], also known as *on-off* [34] or *traitor* attacks [39], are performed by malicious agents that alternate faithful/unfaithful behaviours in order to mislead the estimation of their own reputation. To be more specific, a malicious node maintains a loyal behaviour until it has obtained a satisfactory reputation, then adopts a selfish behaviour in order to misuse system resources. For the attack to be effective, malicious agents generally alternate cooperative to antisocial behaviours for a limited amount of time, thus exhibiting an oscillating reputation.

Oscillation attacks are frequently reported in applications characterized by a non negligible cost for providing services, such as peer-to-peer applications [48], and multi-hop wireless networks [34]. In these scenarios, the selfish behaviour corresponds to provide low-quality (and low-cost) services or even no service.

The attack pattern, represented as the duration of cooperative/non-cooperative time intervals and the levels of cooperativeness, should be selected according to the specific characteristics of the targeted RMS.

RMSs more vulnerable to this type of attacks are those in which past history weights more than the recent experience, or, also, RMSs characterized by observation windows that are too short as compared to the oscillation period.

2.2.4 Coalition Attacks. When the goal of the attack is to spread fake information over the reputation network, e.g., *promoting* and *slandering*, the attackers usually need the support of other agents. In *slandering attacks*, for instance, a coordinated plan which involves a *coalition* of malicious agents allows to lower the victim's reputation more significantly, i.e., the greater the number of attackers the stronger the effect of the attack. In other cases, e.g., in *promoting attacks*, the collusion with other agents is necessary to elude the basic rule of any RMS, which prohibits an agent from spreading feedbacks and information about itself.

2.2.5 Sybil Attack. In many distributed scenarios, user registration is a simple procedure that does not involve specific security checks. Attackers may easily register new users in order to obtain multiple identities that can be exploited to perform a *Sybil* attack [17, 23].

During a *whitewashing* attack [18], for instance, a malicious agent which has reached a bad reputation as consequence of a selfish behaviour, creates sybil identities to clean up its history and rejoin the network with a restored, default, reputation value. RMSs that are more vulnerable to such attacks are those which assign an optimistic default reputation value and exploit negative feedbacks to discover malicious behaviour. In these systems, the default reputation is quite similar to the long-term reputation obtained by cooperative agents; thus, for attackers it is more convenient to start again with new identities than act cooperatively for a long time. Accordingly, a greater resistance is expected by those RMSs that adopt a low initial reputation value r_0 and exploit positive feedbacks to rise their estimation.

Multiple identities created during a *sybil* attack can also be exploited to amplify the effect of coalition attacks. RMSs more vulnerable to this condition are those that accept feedbacks without verifying that the transactions have actually occurred. Such kind of attack could be afforded by

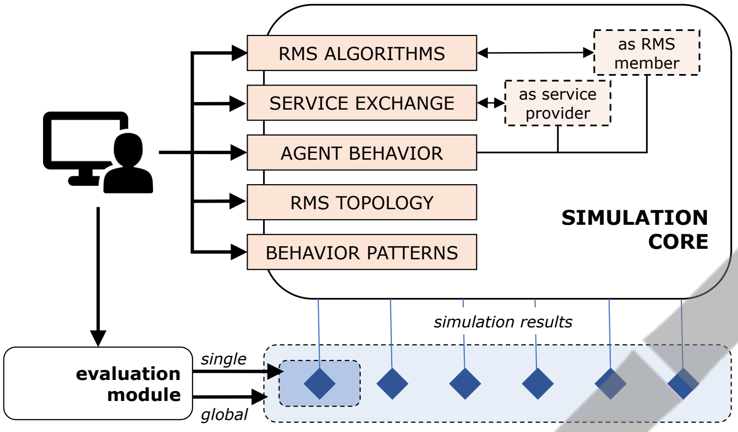


Fig. 1. Overview of the simulation framework.

discarding fake feedbacks coming from sybil accounts that have not participated in real service exchanges.

3 THE SIMULATION FRAMEWORK

The aim of the simulation platform is to easily map the formal model described in the previous section into a running RMS whose security can be assessed according to the behaviours of the simulated agents. To this end, the simulator provides a set of high-level interfaces that allow to quickly setup a simulation scenario, while neglecting low-level implementation details.

This *abstraction* is achieved by considering two distinct logic layers. The first layer allows to model the RMS and the scenario in which it operates; here, the RMS is seen as a distributed system in which the agents act in order to provide/obtain a service. The second one provides the low-level functionalities needed for driving the simulation; at this layer, agents correspond to system processes that communicate with each other by means of a message exchange paradigm.

The designers/developers can use the upmost layer to define from scratch their own distributed application and the adopted RMS, or can combine existing approaches starting from the classes provided within the simulation library.

The architecture of the simulator is shown in Fig. 1. The designers interact with the simulation engine by defining the algorithms of the RMS, the service exchange policies, as well as the agents' behaviours. This last feature is particularly relevant since it allows to define how the agents act both during the exchange of services and in all the general operating phases of the RMS. Moreover, as will be discussed later in this section, some default behaviours are provided which can be used individually, or combined to define new ones. Behaviour patterns, i.e., how the agents' behaviours change over time, and the topology of the distributed RMS, i.e., the set of neighbors each agent can interact with, are the last two parameters needed for starting a simulation.

The RMS performances can be analyzed globally, or as observed by a specific agent. In the first case, the designer can compare the RMS under evaluation with an ideal centralized RMS that is not affected by fake information and bias introduced by the gossip protocol; thus, evaluation can be based on an objective global error index. Moreover, global evaluation allows also to measure the RMS's vulnerability to a variety of security attacks, according to some well-defined metrics. Conversely, when a single agent v_i is selected, it is possible to observe specific aspects of the simulation, such as the trend over time of the reputation of v_i as estimated by its neighborhood, or the percentage of service requests satisfied by v_i.

3.1 Agent-based Model

The interactions that take place in the distributed system in which a RMS operates concern agents, as both providers and consumers of services, that act to obtain some reward. As a general rule, RMSs include incentive mechanisms which ensure that such a reward depends on the agent's behaviour.

In our model, the interactions happen according to the synchronous time-discrete model proposed in [37]; thus, during the simulation, every agent executes the sequence of steps defined by the developer, e.g., requiring/providing services, sending data to the neighborhood to support the distributed reputation estimation. Single agents are defined by inheriting and implementing an abstract class included in the simulation library. At each round of simulation, the platform cyclically manages the agent behaviour through a set of methods that will be described in the following of this section.

3.1.1 Service Exchange Model. Generally, agents are able to act simultaneously as both service producers and consumers. The service exchange model sets up the agent's behaviour in five distinct phases, namely the *service announcement*, *selection*, *request*, *reply*, and *rating*:

(1) Through the *service announcement protocol* each provider communicates the set of available services to a subset of consumers. For instance, an agent can notify its neighbours only, announce its services to the whole network in a single simulation step, or select a group of consumers reachable through a hop-by-hop path. The **`serviceAnnouncement`** method implements this protocol allowing to specify, as input variables, names and number of the **`services`** provided by the agent to the selected **`consumers`**.
(2) Consumers analyze **`services`** announced by the **`providers`**, and select some of the proposals according to the *service selection* policy. The **`serviceSelection`** method allows to specify the logic followed by a consumer agent to perform such a selection. For instance, a simple implementation of this method allows to choose the provider with the highest reputation, but such a policy can cause a lock on a few providers. Another common policy is to randomly select providers according to probabilities linearly or exponentially proportional to their reputation. Such a policy can balance among the exploitation of information about providers' past behaviours and the exploration of new providers.
(3) Once a provider has been selected, consumers send service requests using the **`serviceRequest`** method.
(4) Within a single simulation step, provider agents receive the requests and reply to each of them with variable degrees of cooperation; this behaviour is achieved through the **`serviceReply`** method. Decisions taken by this method are influenced by the agent's *behaviour as service provider*, which depends on its cooperativeness, i.e. $C_i(t)$.
(5) Finally, consumers give feedbacks on the completed transactions, e.g., 1 (success), or -1 if the service has not been provided. Ratings for all services received from a provider till the current time step are stored by each agent in **`txReport`** data-structure.

The *incentive mechanism* adopted by the RMS is generally related to the steps (2) or (4), depending on the agents' interaction model. When the distributed system is based on a service-exchange model in which agents act distinctly as providers or consumers, and the providers' utility depends on the amount of exchanged services, as modeled in equation 7, the incentive mechanism should reward providers with a high reputation during the *service selection* step (2).

Conversely, in application scenarios where agents act both as providers and consumers, and the utility function takes into account provided services (as a cost) and obtained services (as a reward), as modeled by equation 8, the incentive mechanism operates within the *service reply* method (4). For instance, a service reply policy could state that a service is provided only to consumers that have some reputation requirements; thus, rewarding trustworthy agents implicitly discourage selfish behaviours.

3.1.2 *Reputation Algorithms.*

One of the main features of the proposed simulation platform is the possibility of specifying the RMS algorithm, which basically defines how to diffuse agents' opinions over the network, and how to compute the reputations according to the available information. Reputation can be expressed as numbers, labels representing predefined values (e.g., *low*, *normal*, *high*), as well as custom objects summarizing multiple data.

Although the literature presents a great variety of RMSs, it is possible to identify four basic components that are represenative of most of them [1].

The first, i.e., the *local reputation evaluation*, implements the F_{local} function presented in equation 2 and exploits the feedbacks f_{ij} collected during past interactions between agents i and j. From a simulation point of view, the **`localReputation`** method allows the developer to implement any *local reputation* algorithm according to the rates provided in the last **`NT`** time steps and stored in **`txReport`**. The output of this method represents a first, rough, estimation of the reputation of other agents.

Two other components, namely the *gossip protocol* and the *information fusion* mechanism, allow to compute the reputation of agents that have never been seen before. The former consents to agents to exchange information about the reputation of their neighbours, while the latter is used to locally merge estimated and gossiped reputation.

The *gossip protocol* implements the abstract model F_{gossip} function (see equation 3); the method provided by the simulator, i.e., **`gossipProtocol`**, allows to define which **`data`** have to be shared with the agents specified in a **`destination`** list. A common choice, for instance, is to share recent reputation values, i.e., $r_{ij}(t)$, with neighbors only.

As regards the *information fusion* mechanism, which implements the F_{fusion} function from the model equation 4, the simulation platform provides the **`informationFusion`** method that allows an agent to estimate at each time step the reputation of the others by merging all the available information. Generally, this method takes as input the whole history of direct feedbacks and the received messages. Nevertheless, in some RMSs, the information fusion can be simplified in order to consider only the last received messages and the local reputation value, as expressed in equation 5. If no information is available, *information fusion* can be performed by exploiting a default reputation value.

The fourth component is the *incentive mechanism* operated by the RMS in order to discourage antisocial behaviours by rewarding collaborative agents. This last component is implemented by the service exchange methods, as described above.

3.1.3 *Agent behaviour.*

The simulation framework makes it possible for a designer to define the detailed behaviour of the agents with respect to their trustworthiness as members of the RMS, and their cooperativeness as service providers. The former reflects behaviours such as *honest*, *slander* (if neighbors are attacked with fake negative feedbacks), and *promoter* (in case of fake positive opinions). Similarly, service providers can be *cooperative*, *selfish*, or partially cooperative, with a degree that can be specified by the user.

Malicious behaviours *as RMS members* are obtained by altering the content of **`data`** sent through the **`gossipProtocol`** method; conversely, different behaviours *as service providers* can be defined by modifying the **`serviceReply`** method.

We present here some methods, already available for developers, which can be adopted in generic producers/consumers scenarios, where message exchanged through the gossip protocols contain the reputation values of providers. Nevertheless, these can be further combined and extended by new definitions with the aim to describe any type of attack.

The `spreadRep` method creates the messages through which a malicious agent can diffuse a fake reputation value for a `target` agent. If the announced value is lower than the true perceived reputation r_{ij}, such behaviour produces a slandering effect; conversely, the behaviour results in promoting the `target`.

The `oscillatingProvider` method allows to model a provider involved in an oscillating attack that can be directed against all consumers, or the subset specified in a targets' list. An honest behavior, associated with a predetermined cooperativeness degree value is maintained for some time steps; then, a malicious conduct continues for some other time steps, specified as input variables.

Finally, the `whileReputationBelow` and `whileReputationOver` methods allow the attacker to tune its behaviour *as service provider* according to a local estimate of its own reputation. Since these methods alter the cooperativeness degree, they are invoked before `serviceReply`. To be more specific, `whileReputationBelow` can be used to make an agent operate with a given cooperativeness deegree, as long as its reputation is under some threshold. This method, for instance, can be adopted when an attacker is trying to rising its reputation by acting cooperatively, until a safe reputation value is reached. Conversely, `whileReputationOver` makes the agent adopt a chosen cooperativeness degree, as long as its reputation is over a given threshold. This method is run when a malicious provider tries to abuse the system resources by acting selfishly, until an alarm reputation value is reached. In both cases, when the threshold value is reached, the agent adopts the default cooperativeness degree. By combining the two methods is possible to perform an *oscillation attack*, whose time intervals depends on the effect of agent's actions on its reputation.

More generally, the simulation framework includes also a set of behavioural patterns users can combine to describe how specific agents should act. While configuring a specific simulation scenario, it is possible to define how many agents should exhibit a single behaviour b, or an ensemble of n different behaviours, $b = [b_1, ..., b_n]$. When planning coalition attacks, for instance, a hostile agent might be interested in rising the reputation of its associate, while also reducing that of the victim. In such a case, the behaviour of the attacker A could be expressed as $b_A = [b_{promoter}(targets), b_{slander}(victims)]$. Similarly to simple attacks, even complex behaviours can be implemented by altering the reputation values sent through the `gossipProtocol()`. For instance, we could define a *promoter* behaviour from time step 0 to 50 for a set of `targets`, and a *slander* behaviour from step 10 to 60 for a set of `victims`:

```
behaviours.add("promoter", targets_list_p, 0,50);
behaviours.add("slander", victims_list_s, 10,60);
```

During the simulation, before the `gossipProtocol()` is run, it is possible to generate the data to be sent according to the *behaviour as RMS members* defined for the current time step:

```
data = behaviour_as_RMS_members(data,behaviours.getCurrentBehaviours());
gossipProtocol(data,destination);
```

In this example, `behaviour_as_RMS_members` would run the `spreadRep` method twice; first to implement *promoting*, then to carry out *slandering*.

All the parameters needed by these methods during the simulation are specified in two configuration files, processed at startup, that adopt a simple language to detail the characteristics of the reputation network and its agents.

The first file specifies the network topology, indicating for each node the list of its neighbours. Generally, the topology is dynamic, thus it is possible to specify the edges to be added to, or removed from, the network for each time step. The second configuration file contains the agents' behaviour parameters.

4 EVALUATION METRICS

The simulation framework produces as output a set of measures that allow the designers to evaluate the performance of the RMS under analysis, with respect both to its accuracy and its vulnerability to security attacks.

4.1 Accuracy Metrics

The accuracy of a RMS can be evaluated by measuring the error between the true cooperativeness of providers and their reputations as estimated by other agents.

The accuracy metrics proposed here can be adopted in RMSs characterized by absolute reputation indexes, that is where the reputation value assigned to a single agent is independent of the values assigned to other ones. On the contrary, RMSs that adopt relative reputation indexes perform a normalization among reputation values. As a consequence, two agents may estimate different reputation values for the same provider, due to their different knowledge about other agents.

The following definition of accuracy requires that reputation values and cooperativeness belong to the same domain and range. If this condition is not satisfied, a normalization step is needed before evaluating the accuracy.

Since there is not a unique value of reputation for each agent, we propose to evaluate, for each agent v_i, at each time step t, the error between its true cooperativeness, i.e., $C_i(t)$, and its *average reputation*, defined as:

$$r_i(t) = \frac{\sum\limits_{v_j \in H_i(t)} r_{ji}(t)}{\#\left\{H_i(t)\right\}}, \tag{10}$$

where $H_i(t)$ is the set of agents that hold an opinion about the agent v_i at time t, and $\#\left\{H_i(t)\right\}$ is the number of these agents. Consequently, the average error on a single agent v_i can be computed as:

$$e_i(t) = |C_i(t) - r_i(t)|. \tag{11}$$

Moreover, besides such absolute measure of error, the simulator allows to highlight whether, and to what extent, the error depends on the distributed gossip protocol. Such assessment is performed by comparing the error obtained by the RMS under analysis, with that achieved by an ideal RMS which adopts the same reputation model, but exploits true interaction evaluations rather than messages received through the gossip protocol.

Such a RMS is implemented by means of a *truth-holder* agent (see Fig. 2) that does not participate to the gossip protocol and collects true interaction outcomes in order to evaluate the ground-truth reputation $R_i^*(t)$ of each agent v_i.

It is worth noticing that it necessary that the reputation aggregation algorithm adopted by the *truth-holder* is the same adopted in the considered RMS model, but neglecting gossiped information; thus, it must be redefined by the designers in order to meet the behaviour of the RMS under analysis.

The *relative error* achieved by the RMS respect to the *truth-holder*, on a single agent, is defined as:

$$e_i^*(t) = |R_i^*(t) - r_i(t)|. \tag{12}$$

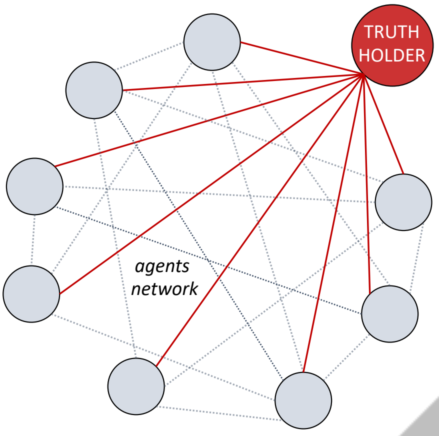


Fig. 2. The role of the *truth-holder* within the agents network.

The *average system error* is then computed by averaging the *relative error* over the whole set of agents:

$$e^*(t) = \frac{\sum\limits_{v_i \in V(t)} e_i^*(t)}{\#\{V(t)\}}. \tag{13}$$

4.2 Vulnerability Metrics

In order to measure the robustness of a RMS to different security attacks, the simulation platform provides the developers with a set of predefined metrics, while also giving the possibility to define new ones according to the specific simulation needs.

The definition of the vulnerability metrics is strictly connected to the specific *success conditions* of the various attacks. As regards the two simple attacks *as RMS member* described in Section 2.2, the *success conditions* can be defined by observing the long-term reputation of an agent, i.e., its stable reputation estimated after T_{max} time steps:

- a *promoting* attack succeeds if the long-term reputation of the target (a selfish agent) is greater than half of the maximum reputation value R_{max}, i.e., $R_{th} = R_{max}/2$;
- a *slandering* attack succeeds if the long-term reputation of the victim (a cooperative agent) is below this limit.

The vulnerability to attacks based on the diffusion of fake feedbacks, i.e., attacks *as RMS member*, can be evaluated through the following set of metrics, preliminarily outlined in [2]:

- *time-to-falsify (TF)*: the time required to achieve the attack success condition, normalized with respect to the simulation time threshold T_{max};
- *collusion-degree (CD)*: the percentage of malicious agents required for a *collusion* attack to succeed within a certain time, e.g., $T_{th} = T_{max}/2$.

The *collusion-degree* is evaluated by performing several simulations with different percentage of malicious agents, so as to find the minimum percentage required to achieve the above described condition. The same two metrics can be adopted to evaluate the vulnerability of the RMS to *sybil* attacks, aimed at making the impact of *promoting* and *slandering* more effective.

When dealing with attacks aimed at misusing the system resources for as long as possible, i.e., when an attacker exhibits a malicious behavior *as service provider*, the following metric can be adopted:

- *exploitation-time (ET)*: the normalized time during which a malicious agent misuses the system resources, before its reputation drops below an alarm threshold.

For instance, considering the selfish behaviour at the basis of *whitewashing*, if we assume that T^* time steps are required to bring the reputation below the threshold R_{th}, the exploitation-time can be defined as $ET(selfish) = T^*/T_{max}$. Thus, changing the threshold value R_{th} corresponds to define stricter or weaker success conditions. In the case of *oscillation* attacks, since cooperative and selfish behaviours are alternated for T_{coop} and $T_{selfish}$ time steps, the normalization required to evaluate the *exploitation time* considers the duration of the oscillation period only, rather then to the whole simulation duration. Thus ET is the greatest (normalized) value of $T_{selfish}$, which guarantees that reputation remains above the R_{th} threshold, i.e., $ET(oscillation) = T_{selfish}/(T_{coop} + T_{selfish})$. Even in this case, the simulation software performs different simulations by varying the value $T_{selfish}$, in order to find the one which satisfies the above described condition.

According to these metrics, the vulnerabilities indexes of a RMS exposed to *slandering* (*Sl*), *promoting* (*Pr*), *selfish* (Se), and *oscillation* (*Os*) attacks can be expressed as:

$$\begin{aligned}
VI_{Sl} &= [1 - TF(slandering)] \times [1 - CD(slandering)];\\
VI_{Pr} &= [1 - TF(promoting)] \times [1 - CD(promoting)];\\
VI_{Se} &= ET(selfish);\\
VI_{Os} &= ET(oscillation).
\end{aligned}$$

These indexes have a numeric score S, $0 \leq S \leq 10$, that corresponds to a qualitative rating as proposed by the Common Vulnerability Scoring System (CVSS) [15]: *none* (if $S < 0.1$), *low* ($0.1 \leq S < 4.0$), *medium* ($4.0 \leq S < 7.0$), *high* ($7.0 \leq S < 9.0$), *critical* ($9.0 \leq S \leq 10.0$).

The overall vulnerability index can be represented as a vector containing the list of the detected vulnerabilities, and the number of vulnerabilities evaluated as *high* or *critical*, i.e., $\#_{hc}$:

$$RMS\_vulnerability = \{[VI^*_{Sl}, VI^*_{Pr}, VI^*_{Se}, VI^*_{O}], \#_{hc}\},$$

where the VI^* values are obtained by applying a gamma correction, with $\gamma = 0.5$, and a scale factor $K = 10$:

$$VI^* = K \times VI^{\gamma}.$$

K and γ values have been experimentally determined in order to allow that the four vulnerabilities metrics would be comparable in a network composed of 100 agents with 6 neighbors for each agent on average. Clearly, users of the simulation software can modify these parameters in order to obtain a different scales for metrics.

4.3 Utility Metrics

The third group of metrics aims to evaluate the effect of different design choices and policies on the utility perceived by the agents. Our simulation software provides three basic utility metrics that can be further enriched by the developers according to the characteristics of specific application scenarios.

In service-exchange applications, such as in the e-commerce domain, the utility of a provider v_i depends on the number of provided services $\mathbf{prv_i}$, as modeled by equation 7. In order to obtain a comparative analysis for providers that operate in the same network, such a quantity should be normalized with respect to the average number of services provided by fully cooperative agents. Being V_{C^*} the set of agents that maintain the average cooperativeness C^*:

$$V_{C^*} = \left\{ v_i \in V(t), \forall t \in [0; T_{max}] : \underset{t \in [0;T_{max}]}{average} \{C_i(t)\} = C^* \right\}, \tag{14}$$

their average utility can be measured through the following metric:

$$U_{prv}(C^*) = \frac{\underset{v_i \in V_{C^*}}{average} \{\mathbf{prv_i}(T_{max})\}}{\underset{v_i \in V_{C_{max}}}{average} \{\mathbf{prv_i}(T_{max})\}}. \tag{15}$$

Providing developers with this average value allows to obtain a high-level estimation of the effect of the adopted policies on whole classes of agents.

In peer-to-peer applications, the utility may be measured by considering each agent separately as a consumer and as provider. In the first case, the utility of v_i at time T_{max} can be measured as the ratio of the obtained $\mathbf{rcv}_i(T_{max})$ and requested $\mathbf{req}_i(T_{max})$ services; the utility is measured by averaging among all the consumers, according to the following equation:

$$U_{p2p_c} = \underset{v_i \in V}{average} \left\{ \frac{\mathbf{rcv}_i(T_{max})}{\mathbf{req}_i(T_{max})} \right\}, \tag{16}$$

Conversely, the utility of agents seen as providers can be evaluated, according to equation 8, as the ratio of received and provided services:

$$U_{p2p_p} = \underset{v_i \in V}{average} \left\{ \frac{\mathbf{rcv}_i(T_{max})}{\mathbf{prv}_i(T_{max})} \right\}. \tag{17}$$

This last metric allows to measure also the fairness of the RMS, since a perfectly fair model should drive this utility value to 1.

5 CASE STUDIES

This section presents how the proposed simulation framework can be used to model three reputation management systems, namely a sample RMS, Beta Reputation [29], and Core Reputation [41].

5.1 Sample RMS

The first RMS chosen as case study is designed for peer-to-peer scenarios in which agents act both as consumers and providers of the same service, and the *service announcement protocol* specifies that agents send announcement messages to their neighborhood only.

In the considered model, reputation values $r_{ij}(t)$ are real numbers in the range $[0, 1]$, and the cooperativeness value $C_i \in [0, 1]$ represents the probability that the provider v_i successfully attends to a service request. Thus, a fully cooperative agent is characterized by $C_i = 1$, while a totally selfish one by $C_i = 0$.

The adopted *service selection* strategy specifies that a consumer asks for services to all the providers in its neighborhood. Such a naive approach is absolutely fair, so avoiding the possible bias introduced by more complex selection strategies.

The *service reply* method implements an *incentive mechanism* that provides random replies, with probabilities proportional to the reputation of the requiring agents, as described in [14]. If the agents behave also as service providers, such probabilities are further enforced according to their cooperativeness. Thus, at time t, v_i will positively reply to service requests coming from v_j, with a probability $p_{ij}(t)$ defined as:

$$p_{ij}(t) = r_{ij}(t) * C_i(t). \tag{18}$$

After each interaction, the consumer generates a feedback $f_{ij}(t) \in \{1, 0\}$ depending on whether the service has been provided or not.

The considered RMS model implements three common components, namely *local reputation* evaluation, *gossip protocol*, and *information fusion*.

The *local reputation* algorithm is inspired to [30]; for each provider, it computes the ratio of the number of successful transactions and sent requests, with respect to a sliding window of ΔT time steps:

$$l_{ij}(t) = F_{local}\left(\mathbf{f_{ij}}\right) = \frac{\sum\limits_{f_{ij} \in \mathbf{f_{ij}}(\Delta T)} f_{ij}}{\#\left\{\mathbf{f_{ij}}(\Delta T)\right\}}, \tag{19}$$

where $\mathbf{f_{ij}}(\Delta T)$ is the set of feedbacks collected during the last ΔT timesteps, i.e.,
$\mathbf{f}_{ij}(\Delta T) = \left\{f_{ij}(t'), \forall t' \in [t - \Delta T; t]\right\}$.

The *gossip protocol* assumes that each agent sends to its neighbors the last estimated reputation values of other agents it knows, as specified by the following equation:

$$\mathcal{M}_{i\to k}(t) = F_{gossip}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right) = F_{gossip}\left(\mathbf{r_{ij}}(t-1)\right) = \left[\left(j, r_{ij}(t-1)\right)\right]_{v_j \in O_i(t-1), v_j \neq v_k}, \tag{20}$$

where $\mathbf{r_{ij}}(t-1)$ are the reputation values of agents v_j as estimated by v_i, i.e.,
$\mathbf{r_{ij}}(t-1) = \left\{r_{ij}(t-1), \forall v_j \in V(t)\right\}$, and $O_i(t)$ is the set of agents about which v_i has an opinion at time t.

The *information fusion* technique is inspired to the reputation algorithm proposed in [16], which requires that gossiped information is weighted with reputation of the gossiper agents. Two parameters α and β specify, respectively, the weight of last direct experience with respect to past history, and the weight of received opinions on the overall reputation. Since this fusion algorithm is explicitly based on the Markov assumption, the information fusion method can be expressed according to equation 5:

$$\begin{aligned} r_{ij}(t) &= F_{fusion}\left(r_{ij}(t-1), l_{ij}(t), \mathcal{M}_i(t)\right) = \\ &= (1-\beta) * \left[\alpha * l_{ij}(t) + (1-\alpha) * r_{ij}(t-1)\right] + \beta * \frac{\sum\limits_{v_k \in N_i(t)} r_{ik}(t-1) * r_{kj}(t-1)}{\sum\limits_{v_k \in N_i(t)} r_{ik}(t-1)}, \end{aligned} \tag{21}$$

where $\alpha, \beta \in [0, 1]$, and $N_i(t)$ is the set of neighbors of v_i at time step t, i.e.,
$\mathcal{N}_i(t) = \{v_k \in V(t) : \exists (v_i, v_k) \in E(t)\}$.

It is worth noticing that α and β, together with the initial default reputation value r_0, can be declared as *varying parameters* so that they can be automatically tuned by the user in different simulation runs.

5.2 Beta Reputation

Beta Reputation [29] is one of the most well-known and cited RMSs in literature. The model is based on a statistical-mathematical formulation, which takes advantage of the Beta probability density function, and combines information about past transactions in order to obtain reputation ratings.

After each transaction, the consumer provides as feedback a positive value if the requested service was correctly provided, or a negative one if the service was not provided, or if its quality was unsatisfactory. In our implementation, feedbacks are defined as $f_{ij}(t) \in \{-1, 1\}$.

The *local reputation* of a given provider is computed taking into account the number of satisfactory and unsatisfactory transactions, over the total number of requests made during the whole simulation:

$$l_{ij}(t) = F_{local}\left(\mathbf{f_{ij}}\right) = \frac{\sum\limits_{t'=0}^{t} f_{ij}(t')\lambda_{Beta}^{(t-t')}}{2 + \sum\limits_{t'=0}^{t} \left| f_{ij}(t')\lambda_{Beta}^{(t-t')} \right|}, \tag{22}$$

where λ_{Beta} is a *forgetting factor*, defined in the range $[0, 1]$, which allows to assign lower weight to past feedbacks. The obtained values of local reputation are in the range $[-1; 1]$.

The *gossip protocol* of Beta Reputation involves that every agent communicates to its neighborhood the feedbacks generated in the last time step:

$$\mathcal{M}_{i\to k}(t) = F_{gossip}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right) = \left[\left(j, f_{ij}(t)\right)\right]_{v_j \in O_i(t), v_j \neq v_k}. \tag{23}$$

Agents exploit received messages in order to build the history of feedbacks generated by each of their neighbors. The reputation value computed through the *information fusion* method is based on the history of all received and locally generated feedbacks. Thus, according to the general equation 4, the information fusion operates as follows:

$$r_{ij}(t) = F_{fusion}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right) = \frac{\sum\limits_{t'=0}^{t} \left[f_{ij}(t') + \sum\limits_{v_k \in \mathcal{N}_i} f_{kj}(t') \right] \lambda_{Beta}^{(t-t')}}{2 + \sum\limits_{t'=0}^{t} \left[\left| f_{ij}(t') \right| + \sum\limits_{v_k \in \mathcal{N}_i} \left| f_{kj}(t') \right| \right] \lambda_{Beta}^{(t-t')}} \tag{24}$$

5.3 Core Reputation

CORE RMS [41] was introduced to prevent selfish and malicious behaviours in Mobile Ad hoc NETworks (MANETs). MANETS correct behaviour depends on the cooperation between nodes that have to perform both packets routing and forwarding. As a consequence, from a RMS point of view, MANETs can be treated as peer-to-peer networks, in which the edge $< v_i, v_j >$ indicates that v_j is in the wireless transmission range of v_i, and vice versa. CORE supports vectors of reputation values, each corresponding to a different function that a MANET node can perform, e.g., routing or forwarding. In this case study implementation, we considered just a single function in order to make the comparison with other RMSs clearer.

After each interaction, the requesting agent generates a feedback $f_{ij}(t) \in \{-1; 1\}$, with -1 corresponding to a bad experience, and +1 to a positive one.

Every agent computes the *local reputation* of its neighbors by weighting the feedbacks so as to give higher relevance to older data. The authors of [41] made this choice in order to prevent sporadic misbehavior from negatively affecting the reputation evaluation. In the implementation of this sample case, we chose to generate the weights according to the function $\rho(t, t') = 1 - \lambda_{Core}^{t-t'}$, with $\lambda_{Core} \in [0; 1]$. The *local reputation* is then evaluated through the following equation:

$$l_{ij}(t) = F_{local}\left(\mathbf{f_{ij}}\right) = K \sum_{t'=0}^{t} \rho(t, t') f_{ij}(t') = K \sum_{t'=0}^{t} (1 - \lambda_{Core}^{t-t'}) f_{ij}(t'), \tag{25}$$

where $K = \left(\sum\limits_{t'=0}^{t} \rho(t, t') \right)^{-1}$ is a normalization factor required to bring the reputation values in the range $[-1, 1]$.

According to the CORE's *gossip protocol*, the agents share only positive information in order to prevent denial of service attacks, which could be based on the broadcasting of fake negative ratings for legitimate nodes. Thus, each agent can spread local reputation only if it has a positive value:

$$\mathcal{M}_{i\to k}(t) = F_{gossip}\left(\mathbf{f_{ij}}, \mathcal{M}_i\right) = \left[\left(j, l_{ij}(t)\right)\right]_{v_j \in O_i^+(t), v_j \neq v_k}, \tag{26}$$

where $O_i^+(t)$ is the set of neighbors of v_i, for which v_i has a positive *local reputation*, i.e., $O_i^+(t) = \left\{v_j \in V(t) : \exists(v_i, v_j) \in E(t), l_{ij}(t) > 0\right\}$.

The final reputation is then computed by v_i as a linear combination of his local reputation and local reputation values received by its neighbors, as defined by equation 5:

$$r_{ij}(t) = F_{fusion}\left(r_{ij}(t-1), l_{ij}(t), \mathcal{M}_i(t)\right) = l_{ij}(t) + \frac{\sum\limits_{v_k \in N(i)} l_{kj}(t)}{\#\left\{\mathcal{M}_i(t)\right\}}. \tag{27}$$

6 EXPERIMENTAL EVALUATION

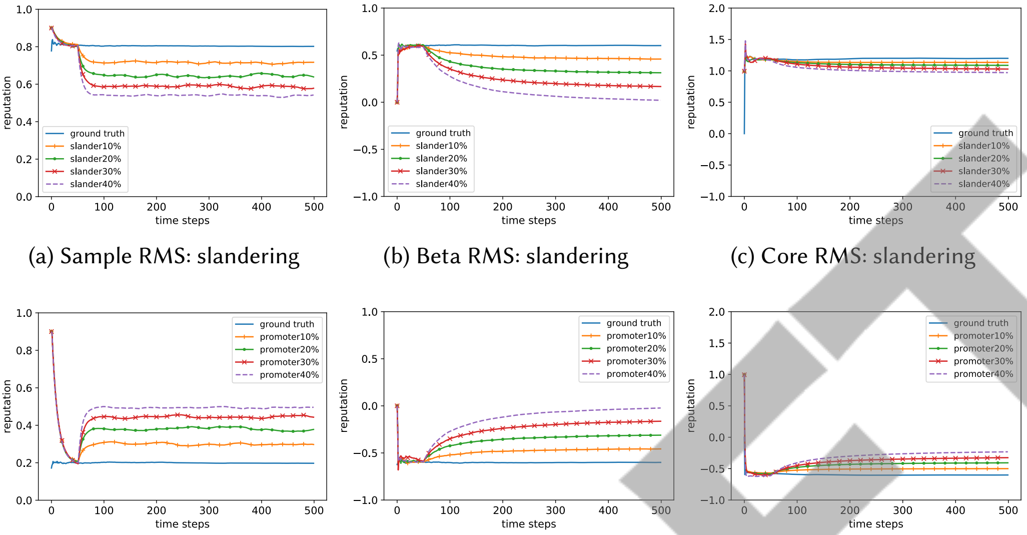
This section describes how the proposed framework can be configured to simulate the behaviour of the three RMSs described above and evaluate their robustness to security attacks. Experiments were performed considering simulations of 500 time steps, i.e., $T_{max} = 500$, on a random network of 100 nodes, each one having six neighbors on average.

6.1 Analysis of the reputation trend

The first analysis focused on the capability of the simulation framework to model *slandering*, *promoting*, *oscillation*, and *sybil* attacks. Results about the observed reputation trends can be exploited to assess the performances of the target RMS.

6.1.1 Slandering and Promoting. *Slandering* was set up by defining the ID of the target agent, which information to observe (i.e., the reputation value), and the number of attacking agents. A relevant aspect to evaluate in this type of attack is how the size of the *coalition* impacts on the reputation of the victim. Intuitively, the greater is the number of attacking agents, the more effective is the attack; however, a simulation can support this assumption by quantifying how many agents should be involved in order to achieve the attack.

Fig. 3-a,b,c show the results of different simulations performed while varying the percentage of malicious agents (10%, 20%, 30%, and 40%) in the networks of the sample RMS, Beta RMS, and Core RMS respectively. Attacks start after 50 time steps; by observing Fig. 3-a, we can notice that the reputation estimated by the sample RMS decreases and settles down approximately at time step 70. Moreover, as the size of the coalition increases, the reputation value deviates more significantly from the ground truth. For instance, when the coalition involves 40% of the agents, *slandering* is able to compromise the reputation of the victim of about 30% of the ground truth value. Fig. 3-b presents the same experiment tailored to Beta RMS, in a scenario in which the ground-truth reputation of the victim is 0.6. Note that the reputation values considered by this RMS are in the range $[-1, 1]$; thus, a reputation of 0.6 reflects a cooperative behaviour of the victim agent of 0.8, i.e., if 80 positive and 20 negative transactions were recorded, the reputation of the agent is $beta\_rep = (80 - 20)/(80 + 20 + 2) \approx 0.6$. As compared to what observed in Fig. 3-a, Beta reputation shows a smoother trend which indicates a longer time before the attack succeeds. Finally, as regards Core RMS, since one of its most notable characteristics is the spreading of positive feedbacks only, it is not surprising that the effect of *slandering* is almost negligible (see Fig. 3-c). Slight variations in the average reputation values can be attributed to the reputation evaluated by malicious agents.



(a) Sample RMS: slandering (b) Beta RMS: slandering (c) Core RMS: slandering

(d) Sample RMS: promoting (e) Beta RMS: promoting (f) Core RMS: promoting

Fig. 3. Simulation of slandering and *promoting* attacks performed against the sample RMS, Beta RMS, and Core RMS. The figures show the reputation of the victim, as observed by neutral agents, while varying the percentage of the attackers involved.

A similar setting was be adopted to simulate *promoting* attacks performed by coalitions of different size. In this case, the three plots in Fig. 3-d,e,f show the reputation of the target of the attack as perceived by the RMS. In the RMS considered as first sample case (Fig. 3-d), when 40% of the agents are malicious, the reputation of the target can be increased by 150% of its actual value in about 20 time steps from the beginning of the attack.

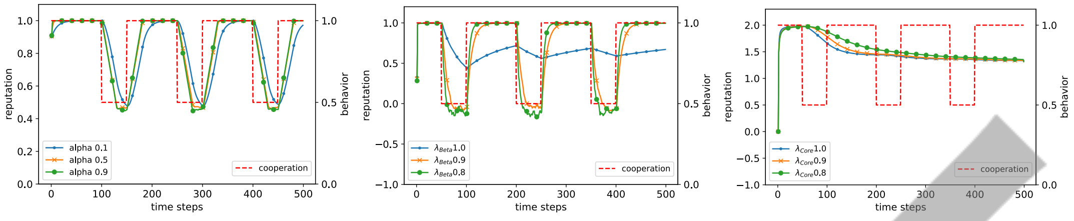
A quite similar behavior is observed in Beta, where the initial reputation value of −0.6 reflects a cooperative behaviour of 0.2. The different slopes of the four curves in Fig. 3-e highlight that also the effect of *promoting* is dependent on the size of the coalition involved. Moreover, as compared to the sample RMS, even for *promoting* it is possible to observe in Beta RMS a longer time before the attack is achieved.

Although *promoting* in Core is more effective than *slandering*, this RMS shows anyway a greater robustness to these kind of attacks because of its attitude to consider past feedbacks more than recent ones (see Fig. 3-f).

6.1.2 *Oscillation.*

Other simulations have been performed with the aim of modeling *oscillation* attacks, that require the definition of complex behaviours in which the attacker alternates cooperative (all the received requests are satisfied) and partially cooperative (only part of the requests are satisfied) periods. Such oscillations are defined as described in Section 3.1.3; during the cooperative phase, the agent is characterized by a value of $C_i(t) = 1$ for $\Delta T_1 = 100$ time steps, while the partially cooperative behaviour ($C_i(t) = 0.5$) is maintained for $\Delta T_2 = 50$ time steps.

Results in Fig. 4 show the reputation trend of an agent performing an *oscillation* attack (red line is the actual behavior) as well as the capability of the RMSs to detect the attack while varying the weight of recent experience. In the case of the sample RMS, this corresponds to choose different



(a) Sample RMS

(b) Beta RMS

(c) Core RMS

Fig. 4. Simulation of *oscillation* attacks performed against the (a) sample RMS, (b) Beta RMS, and (c) Core RMS. The three figures show the reputation of an agent alternating cooperative and selfish behaviours, as observed while varying the parameters that weights the recent experience to build the local reputation.

values of α, e.g., 0.1, 0.5, and 0.9 (see Fig. 4-a). As expected, if recent experience weights more than history, i.e., increasing values of α, changes in the agent's behaviour can be detected earlier.

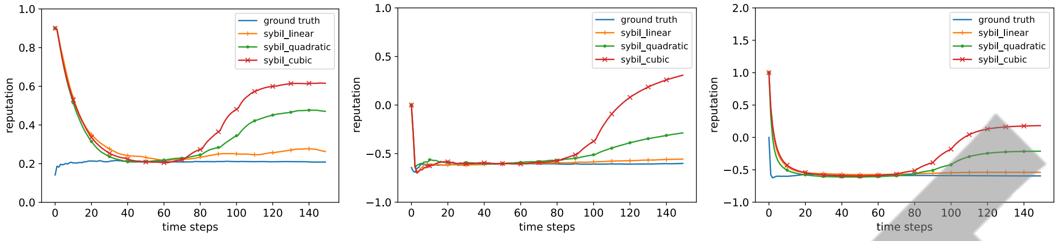
Fig. 4-b shows how, in Beta RMS, the reputation of the attacker varies according to the *forgetting factor* λ_{Beta} . Results indicate that smaller values, i.e., giving more importance to recent transactions, allow Beta RMS to better capture rapid changes in the behaviour of the attacker. On the contrary, taking into consideration all the past history ($\lambda_{Beta} = 1.0$) makes the reputation settle down to an average value, between the low and the high peaks of the oscillation.

The robustness observed in Core when dealing with *slandering* and *promoting* is not confirmed in the case of *oscillation* attacks. Due to its design, CORE is insensitive to rapid changes in agent behaviours; thus, as can be seen in Fig. 4-c, the reputation curves do not follow the oscillatory pattern depicted in red. Such a trend is very similar for the three reputation curves obtained by choosing different weights of the recent feedbacks (λ_{Core}), where the lower is the weight the more is the importance of the last transactions.

6.1.3 Sybil. The last of this set of experiments concerns *sybil* attacks carried out by a malicious agent in order to increase its relevance within the reputation network. The effect of such an attack could be mitigated, or even avoided, by the RMS thanks to the adoption of policies for regulating the registration of new users. Nevertheless, the simulation we carried out were aimed to measure the impact of *sybil* attacks on RMSs in which these countermeasures are not realized.

The three plots in Fig. 5 show the effects of an attack performed by a single malicious agent v_i that starts promoting itself at time step 50. Then, after 10 time steps it is helped by a community of sybil accounts that start joining the network with different replication factors (see Table 1) and continue the attack until time step 100.

Fig. 5-a evidences that the malicious behaviour of agent v_i causes the sample RMS to decrease its reputation from the default value $R_i(0)$ to $R_i(50) \approx 0.2$, which correctly approximates the true cooperativeness $C_i(50) = 0.2$. When the attack is launched, the reputation of v_i increases slowly and start growing faster as the number of sybil accounts increases. The three curves show the impact of different replication rates, namely linear, quadratic, and cubic, on the capability of the RMS to perform a correct estimation of the reputation value. A similar pattern is observable in Fig. 5-b and Fig. 5-c, where the behaviours of Beta and Core confirm that a higher damage is achieved when the growth of the sybil accounts is faster.



(a) Sample RMS

(b) Beta RMS

(c) Core RMS

Fig. 5. Simulation of *sybil* attacks performed against the (a) sample RMS, (b) Beta RMS, and (c) Core RMS. The three figures show the effect of a *promoting* attack supported by a community of sybil accounts created with different replication rates.

Time step	**# of sybil accounts**		
	linear	**quadratic**	**cubic**
50	1	1	1
60	2	3	4
70	3	7	14
80	4	15	41
90	5	31	122
100	6	63	365

Table 1. Replication rates of sybil accounts created to promote a target agent.

6.2 Analysis of accuracy

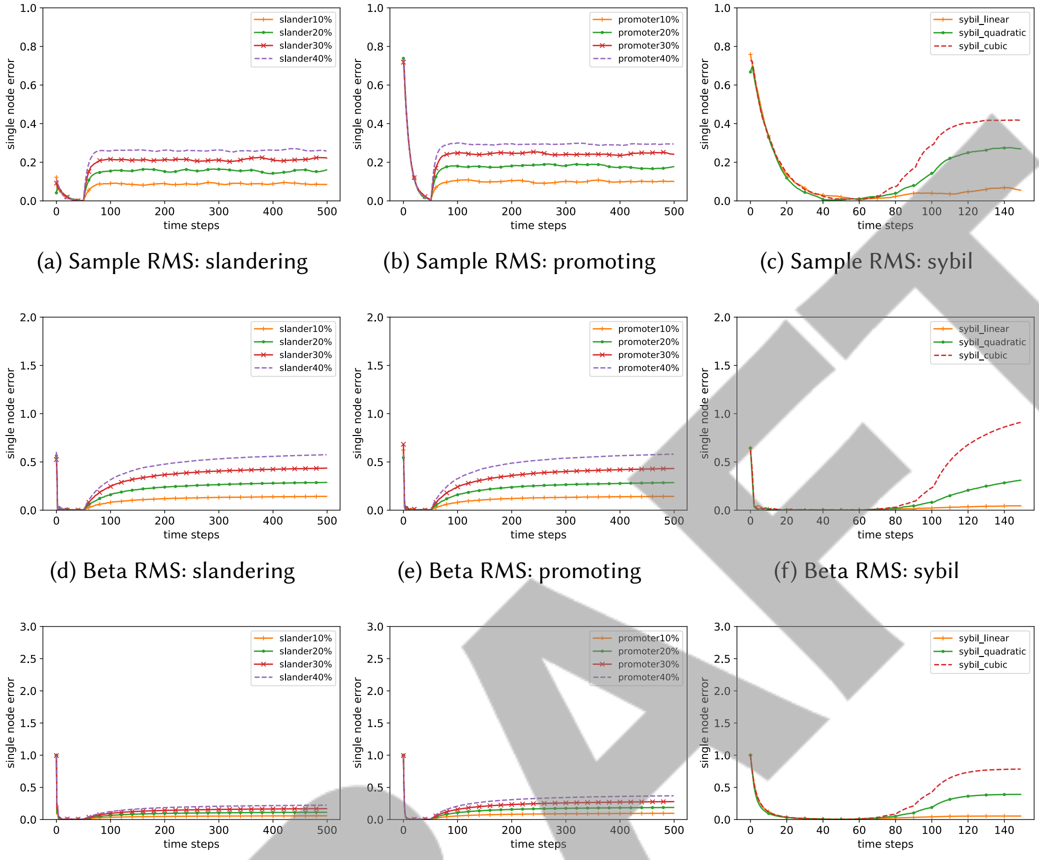
The performance of the three RMSs in managing the attacks described so far were further investigated by computing the accuracy metrics defined in Section 4.1. In particular, the *relative error* $e_i^*(t)$ achieved on a single agent is obtained as the difference between the ground truth and the average reputation of the agent v_i.

In the case of our sample RMS, the *truth-holder* computes the ground truth reputation of v_i, i.e., $R_i^*(t)$, according to the local reputation evaluation algorithm:

$$R_i^*(t) = \frac{\sum_{f_{ij} \in \mathbf{f_i}(\Delta T)} f_{ij}}{\# \left\{ \mathbf{f_i}(\Delta T) \right\}}, \tag{28}$$

where $\mathbf{f_i}(\Delta T)$ is the set of feedbacks collected by other agents about v_i during the last ΔT time steps, i.e., $\mathbf{f}_i(\Delta T) = \left\{ f_{ij}(t'), \forall v_j \in V, \forall t' \in [t - \Delta T; t] \right\}$.

A detailed analysis of the values of $e_i^*(t)$ measured during *slandering*, *promoting*, and *sybil* attacks for the target agent v_i is provided in Fig. 6-(a,b,c) respectively. Results suggest that the estimation error made by this specific RMS when dealing with *slandering* grows almost proportionally to the percentage of malicious agents involved. Thus, the RMS does not boost the fake negative opinions, thanks to the inclusion of the direct experience (i.e., the local reputation) in the aggregation algorithm. Similar results are obtained for the *promoting* attack (Figs. 6-b); this indicates that the RMS is able to handle fake positive and fake negative feedbacks equally. Finally, Fig. 6-c shows the trend of the error in the case of *sybil* attacks performed with various replication rates. In this case, the sample RMS makes an initial error due to the reputation value assigned by default, when no information about the agent is known. After a transition phase (steps 50-70) in which the



(a) Sample RMS: slandering

(b) Sample RMS: promoting

(c) Sample RMS: sybil

(d) Beta RMS: slandering

(e) Beta RMS: promoting

(f) Beta RMS: sybil

(g) Core RMS: slandering

(h) Core RMS: promoting

(i) Core RMS: sybil

Fig. 6. *Relative error* $e_i^*(t)$ on the single target of the attack, for the RMSs considered as case studies during *slandering*, *promoting* and *sybil* attacks, while varying the percentage of malicious agents involved (*slandering* and *promoting*), and the account replication rate (*sybil*).

RMS correctly estimates the reputation of v_i, the error starts growing again because of the *sybil* attack launched at time step 50. Since step 80, the three error curves have different magnitudes, proportionally to the growth of sybil accounts in the network.

In the case of Beta RMS, the ground truth reputation of v_i is computed by the *truth-holder* according to the local reputation described in Section 5.2:

$$R_i^*(t) = \frac{\sum_{t'=0}^{t} \sum_{v_j \in V(t')} f_{ij}(t') \lambda_{Beta}^{(t-t')}}{2 + \sum_{t'=0}^{t} \sum_{v_j \in V(t')} \left| f_{ij}(t') \lambda_{Beta}^{(t-t')} \right|}. \tag{29}$$

The curves shown in Fig. 6-d and Fig. 6-e suggest that the estimation errors made by Beta RMS under *slandering* and *promoting* attacks grow almost proportionally to the percentage of malicious agents involved. Thus, Beta RMS does not amplify the effect of false indirect opinions, and similar errors in the two figures point out how it makes no distinction in dealing with false positive and false negative feedbacks. Fig. 6-f indicates that, from step 80 on, the error starts growing because of the *sybil* attack launched at time step 50. Since step 90, the three error curves have different magnitudes, proportionally to the number of sybil accounts in the network. Nevetheless, Beta RMS exhibits a greater resistance to *sybil* attacks than the first sample RMS; this result is coherent with the observed greater robustness of Beta RMS to promoting attacks.

As regards Core RMS, the ground truth reputation of v_i is estimated according to the local reputation evaluation algorithm, which implements the definition described in Section 5.3:

$$R_i^*(t) = K' \sum_{t'=0}^{t} \sum_{v_j \in V(t')} (1 - \lambda_{Core}^{t-t'}) f_{ij}(t'). \tag{30}$$

where $K' = \left(\sum_{t'=0}^{t} \sum_{v_j \in V(t')} (1 - \lambda_{Core}^{t-t'}) \right)^{-1}$ is a normalization factor that leads the reputation value in the range $[-1, +1]$. Since the Core reputation values (see equation 27) are defined in $[-1, 2]$, a further nonlinear normalization is done to obtain values in the correct range.

Fig. 6-g shows that the error associated to *slandering* is very limited since no negative feedbacks are disseminated among the network. Then, the errors measured during the simulation are only dependent on the presence of an increasing number of agents (the attackers) that alter the reputation values. On the contrary, reputation estimation during *promoting* attack (Fig. 6-h) is less effective and the error made by Core grows proportionally to the size of the coalition. This indicates that Core RMS is resilient to the effect of false indirect opinions, while the different magnitudes of the errors in the two figures highlight its greater difficulty in dealing with false positive feedbacks. Finally, Fig. 6-i shows the error in the case of *sybil* attacks performed with various replication rates. Similarly to what observed in the first sample case study (Fig. 6-c), Core takes a while (about 30 time steps) to correctly estimate the reputation. After the step 80, the error starts growing because of the *sybil* attack launched at time step 50. From that time on, the three curves follow different trends according to the number of sybil accounts in the network.

6.3 Analysis of security

Besides accuracy analysis, our simulation software allows to perform a global evaluation of the considered RMS, according to the chosen set of security metrics. As described in Section 4.2, the base set of metrics includes *time-to-falsify (TF)*, *collusion-degree (CD)* and *exploitation-time (ET)*, which are summarized in a global vulnerability index. Researchers can analyze such index in order to evaluate the impact of the RMS design choices on its vulnerability to security attacks.

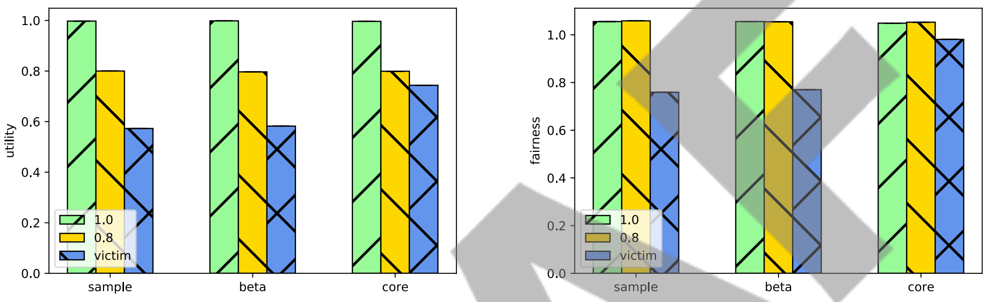
The indexes obtained by the RMS considered as first sample case study, with parameters $\alpha = 0.1$, $\beta = 0.1$ and cooperativeness $C_i(0) = 0.9$, are summarized in left block of Table 2. These results show that the RMS does not exhibit any high or critical vulnerability. In particular, the design choices produce a greater robustness to *selfish* behaviours and medium vulnerability to *promoting*, *slandering* and *oscillation* attacks.

As summarized in the middle part of Table 2, the vulnerability indexes of Beta RMS (with $\lambda_{Beta} = 0.9$) indicate that it has a quite similar resistance to *slandering* and *promoting* attacks as the sample RMS. On the ohter hand, the *exploitation-time* of *selfish* and *oscillation* attacks is lower in Beta, suggesting a greater resistance to these attacks.

Results on the security evalution of Core RMS (with $\lambda_{Core} = 0.9$) indicate a great resistance to *slandering* and *promoting*, with the corresponding vulnerability indexes $VI_{Sl}^* = VI_{Pr}^* = 0$ reflecting

	Sample RMS			**Beta RMS**			**Core RMS**		
Metric	*value*	index		*value*	index		*value*	index	
TF (slandering)	0.54	$\mathbf{VI}^*_{Sl}$	**4.80**	0.59	$\mathbf{VI}^*_{Sl}$	**4.51**	1.0	$\mathbf{VI}^*_{Sl}$	**0**
CD (slandering)	0.5			0.5			1.0		
TF (promoting)	0.55	$\mathbf{VI}^*_{Pr}$	**4.74**	0.57	$\mathbf{VI}^*_{Pr}$	**4.61**	1.0	$\mathbf{VI}^*_{Pr}$	**0**
CD (promoting)	0.5			0.5			1.0		
ET (selfish)	0.088	$\mathbf{VI}^*_{Se}$	**2.97**	0.004	$\mathbf{VI}^*_{Se}$	**0.63**	0.04	$\mathbf{VI}^*_{Se}$	**2.0**
ET (oscillation)	0.286	$\mathbf{VI}^*_{Os}$	**5.35**	0.167	$\mathbf{VI}^*_{Os}$	**4.08**	0.99	$\mathbf{VI}^*_{Os}$	**9.95**

Table 2. Security evaluation of the RMSs considered as case studies.



(a) Utility (U_{p2p_c})

(b) Fairness (U_{p2p_p})

Fig. 7. Analysis of agents' utility and fairness in sample RMS, Beta RMS, and Core RMS. Simulations are performed (see the different bars) while considering fully cooperative agents, agents with cooperativeness 0.8, and a single agent victim of *slandering*.

the non-achievement of the success condition for the two attacks. As regards the capability of detecting *selfish* behaviours, the vulnerability index suggests that Core RMS is less effective then Beta RMS, even if its performances are comparable with those of the sample RMS. Moreover, Core RMS exhibits a high vulnerability to *oscillation* attacks that start with a cooperative behavior; this is mainly due to the greater weight given to older feedbacks.

6.4 Analysis of utility

The simulation framework allows also to evaluate the utility perceived by the agents in order to support developers in assessing the effectiveness of different incentive mechanisms, or estimating the final effect of security attacks. Since the three considered RMSs are implemented in peer-to-peer scenarios, the assessment is based on the metrics U_{p2p_c} and U_{p2p_p} (see equations 16 and 17), which measure the ratio of received and requested services, and received and provided services respectively. It is important to recall that U_{p2p_p} can be also interpreted as a measurement of the fairness of the analyzed RMS, and this is how it is named in Fig. 7.

Experiments were performed considering a network where 80% of agents have a cooperativeness $C_i = 1$, 20% is characterized by $C_i = 0.8$, and a single agent with cooperatives $C_i = 0.8$ is victim of *slandering* performed by a coalition that involves 40% of network agents. The first analysis (leftmost bars in Fig. 7) concerns fully cooperative agents ($C_i = 1.0$), which represent a base-line for comparing other results. As expected, in such a case, the three RMSs produce the maximum utility and fairness values.

As regards agents with cooperativeness $C_i = 0.8$, as indicated by the middle bars in Fig. 7-a, the three RMSs cause a reduction of their utility, proportionally to their cooperativeness. Conversely, as shown in Fig. 7-b, all the RMSs are fair, i.e., each agent obtains as many resources as it provides.

The third analysis (rightmost bars in Fig. 7) is focused on a single agent, with cooperativeness degree $C_i = 0.8$, which is victim of *slandering*. Both in the sample RMS and in Beta RMS, this attack causes a 40% drop of the victim's utility, while in Core RMS the utility decrease is smaller because of the greater resistance of Core to this type of attacks. The same behaviour can be observed by analyzing the fairness values.

7 RELATED WORK

In recent years, a number of works presented new methodologies and tools aimed to assess the validity and robustness of RMSs. An effective taxonomy of related literature according to four different approaches is presented in [20].

The first type of approaches concerns *ad-hoc evaluations,* in which RMSs are analyzed by using domain-specific and ad-hoc methodologies that cannot easily be applied to other RMSs. Ad-hoc evaluations, for instance, are often carried out for the assessment of specific security issues in popular application scenarios, such as Wireless Sensor Networks [6, 21, 38, 49], or in the context of e-commerce. Some works try to provide common criteria to compare different approaches, such as in [12], where a review of 40 relevant papers that focus on computational trust and reputation models is presented. Nevertheless, although significant efforts have been made to define common strategy, the authors highlight that there are still critical open issues about the aggregation of reputation values and the most suitable incentive mechanisms. *Ad-hoc evaluation* methods do not allow to consistently address these issues, and often lead to ambiguous results.

Other works propose the adoption of a formal approach based on a *mathematical and theoretical analysis* [31, 46]. An Induced Trust Computation method (ITC), based on information theory, is presented in [57] in order to measure the usefulness of recommendations received from different agents. Such an approach is applied to three known trust and reputation models, i.e., TRAVOS [53], BLADE [44] and MET [26]. Unfortunately, despite being very accurate, *mathematical and theoretical analysis* requires a huge effort to adapt a given model to new aspects that were not intially considered. As an example, the same authors of [57] presented a new work [58] to expand their original model so as to consider also the effect of collusion-based attacks. Similarly, the evaluation method in [13] requires the RMS to be modeled as a sequence of graphs transformations and allows to manage *slandering* and *self-promotion* attacks only. Moreover, the framework does not allow to simulate dynamic agents behaviours, which may represent a relevant limitation for many RMS designers.

A third class of works follows a formal *verification* approach [4, 5, 10, 11, 24, 42] that allows to obtain precise evaluations of computational aspects of RMSs, often through automatic tools, resulting in an easier analysis as compared to the *mathematical and theoretical* one. One of the earliest works in this area [9] proposes a verification method for assessing the performance of Beta Reputation [29] against security attacks; to this aim, a probabilistic technique based on Markov Decision Processes is discussed. However, such a solution is neither unable to deal with medium-sized networks nor suitable for different RMSs. The authors of [54] and of [35] adopt, respectively, semi-ring and game theories to assess cooperation strategies between agents. An algebraic framework to evaluate access control policies based on trust management is proposed in [40]. RepSyFire [24] defines a formal verification strategy to asses strengths and weaknesses of RMSs in different environments and against different security attacks. It has been used to evaluate some well-known RMSs, but it has some limitations in describing other RMSs, as well as difficulties

	Type	**Security Attacks**		**Agent Behaviour**		**Evaluation Metrics**		
	(AH-M-S-V)	*standard*	*custom*	*standard*	*custom*	*security*	*accuracy*	*utility*
TREET [32]	S	✓		✓				✓
ITC [57]	M	✓		✓			✓	✓
ART [19]	S			✓			✓	✓
ATB [25]	S	✓	✓	✓	✓		✓	✓
DART [45]	M/V	✓		✓	✓		✓	✓
TOSim [61]	S	✓		✓		✓	✓	
RepSyFire [24]	V	✓		✓		✓		
Bidgoly(2013)[9]	V	✓		✓				✓
QTM [60]	S	✓		✓				✓
TRIM-checker [20]	V	✓	✓	✓	✓	✓		
This work	S	✓	✓	✓	✓	✓	✓	✓

Table 3. Comparison between the features provided by some relevant related works that adopt *ad-hoc* (AH), *mathematical* (M), *simulation* (S), and *verification* (V) approaches.

in considering new attacks. DART [45] adopts a model based on prisoner's dilemma games to analyze different RMSs, even against some security attacks. Its main limitation is that the chosen decision making mechanism highly influences the experimental evaluation, so preventing for a clear and unbiased assessment of other RMS's components. More recently, TRIM-checker [20], a logic-based verification framework aimed at assessing RMSs in hostile environments was presented. Such a system shows a great efficiency in evaluating modeled RMSs; however, the definition of new RMS models, especially with regards to the gossip protocols, is quite difficult for the developers. Moreover, TRIM-checker can be adopted only in scenarios with a single consumer; thus, it is not suitable to assess the effect of different design choices on a population of agents. Despite of the great potential of these approaches, the work in this area is yet immature and proposed solutions are characterized by several drawbacks or weaknesses.

Finally, other works propose *simulation-based* analyses, and the framework we describe in this paper belongs to this category. These approaches model the interactions between agents, whose behaviour is defined stochastically or by following predefined patterns. Generally, the less aspects a framework is able to deal with during the simulation (e.g., different RMSs properties, agents interactions models, security attacks), the weaker is its generality and capability of being applied in various scenarios. ART (*Agent Reputation and Trust*) [19] represents one of the earliest simulated environment that allows to compare two RMSs through objective metrics. However, it has some limitations, the most significant of which are the lack of any specific analysis of the vulnerabilities brought by the distributed RMS protocol, and forcing the designer to model the RMS while meeting the specifications of a Multi-Agent System (MAS). Such a drawbacks is common to several other simulation frameworks. TREET [32], for instance, focuses on the evaluation of RMSs in marketplace scenarios and allows to measure their robustness to some attacks. This tool introduces a certain degree of dynamism by allowing agents to randomly join or leave the simulation, but such events cannot be scheduled in advance. ATB [25] mainly addresses the modeling of the decision making mechanism, so limiting the variety of RMSs that can be evaluated. QTM [60], instead, consents to define new RMSs without limiting the characteristics of the RMS or the application scenario; however, it provides only the *hit rate* value, i.e., the percentage of successful transactions performed by cooperative users, as metric to evaluate the RMS's performance. As a consequence, the set of experiments that can be run is quite limited. TOSim [61] addresses the need for a flexible simulation environment by proposing a modular framework, also intended for scalable simulations, whose application is however limited to the evaluation of P2P overlay systems.

Table 3 briefly summarizes relevant features of some of the most important related works, in order to enable an immediate comparison with the proposed simulation framework.

8 CONCLUSIONS AND FUTURE WORK

Many distributed applications are ruled by Reputation Management Systems aimed to estimate the behaviour of unknown agents before starting a service exchange. Because of the complexity of distributed systems, designing and evaluating RMSs is very complex; for this reason, in recent years, a number of works presented new methodologies and tools aimed to assess the validity and robustness of RMSs since their design. Nevertheless, to the best of our knowledge there are no comprehensive frameworks which are able to deal with all the characteristics of a distributed RMS, regardless of the application scenario.

In order to address this lack, in this paper we presented a novel simulation platform that allows developers to model a distributed environment where several agents interact, to define the specific features of the RMS, the behavior of each agent, as well as the set of security attacks to be simulated. At the end of a simulation, the platform provides a set of *accuracy*, *vulnerability*, and *utility* metrics that can be exploited to evaluate the robustness of the RMS to security attacks which can influence its performance.

The validity of the proposed solution has been extensively tested by using the simulation framework to model three reputation management systems, namely a sample RMS, Beta Reputation, and Core Reputation. The results, aimed not at the actual evaluation of these RMSs but rather at demonstrating the capability of the framework in highlighting their peculiarities, show the effectiveness of such a simulation-based approach.

As future work we plan to investigate the performance of the platform while varying the capabilities of the exploited hardware infrastructure, both in terms of the number of cluster cores and their features, and the scale of the simulated community of agents. We are also working on the release of an open source version of the plarform that includes some popular RMSs and common attack models proposed in the literature.

REFERENCES

[1] V. Agate, A. De Paola, G. Lo Re, and M. Morana. 2016. A Simulation Framework for Evaluating Distributed Reputation Management Systems. In *Distributed Computing and Artificial Intelligence, 13th International Conference*. Springer International Publishing, Cham, 247–254.

[2] V. Agate, A. De Paola, G. Lo Re, and M. Morana. 2016. Vulnerability Evaluation of Distributed Reputation Management Systems. In *InfQ 2016 - New Frontiers in Quantitative Methods in Informatics*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 1–8.

[3] V. Agate, A. De Paola, G. Lo Re, and M. Morana. 2018. A Platform for the Evaluation of Distributed Reputation Algorithms. In *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 1–8.

[4] Alessandro Aldini. 2014. A Calculus for Trust and Reputation Systems. In *Trust Management VIII*, Jianying Zhou, Nurit Gal-Oz, Jie Zhang, and Ehud Gudes (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 173–188.

[5] Alessandro Aldini. 2018. Design and verification of trusted collective adaptive systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28, 2 (2018), 1–27.

[6] Hani Alzaid, Manal Alfaraj, Sebastian Ries, Audun Jøsang, Muneera Albabtain, and Alhanof Abuhaimed. 2013. Reputation-Based Trust Systems for Wireless Sensor Networks: A Comprehensive Review. In *Trust Management VII*, Carmen Fernández-Gago, Fabio Martinelli, Siani Pearson, and Isaac Agudo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 66–82.

[7] F. M. Awuor, C.-Y. Wang, and T.-C. Tsai. 2018. Motivating Content Sharing and Trustworthiness in Mobile Social Networks. *IEEE Access* 6 (2018), 28339–28355.

[8] S. Ba and P. A. Pavlou. 2002. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS quarterly* (2002), 243–268.

[9] Amir Jalaly Bidgoly and Behrouz Tork Ladani. 2013. Quantitative verification of beta reputation system using PRISM probabilistic model checker. In *2013 10th International ISC Conference on Information Security and Cryptology (ISCISC)*. IEEE, 1–6.

[10] Amir Jalaly Bidgoly and Behrouz Tork Ladani. 2015. Modelling and quantitative verification of reputation systems against malicious attackers. *Comput. J.* 58, 10 (2015), 2567–2582.

[11] Amir Jalaly Bidgoly and Behrouz Tork Ladani. 2016. Modeling and quantitative verification of trust systems against malicious attackers. *Comput. J.* 59, 7 (2016), 1005–1027.

[12] Diego De Siqueira Braga, Marco Niemann, Bernd Hellingrath, and Fernando Buarque De Lima Neto. 2018. Survey on Computational Trust and Reputation Models. 51, 5, Article 101 (Nov. 2018), 40 pages. https://doi.org/10.1145/3236008

[13] P. Chandrasekaran and B. Esfandiari. 2015. Toward a testbed for evaluating computational trust models: experiments and analysis. *J. of Trust Management* 2, 1 (2015), 1–27.

[14] C. Crapanzano, F. Milazzo, A. De Paola, and G. Lo Re. 2010. Reputation management for distributed service-oriented architectures. In *Fourth IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*. 160–165.

[15] CVSS. 2015. Common Vulnerability Scoring System v3.0. https://www.first.org/cvss.

[16] A. De Paola and A. Tamburo. 2008. Reputation Management in Distributed Systems. In *3rd Int. Symp. on Communications, Control and Signal Processing (ISCCSP)*. 666–670.

[17] John R Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer, 251–260.

[18] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. 2004. Free-riding and whitewashing in peer-to-peer systems. In *ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*. 228–236.

[19] K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. 2005. A specification of the agent reputation and trust (ART) testbed: experimentation and competition for trust in agent societies. In *4th Int. joint Conf. on Autonomous agents and multiagent systems*. 512–518.

[20] Seyed Asgary Ghasempouri and Behrouz Tork Ladani. 2020. Model checking of robustness properties in trust and reputation systems. *Future Generation Computer Systems* 108 (2020), 302 – 319. https://doi.org/10.1016/j.future.2020.02.070

[21] Guangjie Han, Jinfang Jiang, Lei Shu, Jianwei Niu, and Han-Chieh Chao. 2014. Management and applications of trust in Wireless Sensor Networks: A survey. *J. Comput. System Sci.* 80, 3 (2014), 602 – 617. https://doi.org/10.1016/j.jcss.2013.06.014 Special Issue on Wireless Network Intrusion.

[22] F. Hendrikx, K. Bubendorfer, and R." Chard. 2015. Reputation systems: A survey and taxonomy. *J. Parallel and Distrib. Comput.* 75 (2015), 184 – 197.

[23] K. Hoffman, D. Zage, and C. Nita-Rotaru. 2009. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)* 42 (2009).

[24] Amir Jalaly Bidgoly and Behrouz Tork Ladani. 2016. Benchmarking reputation systems: A quantitative verification approach. *Computers in Human Behavior* 57 (2016), 274 – 291. https://doi.org/10.1016/j.chb.2015.12.024

[25] D. Jelenc, R. Hermoso, J. Sabater-Mir, and D. Trček. 2013. Decision making matters: A better way to evaluate trust models. *Knowledge-Based Systems* 52 (2013), 147–164.

[26] Siwei Jiang, Jie Zhang, and Yew-Soon Ong. 2013. An evolutionary model for constructing robust trust networks.. In *AAMAS*, Vol. 13. 813–820.

[27] Håvard D Johansen, Robbert Van Renesse, Ymir Vigfusson, and Dag Johansen. 2015. Fireflies: A secure and scalable membership and gossip service. *ACM Transactions on Computer Systems (TOCS)* 33, 2 (2015), 5.

[28] Audun Jøsang. 2007. Trust and reputation systems. In *Foundations of security analysis and design IV*. Springer, 209–245.

[29] Audun Josang and Roslan Ismail. 2002. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, Vol. 5. 2502–2511.

[30] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. 2003. The Eigentrust algorithm for reputation management in P2P networks. In *12th Int. Conf. on World Wide Web*. 640–651.

[31] Reid Kerr and Robin Cohen. 2007. Towards provably secure trust and reputation systems in e-marketplaces. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. 1–3.

[32] R. Kerr and R. Cohen. 2010. TREET: the Trust and Reputation Experimentation and Evaluation Testbed. *Electronic Commerce Research* 10, 3 (01 Dec 2010), 271–290.

[33] E. Koutrouli and A. Tsalgatidou. 2016. Reputation Systems Evaluation Survey. *ACM Computing Surveys (CSUR)* 48, 3 (2016), 35.

[34] Nabila Labraoui, Mourad Gueroui, and Larbi Sekhri. 2015. On-off attacks mitigation against trust systems in wireless sensor networks. In *IFIP International Conference on Computer Science and its Applications*. Springer, 406–415.

[35] Ze Li and Haiying Shen. 2011. Game-theoretic analysis of cooperation incentive strategies in mobile ad hoc networks. *IEEE Transactions on Mobile Computing* 11, 8 (2011), 1287–1303.

[36] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. 2007. An empirical study of collusion behavior in the Maze P2P file-sharing system. In *27th Int. Conf. on Distributed Computing Systems (ICDCS'07)*. 56–56.

[37] N. A Lynch. 1996. *Distributed algorithms*. Morgan Kaufmann.

[38] F. G. Mármol and G. M. Pérez. 2009. TRMSim-WSN, trust and reputation models simulator for wireless sensor networks. In *IEEE International Conference on Communications (ICC'09)*. 1–5.

[39] S. Marti and H. Garcia-Molina. 2006. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks* 50, 4 (2006), 472–484.

[40] Fabio Martinelli. 2005. Towards an integrated formal analysis for security and trust. In *International Conference on Formal Methods for Open Object-Based Distributed Systems*. Springer, 115–130.

[41] Pietro Michiardi and Refik Molva. 2002. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced communications and multimedia security*. Springer, 107–121.

[42] Tim Muller, Yang Liu, Sjouke Mauw, and Jie Zhang. 2014. On robustness of trust systems. In *IFIP International Conference on Trust Management*. Springer, 44–60.

[43] P. Naghizadeh and M. Liu. 2016. Perceptions and truth: a mechanism design approach to crowd-sourcing reputation. *IEEE/ACM Transactions on Networking (TON)* 24, 1 (2016), 163–176.

[44] Kevin Regan, Pascal Poupart, and Robin Cohen. 2006. Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. In *AAAI*. 1206–1212.

[45] A. Salehi-Abari and T. White. 2012. DART: a distributed analysis of reputation and trust framework. *Computational Intelligence* 28, 4 (2012), 642–682.

[46] Vladimiro Sassone, Karl Krukow, and Mogens Nielsen. 2007. Towards a Formal Framework for Computational Trust. In *Formal Methods for Components and Objects*, Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem-Paul de Roever (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 175–184.

[47] Anna Satsiou and Leandros Tassiulas. 2009. Reputation-based resource allocation in P2P systems of rational users. *IEEE Transactions on Parallel and Distributed Systems* 21, 4 (2009), 466–479.

[48] Chithra Selvaraj and Sheila Anand. 2012. A survey on security issues of reputation management systems for peer-to-peer networks. *Computer Science Review* 6, 4 (2012), 145–160.

[49] Pengzhi Shi and Haiguang Chen. 2012/08. RASN: Resist on-off Attack for Wireless Sensor Networks. Atlantis Press, 690–693. https://doi.org/10.2991/iccasm.2012.175

[50] Mudhakar Srivatsa, Li Xiong, and Ling Liu. 2005. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on World Wide Web*. 422–431.

[51] Y. Sun and Y. Liu. 2012. Security of online reputation systems: The evolution of attacks and defenses. *IEEE Signal Processing Mag.* 29, 2 (2012), 87–97.

[52] S. Tadelis. 2016. The economics of reputation and feedback systems in e-commerce marketplaces. *IEEE Internet Computing* 20, 1 (2016), 12–19.

[53] WT Luke Teacy, Jigar Patel, Nicholas R Jennings, and Michael Luck. 2006. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems* 12, 2 (2006), 183–198.

[54] George Theodorakopoulos and John S Baras. 2006. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on selected areas in Communications* 24, 2 (2006), 318–328.

[55] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. 2015. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 55.

[56] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad. 2015. A survey on trust and reputation models for Web services: Single, composite, and communities. *Decision Support Systems* 74 (2015), 121–134.

[57] Dongxia Wang, Tim Muller, Athirai A Irissappane, Jie Zhang, and Yang Liu. 2015. Using information theory to improve the robustness of trust systems. In *Proceedings of the 2015 international Conference on autonomous Agents and multiagent systems*. 791–799.

[58] Dongxia Wang, Tim Muller, Jie Zhang, and Yang Liu. 2015. Quantifying robustness of trust systems against collusive unfair rating attacks using information theory. International Joint Conferences on Artificial Intelligence.

[59] K. Wang, X. Qi, L. Shu, D.-J. Deng, and J. J. Rodrigues. 2016. Toward trustworthy crowdsourcing in the social internet of things. *IEEE Wireless Communications* 23, 5 (2016), 30–36.

[60] A. G. West, S. Kannan, I. Lee, and O. Sokolsky. 2010. An evaluation framework for reputation management systems. *Trust Modeling and Management in Digital Environments: From Social Concept to System Development* (2010), 282–308.

[61] Y. Zhang, W. Wang, and S. Lü. 2007. Simulating trust overlay in p2p networks. *Computational Science–ICCS 2007* (2007), 632–639.