COMPUTERS & SECURITY XXX (2013) 1–15

Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Trust management system design for the Internet of Things: A context-aware and multi-service approach

Yosra Ben Saied [a,*], Alexis Olivereau [a,1], Djamal Zeghlache [b,2], Maryline Laurent [b,3]

[a] CEA, LIST, Communicating Systems Laboratory, F-91191 Gif-sur-Yvette, France
[b] Telecom SudParis, 91011 Evry, France

ARTICLE INFO

Article history:
Received 15 May 2013
Received in revised form
30 July 2013
Accepted 2 September 2013

Keywords:
Trust management
Collaboration
Internet of Things
Trust
Reputation

ABSTRACT

This work proposes a new trust management system (TMS) for the Internet of Things (IoT). The wide majority of these systems are today bound to the assessment of trustworthiness with respect to a single function. As such, they cannot use past experiences related to other functions. Even those that support multiple functions hide this heterogeneity by regrouping all past experiences into a single metric. These restrictions are detrimental to the adaptation of TMSs to today's emerging M2M and IoT architectures, which are characterized with heterogeneity in nodes, capabilities and services. To overcome these limitations, we design a context-aware and multi-service trust management system fitting the new requirements of the IoT. Simulation results show the good performance of the proposed system and especially highlight its ability to deter a class of common attacks designed to target trust management systems.

1. Introduction

Recently, we have witnessed the emergence of collaboration between nodes in wireless systems, opening up opportunities to improve the network performance in many respects such as increasing the radio coverage, enhancing the security or saving bandwidth and energy resources. Collaboration gained momentum with the advent of new communication schemes, more complex than those of legacy wireless systems.

Wireless sensor networks (WSNs) initiated this transition by introducing unattended wireless topologies, mostly made of resource-constrained nodes, in which radio spectrum therefore ceased to be the only resource worthy of optimization. Today's Machine to Machine (M2M) and Internet of Things (IoT) architectures further accentuated this trend, not only by involving wider architectures but also by adding heterogeneity, resource capabilities inconstancy and autonomy to once uniform and deterministic systems. In view of the IoT scenarios, a resource-constrained sensor node is considered as a part of the Internet able to establish secure end-to-end communications with external nodes not belonging to the same sensor network. However, the setup of any secure channel could be either unaffordable or prohibitively expensive for these nodes limited in terms of computing power and/

* Corresponding author. Tel.: +33 1 69 08 01 69; fax: +33 1 69 08 83 95.
E-mail addresses: yosra_bensaied@yahoo.fr, yosra.ben-saied@cea.fr (Y. Ben Saied).
[1] Tel.: +33 1 69 08 92 33; fax: +33 1 69 08 83 95.
[2] Tel.: +33 1 60 76 47 15; fax: +33 1 60 76 45 78.
[3] Tel.: +33 1 60 76 44 42; fax: +33 1 60 76 47 11.

0167-4048/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2013.09.001

or radio range. Battery-powered sensor nodes can be disseminated in hazardous environments. Some are built-in within products and are expected to have at least the same lifetime as their hosts. Changing a discharged battery could therefore be either demanding, or unacceptable. This even without considering the consequences on other neighbouring nodes, which may find themselves disconnected from the infrastructure if their default route passed through a battery-depleted node. Hence, constrained IoT nodes have a greater need to collaborate with one another in order to establish secure communications or to resolve coverage and packet delivery problems. For these reasons, multiple cooperative techniques are being proposed for many networking services in the field of modern wireless communications. Examples of such services include cooperative routing tasks (Royer and Tob, 1999) proposed to ensure reliable end-to-end delivery of an IP packet between two endpoints, cooperative radio services (Krohn et al., 2006; Hong et al. 2007) aiming to increase the radio transmission power of a delivered message or to reach a disjointed group of nodes and also cooperative security services (Mambo et al., Sep. 1996), Ben saied et al. (2011) proposed to efficiently perform heavy cryptographic operations needed to ensure security of communications involving constrained nodes.

The reliance on collaboration for any kind of service should however be done on a controlled basis. Collaboration per se may indeed open the way to a new class of attacks, all the more insidious as they involve internal attackers. During a collaborative task, it is hard to detect behaviour anomalies of a single node or a group of nodes that can disturb the proper functioning of the entire system. Conventional cryptographic mechanisms such as authentication and encryption can provide data confidentiality, data integrity and node authentication for exchanged messages and protect the system from external attacks, however, they fail to deal with insider attackers. A cooperating node owning legitimate cryptographic keys can easily launch an internal attack inside the group by altering data or injecting bogus information without being identified. Or it can act selfishly and refuse to participate to a collaborative process in order to save its energy resources and maximize its own performance. This amounts, in a nutshell, to introducing the concept of trustworthiness within a networked architecture. Having already passed cryptographic filtering barriers during network access control procedures, untrustworthy nodes have to be identified and excluded based on their behaviours. Dynamic evaluation of trustworthiness is managed by dedicated security procedures called trust management systems that aim to track nodes past interactions in the network to detect malicious attacks and selfish attitudes.

A trust management system can be instantiated at multiple places in the network, use different views and involve different algorithms with the objective to efficiently manage collaboration among nodes. This efficiency is assessed by considering how the trust management system meets its objectives in collaboration management and fulfils network requirements. However, the current transition from legacy Internet to the Internet of Things exhibits new requirements that have to be considered for the design of an efficient TMS.

In the context of the IoT, trustworthiness can be difficult to measure when different nodes providing different services have to be assessed by the same trust management system, especially when these nodes, subject to regular exhaustion of their (low) resource capabilities, become temporarily unable to provide assistance to their peers without being nevertheless to be qualified as malicious. Of course, truly malicious nodes do exist too and have to be dealt with, even though these would likely try to fail the trust metric by camouflaging their misbehaviours.

In turn, a trust management system is instantiated on a collaborative basis allowing multiple nodes to share their views about one another's trustworthiness, as induced from past interactions. Trusting reports received from other nodes is challenging in today's IoT communications, where networked entities are not only intrinsically vulnerable, but also highly heterogeneous and owned by multiple self-interest communities. Hence, nodes may lack the motivation to provide reliable evidences; instead, malicious ones may intentionally send false reports to specific victims in order to fake their decisions.

In the literature, trust management issues have long been investigated in Mobile Ad hoc Networks (MANETs) and Wireless sensor networks (WSNs). Yet, the proposed approaches lack flexibility and adaptability to both the specific requirements of IoT and the complex malicious patterns (Roman et al., 2011) that arise with the coexistence of heterogeneous and self-concerned nodes.

Based on new IoT requirements and identified shortcomings of the related work, we propose a novel trust management system for the IoT that is able to induce from nodes past behaviours in distinct cooperative services how much trust can be put into a node for accomplishing a required task. Eventually, only the best partners with respect to a sought cooperative service are proposed to a requesting node. Our system effectively fine-tunes nodes trust levels, even in presence of erroneous or malicious witnesses. Section 2 reviews the most studied trust management systems and identify a set of best practices pertaining to TMS design with respect to IoT requirements. Section 3 presents the new trust management system we propose. Performance analysis results are discussed in Section 4. Section 5 concludes this paper.

2. Related work and TMS design decisions

2.1. *Assessment of prior trust management systems*

In the literature, several trust management systems (Yuet al, 2010) have been proposed to enforce cooperation in wireless networks and analyze nodes behaviours.

In (Buchegger and Boudec, 2002), a distributed TMS called CONFIDANT is proposed for packet forwarding services. The goal of this system is to detect misbehaving nodes causing routing disruptions and therefore to enhance decisions making in the future. The proposed model takes into account both first-hand information (direct observations and own experiences) and second-hand information (indirect experiences and observations reported by neighbouring nodes) to update trust values. While reasoning over first-hand information only would have been safer, this would have been doable only for a node involved in numerous transactions with other peers.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

However, a constrained node having only sparse interactions or whose requirements are changing frequently may lack first-hand information to make trust decisions about other nodes and need a wider view of its potential peers. To make both the trust model more robust and the computed trust values more reliable, CONFIDANT therefore extends first-hand information with second-hand one, by which nodes evaluate the trustworthiness of their peers and disseminate the results of their observations throughout the network. CONFIDANT requires, though, that only negative feedbacks be exchanged between nodes. This prevents undeserved praises to propagate, but implicitly relies on the assumption that misbehaving nodes sending false negative reports will be the exception and not the norm. Obviously the system, with such assumption, is vulnerable to false reports causing the trustworthiness of benign nodes to decrease (*bad mouthing attack*).

Authors in (Michiardi and Molva, 2002) propose the CORE model. CORE is a generic trust-based mechanism aiming to detect selfish behaviours for different cooperative services. The watchdog mechanism is implemented to monitor interactions between nodes performing a cooperating service, which is not limited to packet forwarding. The model assigns a global trust value to a cooperating node for all provided services. Obviously, this encourages a malicious node to favour a *selective behaviour attack*, by which it is showing a high level of benevolence for a non-demanding service while behaving badly for a resource-intensive one, thereby being able to keep an overall fair trust value.

Unlike CONFIDANT, CORE mitigates bad-mouthing attacks caused by malicious nodes reporting false evidences to decrease the reputation value of a node. Indeed, it allows only positive witnesses to be propagated in the network, assuming that a node has no advantage to give a false praise about unknown nodes. Nevertheless, this system overlooks the case where nodes collude with one another by disseminating false positive evidences to increase their reputation values (*ballot stuffing attack*).

In (Buchegger and Le Boudec, June 2004), authors highlight that previous trust models are vulnerable to erroneous reports. They propose that reputation values be kept local while the node monitors only its one-hop neighbours through direct observations. Once a non-cooperative behaviour is detected, benign neighbours will redirect received packets through another route to avoid the misbehaving next-hop node. This latter is implicitly rejected from the network since in turn all of its neighbours will reject its packets, as responses to its future routing service requests.

RFSN (Ganeriwal and Srivastava, 2004) is the first trust-based model proposed for wireless sensor networks to monitor sensor nodes interactions. Each node maintains trust values of other nodes using both its direct observations from the watchdog mechanism and second-hand information from other nodes observations. Like CORE, the proposed system allows only positive observations to be propagated, making the bad mouthing attack impossible. It relies on the trustworthiness score of the witness node to weigh its reports, in order to overcome ballot stuffing attacks.

An agent-based trust model for wireless sensor networks is presented in (Chen et al., 2007). Authors highlight that sensor nodes may not be able to handle complex computations or process the required information to build trust views about other nodes. They propose to move the heavy computational and storage cost from these constrained devices to dedicated agents in charge of cooperation management inside the network. The proposed system claims to be safe from bad mouthing and ballot-stuffing attacks, assuming that the deployed agents are trusted-third parties and would not engender these types of attacks.

Very recently, limited work on trust management has been proposed in the context of the IoT. Chen et al. (Oct. 2011) have oriented the design of TMS towards a specific IoT environment consisting of wireless sensors only and evaluated trustworthiness for the packet forwarding service. The trust computation load is kept at the constrained device and reports are disseminated among neighbours. Assuming an IoT environment made of sensor nodes only is a blemish for the design of their proposed TMS. The IoT paradigm, largely extending the sensor networking model, targets indeed universality and global interoperability and aims to interconnect much wider sets of objects and networks ranging from resource-constrained wireless sensors to powerful servers. The heterogeneous aspect of IoT architecture is a key criterion that has to be considered when managing trust among the different nodes it is made up of. A low-resource device and a powerful server providing poor assistance for the same collaborative service should not be treated in the same way. Likewise, an IoT node may switch from a context to another due to changes in its amount of available resources (energy exhaustion, energy harvesting) and/or changes affecting other variable parameters (mobility, availability). Hence, the fact that a node has a good reputation when in a specific context gives few or even no information about how much it can be trusted to provide assistance for a cooperative service after having switched to another context. A node classified as honest could behave well with 80% of available resources. Yet, there would not be any guarantee that the same level of benevolence would be obtained for the same service in another situation where only 20% of its resources would be available. In order to improve trustworthiness predictability in the context of the IoT, additional contextual parameters are therefore to be considered, such as energy resources and availability of the scored entities.

A second work on trust management systems for the IoT is presented in Bao and Chen (September 2012). Authors recognize the dynamic statuses of IoT nodes and propose different metrics to compute the trust level of a node, including its social cooperativeness as a service provider, its community-interest and discrepancy in its recommendations. The proposed TMS defines a weighting factor to evaluate the confidence put in recommendations received from other nodes. This factor increases proportionally with the global trust level of the node as a service assistant and can be adjusted through a β parameter in accordance with the environment hostility and the resilience of the system against bad mouthing and ballot-stuffing attacks.

However, it can be argued against the weighting factor design in Bao and Chen (September 2012) that estimating a node's trustworthiness when providing reports basing on its trustworthiness score when assisting in a service may lead to inaccuracies. An honest low-resource node can indeed be untrusted for providing assistance for cooperative services

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

because of its resource constraints while still being able to provide good recommendations about other nodes assisting it. On the other hand, mixing these two trustworthiness scores together would encourage a malicious node to take advantage of this fact and send correct recommendations while misbehaving as a service assistant. It would then remain overall trusted, since its bad behaviour in service setup would be compensated with good behaviour in recommendations. Then again, malicious nodes may provide dishonest reports, while behaving well as assistant nodes, in order to boost the trust values of malicious peers (ballot stuffing attack) or to drop trustworthiness of honest parties (bad mouthing attack). Currently, there is no trust management system in wireless communications that handles reports basing on the trustworthiness of nodes as reporting nodes.

2.2. *Synthesis*

In Table 1, we summarize our assessment of prior trust models practices with respect to new requirements of the IoT. From these constraints that apply in the IoT world, we identify a set of corresponding design decisions that provide us guidance to design an effective trust management system for the IoT.

3. Proposed trust management system

3.1. *Overview*

The main objective of the proposed solution is to manage cooperation in a heterogeneous IoT architecture involving nodes with different resource capabilities, in order to establish a community of trusted elements assisting one another with respect to the operation of a set of collaborative services.

The choice between centralized and decentralized instantiation of the trust management system must take into account its complexity in terms of trust computation formulae and processed information quantity. The alternative between centralized and decentralized approaches can be stated as follows. The trust information can be computed on demand, whenever a node needs to rely on cooperative peers, and delivered to the requesting node at that moment (centralized approaches). Or it can be computed on a regular basis and be propagated throughout the topology (decentralized approaches). A real-time trust information flow would result in communication overhead, detrimental to network performance as well as to constrained nodes battery life. In addition, unsolicited trust information would have to be stored for subsequent use, which memory-constrained nodes could not afford. This leads us to favouring a centralized approach wherein different trust management servers covering different geographical locations handle trust computational load, based on a wide range of reports.

A description of the different phases of the proposed model is presented in Fig. 1 below. This model involves a cyclic succession of operations wherein: 1) the trust management system (trust manager) obtains information about the trustworthiness of the available proxies, 2) the trust management system issues recommendations about proxies to a requesting node that intends to set up a collaborative service, 3) the requesting node relies on the collaborative service provided by the recommended proxies, 4) the

Table 1 – Assessment of TMS design decisions.

Prior trust models practices	IoT requirements/constraints	TMS design decisions
Assess trust levels similarly for different nodes in the network.	Variable contexts of IoT nodes and different resource capabilities.	Evaluate the trust level of a node by taking into account additional parameters concerning its current context and resource capabilities.
Define a global trust score for all assisted services (thereby making the system vulnerable to selective attacks).	A node trusted to provide assistance for a lightweight service is not necessarily trusted to assist for a service that requires more resources and increased availability during the service execution.	Design a functional trust model that takes into account the specific demanding aspects of the assisted service when computing the trust level of a node having accomplished it.
Restrict the reception of certain reports from witness nodes to avoid bad mouthing (only positive reports) or ballot stuffing attacks (only negative reports).	Lack of information to take trust decisions due to the sparse interactions of constrained IoT nodes.	Consider all received reports and past interactions in making trust decisions by defining new methods to perform their combination and bypass the underlying attacks.
Do not separate received reports from witness nodes basing on their quality of recommendation.	IoT nodes belong to different self-interest groups and may provide false witnesses since they do not work towards the same goal.	Weigh reports basing on the trustworthiness of nodes as reporting nodes.
Correlate the node quality of recommendation to its trustworthiness when assisting services.	Trusting a node as an assistant node does not imply trusting it as a reporting node.	
Consider both centralized and decentralized instantiation of the trust management system.	Most of IoT nodes are characterized by low capabilities in terms of both memory and computing resources, which make them unable to support the complexity of trust computation and data storage.	Favour the centralized approach to offload the underlying charge from constrained nodes and reduce the communication overhead between nodes.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

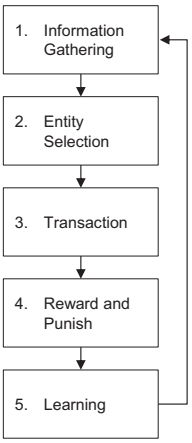

Fig. 1 – Proposed model phases.

requesting node assesses the quality of each individual service provision from each assisting proxy and 5) the trust management system learns from its past operation by performing self-updates intended to improve its future operation. These five phases our proposed model is made up of are detailed in the next subsection.

3.2. *Operation phases*

3.2.1. *Initialization and information gathering*

At the beginning of the lifetime of the network, all nodes are assumed to be trustworthy and well-behaving since they are supposed to be verified for failures before deployment. Once the network becomes operational, it may happen that nodes be compromised. Their trustworthiness levels can therefore change with respect to their behaviours.

Before being able to produce trustworthy results, a trust management system has to gather enough information from the network, during a so-called bootstrapping period whose precise definition depends on the requirements on the recommendation quality. A trust management system is indeed expected to produce better results over time: a compromised node may remain unnoticed for a while; but it will be all the easier spotted if it gets involved in a large number of transactions, all of which are poorly rated.

However the bootstrapping period can be very long since a true assessment of nodes behaviours can be a lengthy operation. In order to minimize the bootstrapping period, we assume that, with respect to some services for which it can masquerade as a demanding node, the trust manager initially participates in the setup of the trust management process by targeting some nodes and inducing artificial interactions between them, in order to rate their trustworthiness. When a service is provided, the requesting node is able to evaluate the behaviour of each assisting node as either positive or negative, depending on whether it has accomplished its assigned task properly or not.

These evaluations are stored in the trust manager and used as inputs for the trust management system. In order to make assisting nodes recommendations more accurate and specific, there is a need to store additional contextual metrics concerning the type of executed service, namely the time of execution and the current state of the evaluated node (ageing, resource capacity, etc.) It is indeed important to know in which circumstances the cooperating node has obtained the reported evaluations.

Contrary to what is proposed in the literature, our trust model proposes an objective mechanism providing dynamic trust ratings for the same node, adapted to the different behaviours exhibited in different contexts. This mechanism states that an evaluation report is accompanied with a set of contextual parameters, as follows.

The report R_{ij} refers to the jth report sent to evaluate the quality of the service provided by an assisting node P_i is therefore made up of the following information:

- $[S_j]$ (Service): Service for which the node Pi provided assistance.
- $[C_j]$ (Capability): Capability of node Pi when assisting the service.
- $[N_j]$ (Note): Score given by the requester node to Pi for evaluating the offered service. This value belongs to $\{-1, 0, 1\}$.
- $[T_j]$ (Time): Time at which the service was obtained.

3.2.2. *Entity selection*

Upon receiving a request from a node asking for assistance, the trust manager starts the entity selection process to return a set of trustworthy assisting nodes to the requester. We propose a step-by-step selection process. This process is the most important of the trust management system. It is made up of five consecutive steps.

3.2.2.1. Step 1: restriction of the set of proxies p_i. At this stage, the system restrains the set of nodes by selecting the potential candidates. This selection depends on the requirements of the service. Lightweight communications may require that all assisting nodes belong to the same multicast group and can therefore be contacted simultaneously through the sending of a single message. In the case of signature delegation schemes, the requesting server looks for assisting nodes dispersed in specific locations in the network in order to sign messages on its behalf, and hence to ensure service availability to all of its clients. In radio transmission services, neighbours in the same radio range are likely to be the possible candidates for assistance.

3.2.2.2. Step 2: restriction of the set of reports R_{ij} for each proxy P_i. After the prior selection, a set of nodes are designated to compete for the final selection. In order to rate the trust level of each of these candidates, the trust manager needs first to narrow the set of collected reports about each node independently. The most meaningful reports are those that pertain to the same context as of the current request: ideal reports would be pertaining to the same service that is being requested; they would also have been issued when the evaluated nodes were in the same status as of the new request moment.

It is very likely, though, that the system will not find enough such ideal reports to calculate the trustworthiness of a

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

node in a specific context. This may happen either because the candidate node has not yet been evaluated for the current requested service, or because it was in a different condition when evaluated for the same service. To resolve the problem of this lack of information, we proposed to calculate the context similarity.

The graph below in Fig. 2 describes how we restrain the set of potential reports needed to evaluate the trust level of a node by considering the principle of context similarity in terms of type of service (*x*-axis) and node capabilities (*y*-axis). This two-dimensional context representation assumes that one is able to quantify the two values it relies on. Node capabilities can easily be quantified, for example as a percentage of node resources in terms of processing power, memory and/or battery level. It is more complex to quantify the former term, namely context similarity in terms of type of service, since multiple collaborative services exist that share little in common.

We consider that an adequate metric for assessing service similarity is the amount of resources that are required to run a given service. Within the resources that can be measured, we recommend to consider energy consumption whose decrease is generally a strong incentive to selfish behaviours. Let us take an example of how we use service similarity in order to measure a context similarity. We assume for example that both a cooperative key establishment service and a signature delegation service require the same level of resource capabilities (delegation of asymmetric cryptography operations). So that, receiving a report about a node performing one of these security services at around the same resource capabilities level can be used to evaluate the trust level of this node for performing the other security service.

Fig. 2 presents various reports R_{ij} (Service S_j, Capability C_j, Note N_j) stored at the trust manager, sent by all nodes j evaluating past interactions with a common assisting node P_i. This figure can be read as follows. The horizontal axis on the graph shows the different services for which the evaluated node P_i has provided assistance before. These services are ordered according to their resource-demanding requirements. The vertical axis shows the capabilities of the P_i node when assisting for these services. Each graph is characterized by the target report R_{Target} (S_{Target}, C_{Target}) depicted as a black diamond on Fig. 2:

- [S_{Target}] (Service Target) is the current service in request.
- [C_{Target}] (Capability Target) is the current P_i capability.

R_{Target} refers to the next report to be received, in case the proxy P_i is selected for the current service assistance. The goal of the context similarity computation process is to retrieve from the graph the most relevant reports, helping the trust manager to foresee the score received within the target report if the proxy P_i is retained for the service assistance.

Context similarity between a report about a previous interaction and the present target report is computed by considering a global contextual distance *dij* between the old report and the target one. To compute d_{ij}, we first define dS_j as the difference between the target service S_{Target} and the report service S_j and dC_j as the difference between the target capacity C_{Target} and the report capacity C_j.

$$dS_j = |S_{Target} - S_j| \tag{1}$$

$$dC_j = |C_{Target} - C_j| \tag{2}$$

We then obtain d_{ij} as:

$$d_{ij} = \min\left(\sqrt{(dS_{max}^2 + dC_{max}^2) \times \left(\frac{dS_j^2}{dS_{max}^2} + \frac{dC_j^2}{dC_{max}^2}\right)},\ \sqrt{(dS_{max}^2 + dC_{max}^2) \times \left(\left(\frac{(S_{max} - S_j)}{S_{max} - (S_{Target} - \eta)}\right)^2 + \left(\frac{C_j}{C_{Target} + \eta}\right)^2\right)}\right) \tag{3}$$

(for reports carrying a positive evaluation)

Or:

$$d_{ij} = \min\left(\sqrt{(dS_{max}^2 + dC_{max}^2) \times \left(\frac{dS_j^2}{dS_{max}^2} + \frac{dC_j^2}{dC_{max}^2}\right)},\ \sqrt{(dS_{max}^2 + dC_{max}^2) \times \left(\left(\frac{(C_{max} - C_j)}{C_{max} - (C_{Target} - \eta)}\right)^2 + \left(\frac{S_j}{S_{Target} + \eta}\right)^2\right)}\right) \tag{4}$$

(for reports carrying a negative evaluation)

The purpose of this computation is to make the distance metric more subtle than if it was merely measuring the sole similarity of an old report to a current situation. Indeed, some reports are meaningful although they are not close to the (S_{Target}, C_{Target}) target on the graph. To that respect, an asymmetry arises. A node behaving well for an expensive service is likely to behave well for a less demanding service too, whereas the fact that a node behaves well for a simple service gives no information about its expected quality when providing assistance for a demanding service.

The computation of d_{ij} takes this asymmetry into account by decreasing the distance (hence, increasing the probability to be selected) for the reports that give a good score when the evaluated node was at a much lower capability level, or those that give a bad score when the evaluated node was at a much higher capability level. Fig. 3 below explains how Equations (3) and (4) orient the selection of the most relevant reports, and how the chosen parameters affect the distance computation. Indeed, each of these equations is obtained as the min of two terms. The first term merely relates to the distance between

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

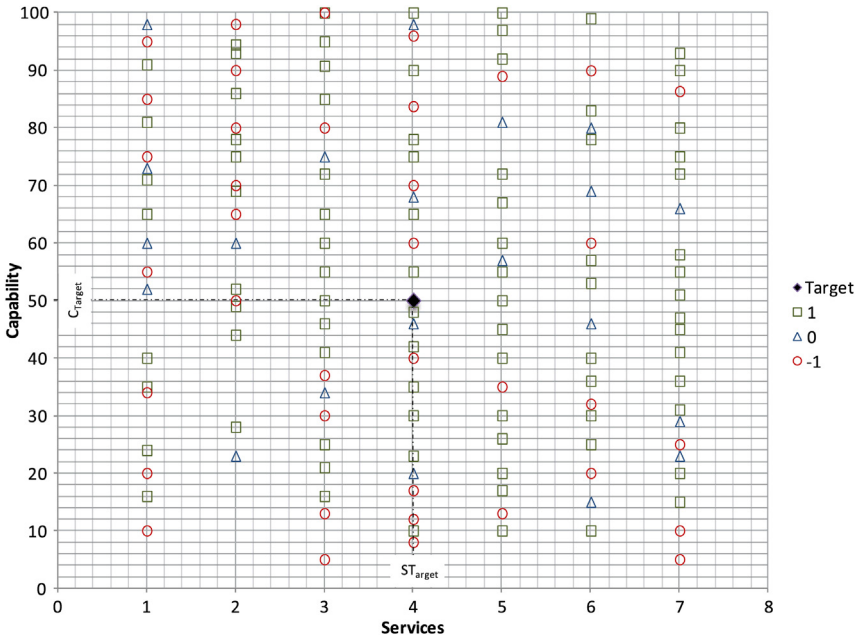

Fig. 2 – Proxy reports history.

the evaluated report and the target. It is equal to $\sqrt{(dS_{max}^2 + dC_{max}^2)}$ for the points that belong to the ((S_{Target}, C_{Target}), dS_{max}, dC_{max}) ellipse, and tends to zero when a report gets closer to the centre of that ellipse. The dS_{max} and dC_{max}, respectively x and y semi-axes of the ellipse, express the tolerance of the selection mechanism. The larger dS_{max} (resp. dC_{max}), the smaller the increase of distance when Sj (resp. C_j) gets further from S_{Target} (resp. C_{Target}).

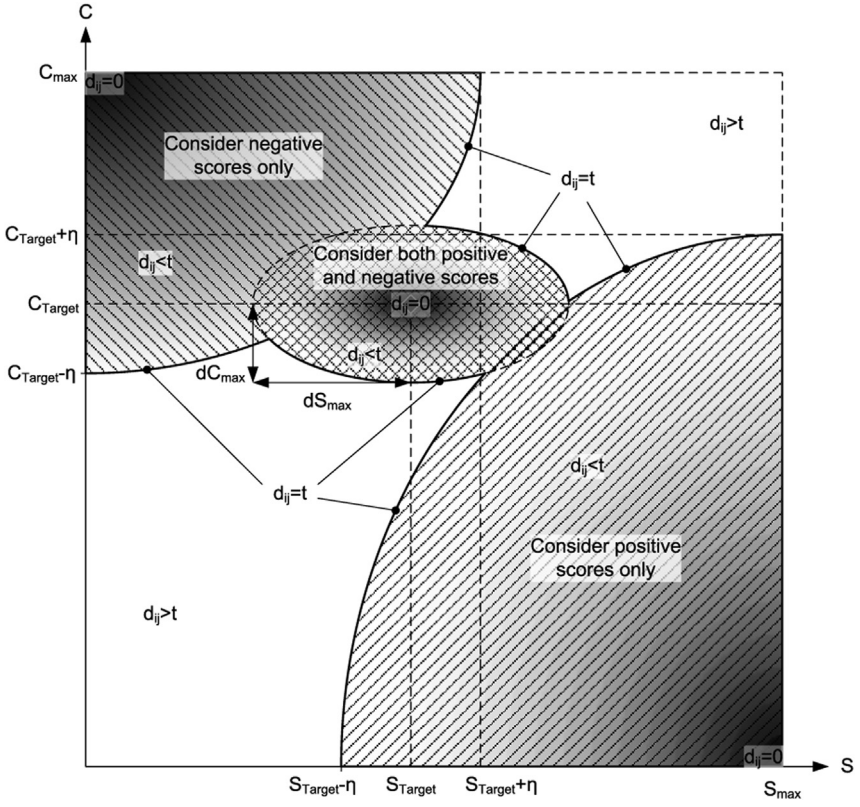

Fig. 3 – Schematic representation of reports selection functions.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

The second term is where said asymmetry comes into play: it is proportional to the distance between the evaluated report and the point (S_{max}, 0) for positive scores, or to the distance between the evaluated report and the point (0, C_{max}) for negative scores. S_{max} refers to the most complex service in terms of resource consumption and C_{max} is the maximum resource level that could be available at a node. A positive report close to (S_{max}, 0) means that the candidate node performed well for a complex service while having only few available resources. A negative report close to (0, C_{max}) means that it performed poorly for a simple service, while being nevertheless at the maximum of its resource availability.

The parameter η is an adjustable parameter that allows to take into account through the second term a greater number of significant reports, by enlarging the upper-left and lower-right quarters of ellipses, thereby increasing the number of considered reports.

Finally, the computed d_{ij} distance is used as follows: a retained report R_{ij} should have a distance d_{ij} (R_{ij}, R_{Target}) $< t$, with $t = \sqrt{dS_{max}^2 + dC_{max}^2}$ acting as an adjustable threshold, characterizing the similarity interval we want to use.

Three domains are represented on Fig. 3. The central ellipse is where reports are considered relevant under the dS_{max}, dC_{max} tolerance factors. The upper-left and lower-right quarters of ellipses, whose size can be adjusted through the η parameter, represent the areas where reports are meaningful in accordance with the score they carry. For each domain, the darker the shading colour, the lower the d_{ij} distance. The white areas represent the portions of the graph where the reports are not selected, since their computed distance exceeds the threshold t.

An example of the d_{ij} variation with (S_j, C_j) positive reports for (S_{Target}, C_{Target}, dS_{max}, dC_{max}) = (50, 70, 25, 15) is provided in Fig. 4.

Using the defined distance for restricting the set of considered reports leads to selecting only a subset of them (from Fig. 2), as shown below in Fig. 5.

3.2.2.3. Step 3: computation of the weights $w_{R_{ij}}$ for each retained report R_{ij} in the step 2. Among the set of selected reports, not all have the same significance: those exhibiting a smaller contextual distance d_{ij} are more relevant than those with higher d_{ij} values. Meanwhile, old reports may not always be relevant for the ongoing trust rating, because a node may change its behaviour over time: recent reports are thus more meaningful than reports obtained for a long time. It is therefore necessary to assign a weighted value for each report, which bases on those two considerations and expresses the overall report relevance for the selection phase.

The weight w_{Rij} of the R_{ij} report is thus calculated as a product of two exponential factors that respectively decrease with the report age ($t_{now} - t_j$) and the report contextual distance calculated above in step 2. The adopted scheme gives progressively less weight to older and contextually more distant reports.

$$w_{R_{ij}} = \lambda^{d_{ij}} \theta^{(s+1)\left(t_{now} - t_j\right)} \tag{5}$$

With:

- λ, θ being parameters in the range of [0, 1] that express the 'memory' of the system. θ (resp. λ) is adjusted according to the expected rapidity of change in the observed node along the time (resp. among services). The lower θ (resp. λ), the lower importance the system gives to past (resp. contextually more distant) reports.
- $s = 1/2 * (N_j^2 - N_j)$ being a parameter computed from N_j (the score given by the witness node in the report R_{ij}) such that s

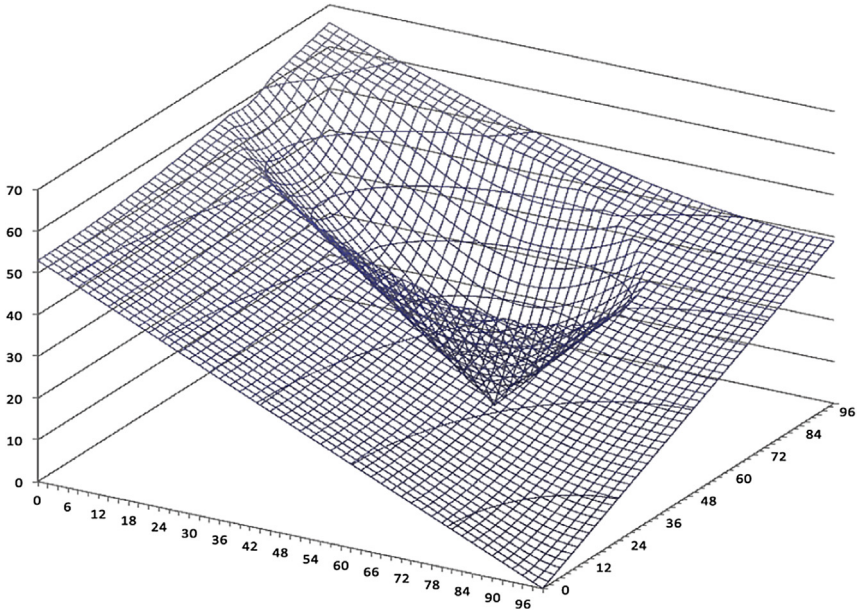

Fig. 4 – Contextual distance for positive reports. As can be seen, the reports having the minimal d_{ij} distances are those that are either close to the target (central ellipse), or that reflect a node behaving particularly well in difficult conditions (bottom right corner).

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

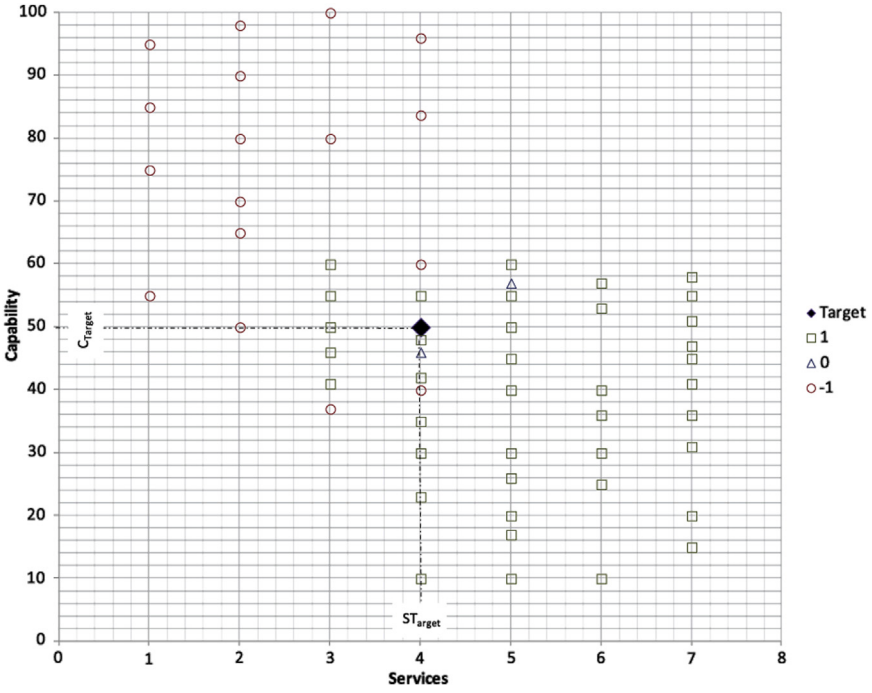

Fig. 5 – Retained proxy reports. After computing the contextual distance of proxy P_i reports originally depicted in Fig. 2, only those for which $d_{ij} < t$ are retained.

is equal to 1 when this score is equal to −1 and 0 when this score is either 0 or 1. This way, the weight of negative score is doubled as compared to that of neutral or positive scores. The goal of the weighting factor *s* will be clarified in what follows.

3.2.2.4. *Step 4: computation of the trust value Ti for each proxy pi.* At this stage, the system is able to combine all opinions about the evaluated proxy P_i. This happens through a weighted average where the trustworthiness T_i of the proxy P_i for the sought collaborative service is eventually obtained as follows:

$$T_i = \frac{1}{\sum_{j=1}^{n} w_{R_{ij}}} \times \sum_{j=1}^{n} \left(w_{R_{ij}} \cdot QR_j \cdot N_j \right) \quad (6)$$

With:

- QR_j (Quality of Recommendation of the node *j* having issued the report R_{ij} about the proxy P_i) is the trustworthiness score assigned to a witness node depending on the accuracy of its past reports. It ranges between −1 and 1, 1 representing a very trustworthy node and −1 a node reporting the opposite of the actual service quality.
- $w_{R_{ij}}$ is the weighting factor computed above in step 3.

3.2.2.5. *Step 5: provision of the best rated proxies p_i.* Upon computing trust levels for all selected candidates, the trust manager responds to the requesting node by securely providing it with the list of the best rated nodes, in accordance with the sought collaborative service and the respective current statuses of the assessed proxies.

3.2.3. *Transaction and evaluation*

In order to perform its planned collaborative service, the client node relies on the list of assisting nodes obtained from the trust manager. At the end of the transaction, the client node is able to assess the offered service received from each assisting node and sends a report to the trust manager in which it either rewards (positive score) or punishes (negative score) the participating nodes. The technique carried out to assess the offered assistance depends on the type of the service. It could be either derived from the client node local observations or from feedbacks received from other peers involved in the collaborative process, such as neighbours or the destination node. For example, in routing services, a source node is able to detect misbehaving nodes that refuse to relay its packets through local watchdog observations that notify it in accordance with routing protocol acknowledgement syntax; the source node is then in position to transmit this information to the trust manager (any lightweight protocol such as the IoT-purposed CoAP (Shelby et al., June 2013) would be relevant for doing so). In the case of a collaborative key establishment service, the client node is unable to locally gather information about proxies assisting its key exchange with a remote entity. Nevertheless, at the end of the key exchange and as part of the regular key establishment protocol, the remote peer provides the client node with a feedback specifying the list of participating proxies and/or those participating to the key exchange but having sent bogus shares. Here again, the client node can transmit this list to the trust management system using a lightweight CoAP-like protocol.

It is then of high importance to deal with received reports in our TMS by adequately taking care of the credibility of the node providing it. This is what the next 'Learning' operation is about.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

3.2.4. *Learning*

The learning phase of our proposed trust management system qualifies it as a cognitive process. This phase is what distinguishes a cognitive process from an adaptive one. Translated to security scenarios, this means that adaptive security consists in dynamically reacting to a change in the environment by applying new security policies while cognitive security introduces a learning step wherein an assessment of the enforced action is carried out, which will eventually modify the system behaviour, so that a different action may be taken when the same situation occurs. Indeed, a cognitive process is classically (Mahmoud, 2007) described as a cycle involving four steps namely observation, planning, action and learning.

These steps almost straightforwardly correspond to the phases proposed in our TMS, as depicted in Table 2.

The proposed learning phase consists of two steps: quality of recommendation update step and reputation update step.

3.2.4.1. Update of witness nodes' qualities of recommendation. Having received a report evaluating an assisting node, the trust manager learns about its behaviour. The trust manager can then update the trustworthiness score of all nodes having already sent a report about the same proxy, in similar contextual conditions. The underlying idea is quite simple: a node having previously marked as 'bad' a proxy node that eventually received a 'good' score will be considered a poor recommender (irrespective of its trustworthiness with respect to assistance in collaborative service, if any) and its Quality of Recommendation (QR) will be decreased (made closer to −1). Likewise, a node having previously given a good mark to a good-rated node will be considered as a good recommender, and its QR will be increased (made closer to 1).

This can be achieved by applying a weighted average function for trustworthiness score for each cooperative node on each node having sent a usable report about this cooperative node. This weighted average function serves two purposes. First, it avoids excessive variations of the QR. For example, a generally good recommender will not suddenly be classified as a poor one if it issues an erroneous report, but its recent history will mitigate its QR decrease. Second, the weighted average function allows to choose precisely to which extent a node's QR must be oriented either towards 1 (good recommender), or 0 (reporting non-usable data), or −1 (maliciously reporting the opposite of what happened). To that respect, weighting is important since a node being erroneous in one old report relative to a contextually distant service will be far less penalized than a node being wrong in a very recent report about the same service, provided at the same capability level. The QR of the node having issued the report used to update the QRs of nodes having sent reports about the same proxy node is also an important parameter to take into consideration in the computation of the weight: saying the opposite of a very good recommender is more penalizing than contradicting a barely trustworthy recommender.

Let X be a witness node that helped the trust manager to evaluate a node P, which was used later as a proxy for assisting the node F. Depending on whether it has successfully accomplished the assigned task, the node F sends a report R_F to the trust manager that contains an evaluation score N: {−1: bad; 0: neutral; 1: good}.

The trust manager uses this report to update the recommendation trustworthiness score QR of each node having participated as a recommender during the proxy selection stage (which means that the report issued by this node must have been judged relevant at step 2, from contextual distance point of view). We defined X as being one such node.

The steps the learning stage is made up of are the following:

- First the system retrieves the n stored quality recommendation scores (that is, the history of their recommendation quality) for all witness nodes. X has for example QR^x (QR_1,…,QR_{n-1},QR_n) with QR_1 being the last updated (the most recent) quality recommendation score.
- The system then extracts the note N from the received report R_F and retrieves the weight w_{R_X} corresponding to R_x.

Afterwards, it calculates QR_F, which represents the direction towards which the QR should evolve. QR_F is computed as follows:

$$QR_F^X = C_F \times r \quad \begin{cases} r = -|N^X - N^F| + 1 \\ C_F = w_{R_X} \times QR_1^F \end{cases} \tag{7}$$

In the above formula, r is computed from N^x (the note previously given by X) and N^F (the note just given by F) such that r is equal to 1 when these notes are identical, to −1 when they are opposite and to 0 when they differ by 1. r is therefore the value towards which the weighted average function should lean the QR of the witness node X, since this latter must tend towards 1 when the report is coherent with the newly received one, and tend to −1 when it contradicts it.

In accordance with our weighted average approach, C_F is the weight of r. As explained above, C_F increases when the weight of the report previously sent by X is high (an error by X is less tolerable if it pertains to a similar context). It also increases if F is a good recommender, as expressed with the QR_n^F factor (the current recommendation quality of node F).

Finally, the system computes the new recommendation quality N_QR for node X as follows:

$$N\_QR^X = \frac{1}{\sum_{i=1}^{n} c_i + |C_F|} \times \left(\sum_{i=1}^{n} c_i \cdot QR_i + QR_F^X \right) \tag{8}$$

The last term of this weighted average, r (in the form of QR_F that includes its weighting factor) has already been discussed above. The other terms are the QR_i, which are representative of the history of X's recommendation quality. Their respective weightings, c_i, are computed such as to be weighting values that assign a higher weight to the latest recommendation quality values. We propose to have c_i defined as (θ being presented in Step 3):

Table 2 – A Cognitive trust management system.

Cognitive process terminology	Proposed TMS terminology
Observation	Information gathering
Planning	Entity selection
Action	Transaction
	Reward and punish
Learning	Learn

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

$$c_i = \theta^{(t_{QR_1} - t_{QR_i})} \quad (9)$$

Once computed, the N_QR value is added to the QR historic list, stored in the trust and reputation manager. It will be used as a recommendation quality for the future processes.

N_QR can fall off to below zero and becomes negative which means that the witness node is reporting the opposite of the real service quality. At that time, instead of applying a report discard, we propose to consider the opposite of what is provided.

3.2.4.2. *Update of assisting nodes' reputation levels*. We distinguish in this work between trust and reputation concepts. While *trust* measures the ability of a node to fulfil a specific task in a specific context, *reputation* refers to the global opinion of a node's trustworthiness in the network after having provided assistance for various services. The reputation level of an assisting proxy P_i is computed as follows:

$$\mathrm{Rep}_{P_i} = \left(\sum_{j=1}^{n} c_j \cdot N^{F_j} \cdot QR_{F_j} \right) \quad (10)$$

N^{F_j} is the score given by the requesting node F_j having obtained the assistance of P_i for a specific service and QR_{F_j} is its quality of recommendation. The weighting factor c_j, presented above, is applied to gradually forget old feedbacks.

It is important to update reputation levels of nodes in the network after each interaction in order to identify assisting nodes commonly judged as untrustworthy. Upon receiving a feedback from the requester node F, the trust manager takes into account its evaluations to recalculate the reputation levels of the involved assisting nodes. If the reputation level of one of these falls below a certain threshold, its activity is interrupted and it is added to a list of ill-reputed nodes. It is also reported by the trust manager to the network operator, which may then examine the reasons for its misbehaviour. Indeed, a node might provide erroneous information or bad services either due to a deliberate, malicious misbehaviour, or just as a result of a malfunction or an environmental change.

4. Performance analysis

We conducted in this section an experimental study to verify and prove the effectiveness of the proposed system in managing trust and enforcing collaboration between nodes. The studied network model depicted in Fig. 6 is deduced from the IoT paradigm we envision: we consider a local IoT infrastructure that interconnects heterogeneous nodes with different capabilities (e.g. battery-powered devices and unconstrained nodes that could be either line-powered or having energy harvesting capability). These nodes are connected to the Internet and able to communicate with external peers not belonging to the same infrastructure through a decentralized pattern. There exists a local trust management system within this infrastructure that manages collaboration between nodes for multiple networking services.

Experiments were run using a purposely designed simulator on which the creation of databases, trust patterns, behaviours and interactions model was easier than with classical network simulators.

4.1. *Initial setup and QR evolution*

The simulation configuration in Table 3 was used:

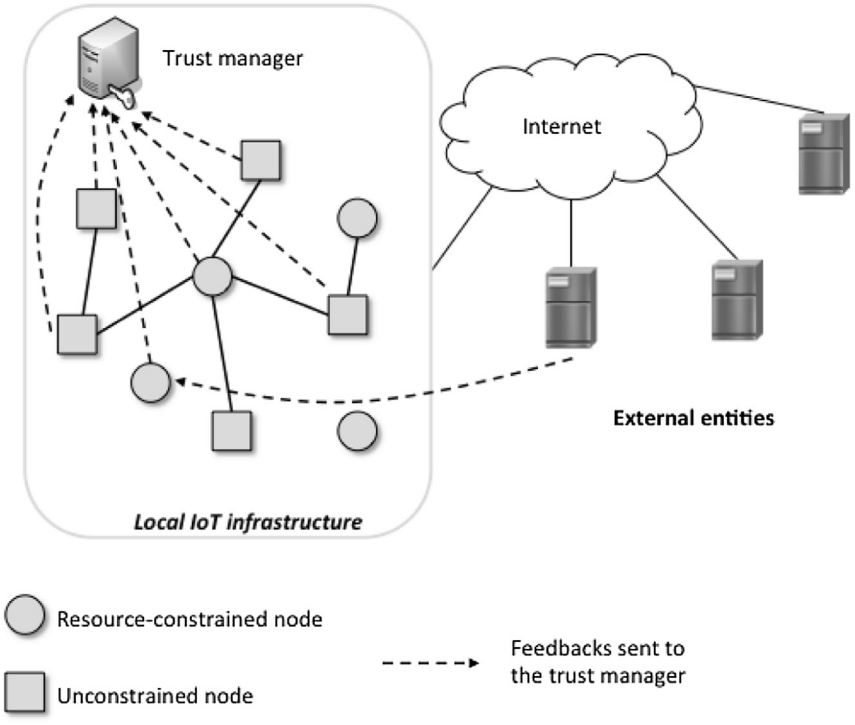

Fig. 6 – Considered network model.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

Table 3 – Simulation configuration parameters.

Number of nodes	200
Number of constrained nodes	100
Poor witness nodes (%)	20%
Malicious assisting nodes (%)	10%
Initial quality of recommendation (QR)	1
Services	6

Table 4 – Simulated node attribute set.

Node ID	12
Position (x,y,z)	(600,20,0)
Energy Level	100
Quality of recommendation (QR)	1
Services	{S4,S2}
Malicious node	False
R_QR	0.8

Constrained nodes are unable to provide assistance to other nodes but are capable to assess the offered assistance and send reports to the trust manager. Each of the simulated nodes is characterized by a set of attributes, e.g. for node 12 (Table 4):

This node is located at location (600, 20) with a maximum energy level. It is able to provide assistance for the service 4 and 2. From the system point of view, its quality of recommendation is initially set to 1 in order to be adjusted progressively revealing its real trustworthiness level as a witness node. Meanwhile, a real quality of recommendation R_QR (set in this example of node to 0.8) transparent to the trust manager is set, that defines the intrinsic behaviour of each node when reporting evaluations about other nodes.

As mentioned in the design of our solution, received reports about other nodes are then used as input in the trust manager in order to calculate trust values; a clear vision of the quality of recommendation influences thus directly trust computations, leading to reliable decision makings and offering the best assistance to requesting nodes: discarding poor/lying recommenders and promoting efficient recommending nodes are indeed required in order to have the computed trust match the actual trustworthiness of an assisting node.

This provides us with a simple means to check whether the proposed TMS behaves properly: if yes, the interpolated quality of recommendation should tend towards the real quality of recommendation. The figures below show examples of the evolution of the quality of recommendation for some nodes (Figs. 7–10).

Simulation results confirm the proper operation of the proposed trust management system. As shown above, positions of curves in the vicinity of the R_QR prove that the integrated learning module performs properly and succeeds in fine-tuning the quality of recommendations.

4.2. *Comparison of reactions against attacks*

In order to prove the effectiveness of our TMS, we evaluate its conduct towards the commonly studied attacks that may occur in trust management systems namely bad mouthing attack, on-off attack and selective behaviour attack (mentioned earlier in the paper):

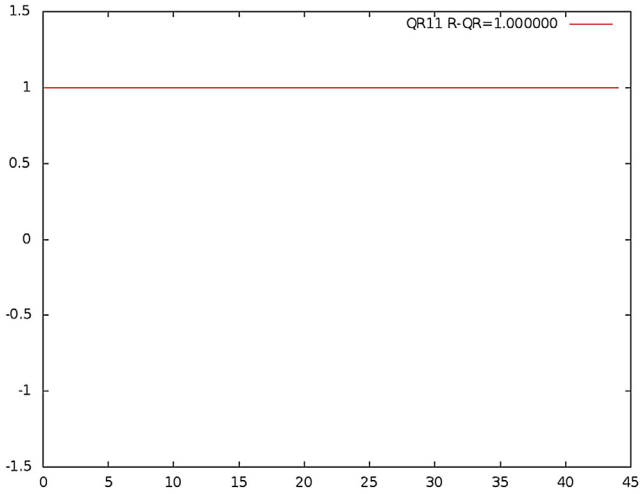

Fig. 7 – A perfect recommender (QR = 1) is recognized as such by the TRM, which constantly assigns it the "1″ score as quality of recommender. No incident interferes with this rating.

4.2.1. *Protection against on-off attack*

This attack exploits the forgetting property of trust management systems, which gives more weight to recent recommendations. A dishonest entity can take advantage of this property and behave alternatively well and badly, since it can compensate past bad behaviours by behaving well for a period of time and eventually regaining trust. This attitude is referred to as an on-off attack.

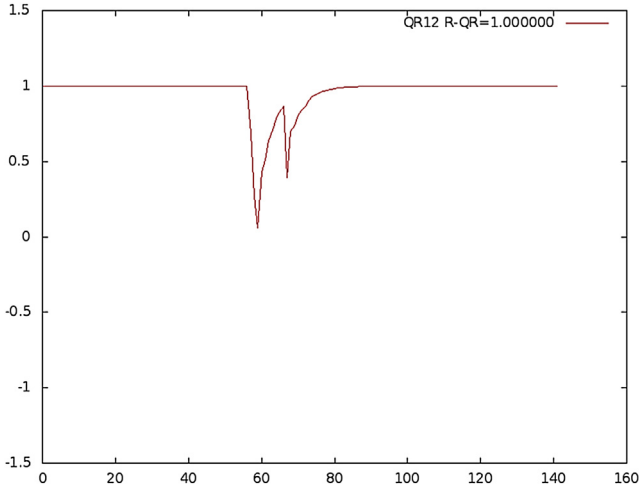

Fig. 8 – Here, a node that is intrinsically a perfect recommender has its quality of recommender score initialized at 1. Two incidents, caused by poor witnesses' errors cause the TRM to decrease the score. However, the system behaves properly and quickly restores the proper value.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

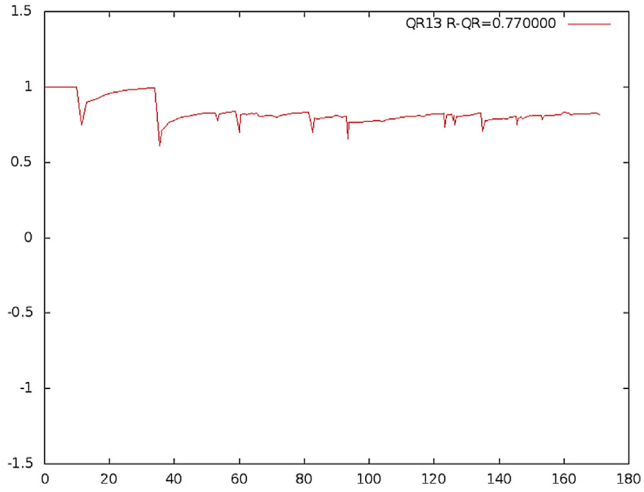

Fig. 9 – We see here a situation where the considered node is a good, yet not perfect, recommender. QR = 0.77 means that the node, though generally giving a good recommendation, will be erroneous 23% of time. Hence, as compared with the previous case, this node's quality of recommender is not only affected by poor witnesses but also by its own errors.

In order to make our trust model robust against such attacks, we adapt our system such that a bad behaviour will be memorized for a longer time than a good behaviour. We accordingly add a weighting factor *s* in the computation of the

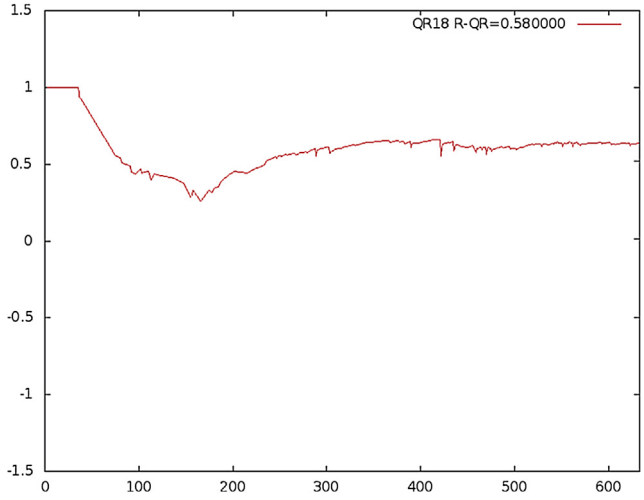

Fig. 10 – With an even lower real QR, the node's score is regularly affected by its own mistakes, in addition to the erroneous reports from poor witnesses. The score therefore oscillates between 0 (node is estimated to issue useless reports) and 1 (node is estimated to issue trustworthy reports), with rare occurrence of negative scores (node is estimated to be intentionally issuing false reports). Mitigation of these oscillations would require relying on non-linear formulae: trying to mask them with slower increase/decrease slopes only would also slow down the convergence of the system for recognizing a fully trustworthy node (more frequent case) and would therefore damage the entire system behaviour.

report age in step 2, so that we make negative scores appear less old, compared with neutral and positive nodes.

This decision discourages dishonest nodes to recurrently switch between bad and good behaviours and require them to perform many good actions to recover their trust values.

We see in Fig. 11 a situation where the node changes its behaviour alternatively. It behaves well for the ten first interactions. Then it provides bad services for the second ten interactions and reverts to normal. Without considering *s* (blue graph), the system takes more time to detect the bad behaviour of the node since the node past good behaviour is more emphasized, and therefore hides the malicious transition for longer. Once the system recognizes this bad behaviour and starts to slightly decrease its trust level, the node stops bad behaviours and regains trust. With the use of *s* (red graph), the system detects earlier the node misbehaviour and decreases its trust level. Since bad behaviours are memorized for a longer time, it takes much longer for the node to regain trust from the system point of view. (in the web version)

4.2.2. *Protection against bad mouthing attack*

As long as reports from witness nodes are taken into account in a trust management system, the risk of receiving wrong recommendations is present. A bad mouthing attack occurs when malicious nodes provide dishonest recommendations to drop trustworthiness of honest parties. Our TMS defends against this attack by building and updating separately trust recommendation values from regular trust values. As presented above, Quality of recommendation scores (QRs) are updated by checking consistency between the current evaluation and previous recommendations used during the proxy selection phase. A malicious witness node can be detected during the learning process by putting its dishonest recommendations up against others' evaluations, which progressively decreases its QR and reduces the impact of its future recommendations. As shown in Fig. 12, without considering the QR, the trust level of an honest node (red graph) significantly drops when impacted by a bad mouthing attack (characterized in this case by a group of ten witness nodes sending negative evaluations about a well-behaving assistant node). Considering the QR of a node when assessing its reports, the system becomes able to decrease the QR of these malicious witness nodes by putting their dishonest

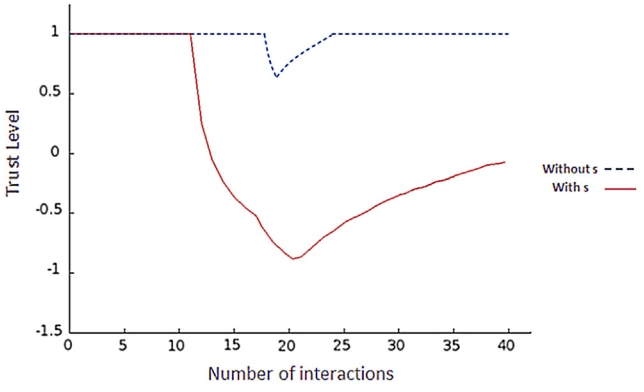

Fig. 11 – Trust level evolution of the in presence of on-off attack.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001

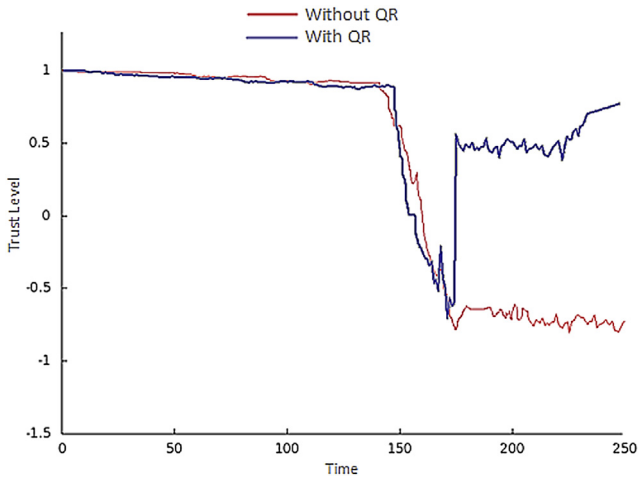

Fig. 12 – Trust level evolution of the in presence of bad mouthing attack.

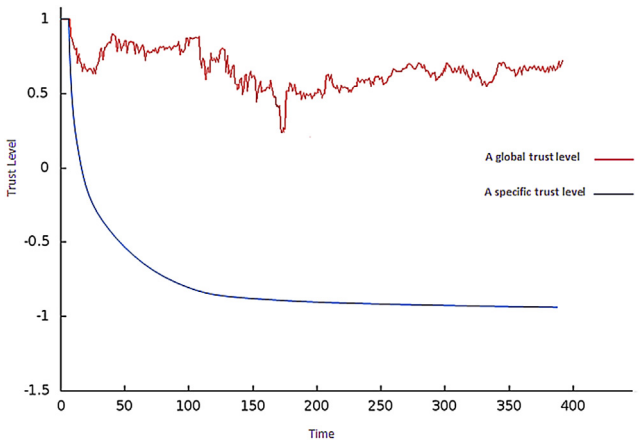

Fig. 13 – Trust level evolution in presence of selective behaviour attack.

recommendations up against others' evaluations. Our trust model (blue graph) decreases in a first time the node trust level but quickly recovers its trustworthiness by reducing the impact of wrong reports provided by malicious nodes. (in the web version)

4.2.3. *Protection against selective behaviour attack*

While a dishonest node switches between bad and good behaviours over time in the on-off attack described above, it can also behave alternatively badly and well between services. This attack is referred to as selective behaviour attack. If a node behaves well for simple services, it can still behave badly for other resource-demanding services. Thereby, the average trust level will remain positive and the node will selfishly save energy.

Existing trust models suffer from this attack since they rely on a unique trust value that globally characterizes a node including all assisted services. Our system defends against selective behaviour attack through the implementation of a functional model that assigns multiple trust values to a node, in relation with all assisted services. A node that would always perform poorly in demanding collaborative services would always receive bad scores, which would not be compensated for by good scores obtained for good behaviour in simpler services. In the short term, this means that the node carrying out this attack would no longer be selected for demanding services, which it would no longer be in position to damage. In the longer term, such behaviour could trigger action from the system administrator, if the accumulation of poor scores reaches a predetermined threshold.

We consider in Fig. 13 a situation where the trust level of a dishonest node is evaluated with respect to a resource demanding service. We can see that this node, being considered under a global trust value, manages to hide its misbehaviour when performing this service. It maintains an overall high trust level (red graph) since it compensates received bad scores with good scores obtained for its good behaviours in simpler services. Our trust model (blue graph) succeeds to decrease the trust level of the node when performing this specific service despite its good behaviours in other services. (in the web version)

5. Conclusion

In this paper, we presented a generic context-aware trust management system for the IoT aiming at addressing shortcomings of prior proposals and new requirements of wireless communications. The proposed model assigns dynamic trust scores to cooperating nodes according to different contexts (what was the status of the assisting node?) and different functions (for which service the assistance of that node was required?). A careful design was given to clarify how much confidence should be put in a report provider. A quality of recommendation score is assigned to each node reflecting its trustworthiness when rating other nodes and adjusted after each interaction during the learning phase. Simulations prove the proper operation of the proposed TMS. We pay a particular attention to its behaviour when subject to a class of attacks specifically designed to target trust management systems. Results show that our proposed system is able to withstand these attacks more efficiently than its counterparts that exist in the literature.

Acknowledgements

This work was financially supported by the EC under grant agreement FP7-ICT-257521 IoT A project.

REFERENCES

Bao F, Chen IR. Dynamic trust management for Internet of Things Applications. In: 2012 International workshop on self-aware Internet of Things, San Jose, California, USA September 2012.

Ben saied Y, Olivereau A, Zeghlache D. Energy efficiency in M2M networks: a cooperative key establishment system. In: 3rd Int. congress on ultra modern telecommunications and control systems (ICUMT) 2011.

Buchegger S, Boudec J-YL. Performance analysis of the CONFIDANT protocol. In: Proc. 3rd ACM int. symp. mobile ad hoc netw. comput., Lausanne, Switzerland 2002. p. 226–36.
Buchegger S, Le Boudec J-Y. A robust reputation system for peer-to-peer and mobile ad-hoc networks. Proceedings of P2PEcon 2004. Cambridge MA, U.S.A: Harvard University; June 2004.
Chen H, Wu H, Zhou X, Gao C. Agent-based trust model in wireless sensor networks. In: Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing 2007. p. 119–24. SNPD 03.
Chen D, Chang G, Sun D, Li J, Jia J, Wang X. TRM-IoT: a trust management model based on fuzzy reputation for Internet of Things. Comput Sci Inf Syst Oct. 2011;8(4):1207–28.
Ganeriwal S, Srivastava MB. Reputation-based framework for high integrity sensor networks. In: Proc. ACM security for ad-hoc and sensor networks 2004. p. 66–7.
Hong Y-W, Huang W-J, Chiu F-H, Kuo C-CJ. Cooperative communications resource constrained wireless networks. IEEE Signal Process Mag 2007;24:47.
Krohn A, Beigl M, Decker C, Riedel T, Zimmer T, Garces D. Increasing connectivity in wireless sensor network using cooperative transmission. In: 3rd International conference on networked sensing systems (INSS), Chicago, USA 2006.
Mahmoud Q. Cognitive networks: towards self-aware networks. John Wiley and Sons; 2007.
Mambo M, Usuda K, Okamoto E. Proxy signatures: delegation of the power to sign messages. IEICE Trans Fundamentals Sep. 1996;E79-A(9):1338–53.
Michiardi P, Molva R. CORE: a COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Proc. 6th int. conf. commun. Multimedia security 2002. p. 107–21.
Roman R, Najera P, Lopez J. Securing the Internet of Things. Computer 2011;44(9):51–8.
Royer E, Tob C-K. A review of current routing protocols for ad hoc wireless networks. IEEE pers commun 1999:46–55.
Shelby Z, Hartke K, Bormann C. Constrained application protocol (CoAP). draft-ietf-core-coap-18 (work in progress); June 2013.
Yu H, et al. A survey of trust and reputation management systems in wireless communications. Proc IEEE 2010;98(10):1755–72.

Yosra Ben Saied graduated in 2010 from the Higher School of Communications of Tunis (Sup'Com), where she obtained her National Diploma in Telecommunications Engineering and Master's degree in Telecommunications. In 2010 she joined as a PhD student the Laboratory of Communicating Systems (LSC) at French Atomic Energy Commission. Her research activities consist in developing network security solutions for constrained systems. She especially focuses on cognitive and collaborative mechanisms.

Alexis Olivereau graduated from École Nationale Supérieure de l'Électronique et ses Applications, Cergy, France in 2000. Between 2000 and 2008 he has been a research engineer in the Motorola Labs research center of Paris, France where he worked on networking security, developing novel protocols for IP-based architectures in the framework of mobile Internet. He participated in various European research projects and earned Motorola "Gold Badge" distinction for his patents filing. He joined the Laboratory of Communicating Systems (LSC) of CEA-LIST in January 2009 as a researcher and is now working on security and privacy aspects of communications in machine-to-machine and cloud environments.

Maryline Laurent, PhD works as a professor at Telecom SudParis, Mines-Telecom Institute, and is the head of the research team R3S (Network, Systems, Services, Security) of the French CNRS UMR 5157 SAMOVAR. Her main topics of interest are related to network security and privacy. She chaired the Security track of the IFIP International Conference on New Technologies, Mobility and Security NTMS 2011, and ICSNA 2011 (International Conference on Secure Networking and Applications). She is currently coediting a book on identity management, Ed. Lavoisier, and she is editor of the special issue on "Privacy-aware electronic society", Annals of Telecommunications.

Djamal Zeghlache Professor graduated from SMU in Dallas, Texas in 1987 with a PhD in Electrical Engineering and joined the same year Cleveland State University as an Assistant Professor. In 1990, 1991 he worked with the NASA Lewis Research Centre on mobile satellite terminals, systems and applications. In 1992 he joined the Networks and Services Department at Telecom SudParis where he currently acts as Professor and Head of the Wireless Networks and Multimedia Services Department. He co-authored around one hundred in ranked international conferences and journals and was an editor for IEEE Transactions on Wireless Communications.

Please cite this article in press as: Ben Saied Y, et al., Trust management system design for the Internet of Things: A context-aware and multi-service approach, Computers & Security (2013), http://dx.doi.org/10.1016/j.cose.2013.09.001