Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Grant Agreement: 786698

D5.4 - Trust management service: security and privacy

Work Package 5: Key proactive technologies cyber-threat intelligence

Document Dissemination Level

P	Public	☒
CO	Confidential, only for members of the Consortium (including the Commission Services)	☐

Document Due Date: 30/04/2020
Document Submission Date: 29/04/2020

Co-funded by the Horizon 2020 Framework Programme of the European Union

Document Information

Deliverable number:	**D5.4**
Deliverable title:	Trust management service: security and privacy
Deliverable version:	V0.1
Work Package number:	WP5
Work Package title:	Key proactive technologies and cyber-threat intelligence
Due Date of delivery:	30/04/2020
Actual date of delivery:	29/04/2020
Dissemination level:	Public
Editor(s):	Costas Vassilakis (UOP)
Contributor(s):	Costas Vassilakis, Christos–Minas Mathas, Nicholas Kolokotronis (UOP) Stavros Shiaeles, Gueltoum Bendiab (UOPHEC) Simone Naldini, Stefano Cuomo (MATH)
Reviewer(s):	Olga Gkotsopoulou, Paul Quinn (VUB) Michael Skitsas (ADITESS)
Project name:	Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Project Acronym	Cyber-Trust
Project starting date:	1/5/2018
Project duration:	36 months
Rights:	Cyber-Trust Consortium

Version History

Version	Date	Beneficiary	Description
0.1	3/2/2020	UOP	Table of contents and distribution of work.
0.2	22/2/2020	UOP	First version of the trust computation algorithm.
0.3	9/3/2020	UOP	Elaboration of the TMS internal architecture (detailed view).
0.4	17/3/2020	UOP	Reiteration on the state of the art of TMS architectures and models and models.
0.45	21/3/2020	UOP	Reiteration on the state of the art of TMS implementations.
0.5	26/3/2020	UOP	Second version of the trust computation algorithm and first version of attack scenarios and validation.
0.6	2/4/2020	UOP	Final version of the trust computation algorithm.
0.7	20/4/2020	UOP	Final version of attack scenarios & first version of tuning and evaluation.
0.8	26/4/2020	UOP	Final version of the evaluation, first complete version of the deliverable.
0.9	28/4/2020	VUB, ADITESS	Reviewer comments received.
1.0	29/4/2020	UOP	Accommodation of the reviewers' comments and submission of final version

Acronyms

ACRONYM	EXPLANATION
AMPQ	Advanced Message Queuing Protocol
APIs	Application Programming Interface
BMA	Bad-mouthing attack
BSA	Ballot stuffing attack
CIDR	Classes Inter-Domain Routing
CPE	common platform enumeration
CPS	Cyber-physical system
CT	Cyber-Trust
CTI	Cyber-Threat Intelligence
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerabilities Scoring System
D	Deliverable
D2D	Device-to-device
DB	Database
DBMS	Database Management System
DDoS	Distributed denial-of-service
DHT	Distributed Hash Table
DoS	Denial-of-service
ENISA	European Union Agency for Cybersecurity
ERNT	Extended RPL Node Trustworthiness
eVDB	Enriched vulnerability database
FR	Functional Requirement
FTBAC	Fuzzy approach to the Trust Based Access Control
GUI	Graphic User Interface
ID	Identification
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
JNI	Java Native Interface
JSON	JavaScript Object Notation
KPI	Key Performance Indicator

M	Month
MAC	Medium access control
MUD	Manufacturer Usage Description
NFR	Non-Functional Requirement
OOA	On-off attacks
ORM	Object-relational mapping
OSA	Opportunistic service attack
PDR	Packet delivery ratio
POJO	Plain Old Java Object
QoS	Quality of service
REST	Representational State Transfer
RFID	Radio frequency identification
RPL	Routing Protocol for Low Power and Lossy Networks
SaaS	Software as a service
SIoT	Social Internet-of-Things
SLA	Service-level agreement
SMA	Sybil-Mobile attack
SOA	Service oriented architecture
SoC	System-on-chip
SOHO	Small office/Home office
SPA	Self-promotion attack
SQL	Structured Query Language
T	Task
TA	Trust agent
TMS	Trust Management Service
TPM	Trusted Platform Module
UI	User Interface
URL	Uniform Resource Locator
VIP	Verification of interaction proof
VM	Virtual Machine
WP	Work Package
WSN	Wireless sensor network
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Executive summary

In the context of IoT, the number of active subjects (devices, users and applications) for which security needs to be regulated is extremely high, severely limiting the applicability of traditional security approaches, which require explicit specification of authentication and authorization parameters for each of the active subjects. To this end, trust-based solutions have been proposed; in the context of a trust-based approach, pairwise and typically unidirectional trust levels between two entities are computed, and the computed trust level is then used to moderate the interaction between devices, including the determination of allowable communications or fine-grained access to information offered by different services, underpinning thus both the security and privacy dimensions.

The Cyber-Trust architecture encompasses the Trust Management Service (TMS) module, which realizes an authoritative entity for trust computation, maintenance and distribution. To accomplish this task, the TMS synthesizes information from components of the Cyber-Trust platform, including the Profiling Service, the Cyber-Defence service as well as the Network architecture and assets Repository (A16). Through this information, the TMS synthesizes a comprehensive view on the device, considering the aspects of *status* (corresponding to the device health level), *behaviour* (corresponding to the activities that the device has detected to be involved in and an assessment of the compliance of these activities) and *associated risk* (corresponding to the level of harm that can be caused by the device). Based on this view, it calculates a trust score. To further exploit the information available in the Cyber-Trust platform towards the synthesis of a comprehensive assessment of the devices' trust levels, TMS modules may establish pairwise mutual trust relationships to exchange trust information, while trust relationships between users can also be established and propagated to the devices they own.

This deliverable reports on the design of the TMS implementation for the Cyber-Trust platform and its assessment, providing information regarding (a) the state-of-the-art models reviewed, (b) the architectural specification of the TMS and (c) the TMS evaluation, regarding the effectiveness under different attack scenarios. The operation of the TMS is regulated by a number of parameters, and the effect of these parameters on the formulation of trust levels is also examined.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Cyber–Trust project aims to develop an innovative cyber–threat intelligence gathering, detection, and mitigation platform to tackle the grand challenges towards securing the ecosystem of IoT devices. In this context, the concepts of *trust* and risk are central ones, being utilized in a multitude of functionalities: trust and risk levels are computed for devices in the defended system perimeter, and these scores are utilized by numerous components of the Cyber-Trust architecture to realize core platform functionalities, including intelligence response to threats and user/security officer alerting. The Trust Management Component (TMS) of the Cyber-Trust platform undertakes the task of computing the trust and risk scores for all devices, and appropriately making these scores available to other components of the Cyber-Trust platform.

1.1 Purpose of the document

This deliverable report reports on the design, implementation and validation of the TMS, as well as on its relation to the state-of-the-art in trust management systems, and more specifically:

- Which is the goal of the TMS and which are its functionalities?
- How are these functionalities accessible to other Cyber-Trust components?
- Which is the state-of-the-art in trust management and how does the Cyber-Trust TMS relates to it?
- Which specific algorithms are used for trust and risk score computation?
- What is the internal structure of the TMS?
- How can the TMS respond to different attacks typical in TMS systems and the IoT?

Furthermore, the trust and risk computation procedures necessitate data and information which are made available by other Cyber-Trust components. This deliverable covers additionally the interrelationships between the TMS and other components of the Cyber-Trust architecture.

1.2 Relations to other activities in the project

This document derives initially from D2.3, where the use case scenarios and the user requirements were surveyed in, as well as from D2.4/D2.6 where the first/final version of user requirements were recorded. Subsequently, D4.1/D4.4 presented the first/final version of the architectural positioning of the TMS within the Cyber-Trust platform and the TMS high-level architecture and the relevant key functionalities and key quality attributes. Finally, the state-of-the-art review performed in D5.1 and the initial version of the TMS design incorporated in D5.3 provided the basis on which the detailed design, implementation and evaluation presented in this deliverable were built. In the overall process of the design and implementation of the TMS, the legal and ethical recommendations catalogued in D3.3, as well as the results documented in D3.4 (data protection and privacy requirements; mitigation of risk related to the TMS development process) were closely observed.

This deliverable provides input for D8.2, which will elaborate the integration between individual Cyber-Trust components.

1.3 Structure of the document

The rest of this document is structured as follows: section 2 presents the trust management service, describing its functionality (both in terms of user requirements covered and Cyber-Trust use cases), its technological innovations with respect to the state of the art in trust management systems, a high-level overview of its architecture, and the APIs through which the TMS functionality can be used. Additionally, the technological stack used for the TMS implementation is presented, as well as prominent physical architectures for the TMS deployment.

Section 3 reiterates on the state of the art review performed in D5.1 [1], analysing the state of the art in trust management. In this context, trust management models and architectures, as well as trust management system implementations are examined.

Section 4 elaborates on the design of the TMS: firstly, the generic trust model employed by the TMS is presented, followed by an in-depth presentation of the trust computation algorithm. The section concludes with a detailed illustration of the TMS architecture.

Section 5 presents the attack scenarios against which the TMS will be evaluated, while section 6 presents the tuning, evaluation and validation of the TMS considering the attack scenarios. Finally, section 7 concludes the report.

2. Trust management service

2.1 Overview / objectives

The objective of the trust management service [A05, A08] is to serve an authority within the Cyber-Trust architecture which undertakes the following tasks:

- Consolidates observations on the status, behaviour and associated risk of devices into a comprehensive trust score, which indicates the degree to which each device is deemed to be trustworthy.
- Can be queried by other Cyber-Trust entities to provide the abovementioned assessments, for the perusal of the entities. Indicatively, trust assessments can be used for the visualization of trust within the network, for making decisions whether actions originating from or being directed to some device should be allowed or not, for raising alerts to security officers and so forth.
- Provides timely notifications to other entities within the Cyber-Trust platform to alert them of noteworthy events related to the level of trust associated with devices. In particular, demotions of device trust level below some threshold and the restoration of previously demoted trust of devices are emitted, allowing relevant components of the Cyber-Trust platform to take appropriate actions, such as enabling or disabling defence mechanisms.

2.2 Functionality coverage

2.2.1 Related requirements

The TMS is involved in a number of scenarios of the Cyber-Trust platform, where the trust level of one or more devices needs to be reassessed or consulted. In more detail, the TMS is involved in the following scenarios:

- *Monitoring and vulnerability assessment:* when a device is found to deviate from normal behaviour (or return to it after a period of deviation), or be vulnerable to new threats, the TMS triggers the recomputation of the device's trust level.
- *Network-level attacks*: when a network-level attack is identified, the TMS exploits the information provided by the iIRS to adjust the trust value of involved devices.
- *Device-level attacks*: Similarly, when a device is involved in some attack, the TMS arranges for recomputing the trust level associated with the device.

These user scenarios have co-shaped a number of functional and non-functional requirements. The relevant functional requirements are described in Table 2.1, while the associated non-functional requirements are described in Table 2.2.

Table 2.1. Functional requirements and use-case references for the TMS

REF_ID	Description of implementation	Use Case
FR9	**Requirement:** Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) before the identification of abnormal behaviour (e.g. cyber-attack) **Implementation:** The TMS underpins this requirement by providing the trust level of the device to the visualization module.	UCG-05-07, UCG-05-05

FR10	**Requirement:** Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) during abnormal behaviour (e.g. cyber-attack) **Implementation:** The TMS underpins this requirement by providing the trust level of the device to the visualization module. The trust assessment is updated synchronously as new data are received by the TMS, therefore the visualization will reflect the evolution of the trust level.	UCG-05-07, UCG-05-05
FR11	**Requirement:** Every device connected to the Cyber-Trust platform has visual representation of the Trust level (scoring) after the mitigation of any abnormal behaviour (e.g. cyber-attack). The TMS underpins this requirement by providing the trust level of the device to the visualization module. The trust assessment is updated synchronously as new data are received by the TMS, therefore the visualization will reflect the evolution of the trust level. **Implementation:** The TMS underpins this requirement by providing the trust level of the device to the visualization module.	UCG-05-07, UCG-05-05
FR21	**Requirement:** The user will be informed for the importance of the alert based on the overall Score of the device (it will be derived based on the abnormal behaviour, detected vulnerabilities etc.) **Implementation:** The TMS sends notifications when the trust level of device is demoted beyond a certain threshold or restored. These notifications may be exploited by other components, notably visualization and user notification modules, to appropriately convey the information to the user.	UCG-06-01, UCG-06-02, UCG-13-01, UCG-16-03
FR69	**Requirement:** The administrator (Trust DB) will be able to update the Trust score of a device manually. The update will include at least three options: Change status, Delete, Take offline. Field for additional information will be provided (e.g. comments). **Implementation:** A relevant API is provided, allowing authorized users to explicitly set the trust level of the device. Explicitly set trust levels are not directly modified by the trust score update procedure, however major discrepancies between explicitly set and computed scores will raise alerts.	UCG-10-04
FR73	**Requirement:** The user will be able to request (through the UI) the trust level of specific device(s) **Implementation:** The TMS provides an API through which authorized entities can retrieve the trust score of a device.	UCG-13-01
UP_FR8	**Requirement:** For each device users are going to visualise the reason for a certain Trust Level Score. **Implementation:** The TMS API will return, upon request, the base data that contributed to the shaping of the reported trust level.	UCG-13-01, UCG13-02

Table 2.2. Non-functional requirements and use-case references for the TMS

REF_ID	Description of implementation	Use Case
NFR43	**Requirement:** Prioritization of cyber-threats: the threats are ordered in descending order of their score. The score will derive based on vulnerability and impact attributes (technical impact, exploitability etc.) **Implementation:** Stems directly from the implementation of the use case.	UCG-16-04
NFR21	**Requirement:** Creation of the Trust DB **Implementation:** Instructions and/or automations for creating the TrustDB will be provided.	-
NFR22	**Requirement:** Trust DB will store records only hashed data **Implementation:** Data that are primarily stored in other databases will be maintained as hashes with relevant pointers.	UCG-04-01

2.2.2 Related use cases

Table 2.3 lists the use cases related to the TMS and the provisions made by the component to support the fulfilment of them.

Table 2.3. Use-cases related to the TMS

REF_ID	Description of implementation
UCG-10-05	**Use case:** Manually curate device profile **Implementation:** The TMS provides an API through which device trust scores can be explicitly set.
UCG-13-01	**Use case:** Retrieve trust level from TMS **Implementation:** Trust levels are computed by the TMS as relevant events occur and stored in the trust database. The trust database realizes an API through which authorized entities can retrieve the trust level assessments, either for a single device or for a bulk of devices.
UCG-13-02	**Use case:** Compute device trust level **Implementation:** The TMS intercepts notifications sent by other Cyber-Trust platform components, and exploits the information therein to compute the trust level. The notifications are received through the Cyber-Trust system message bus.
UCG-15-02	**Use case:** Compute device risk level **Implementation:** The TMS computes a new value for the risk level of a device. Information about the current device trust level, the current status of network attacks and network traffic related to the device (as compared with the baseline), the device vulnerabilities and their exploitability, the device health level and views of peer-level TMSs are taken into account to produce a comprehensive risk score.
UCG-16-04	**Use case:** Identify and prioritize cyber-threats

	Implementation: Distinct cyberthreats are considered and their total impact on the protected network and its resources is assessed, producing a per-cyberthreat score. Cyberthreats are then ordered in descending score order to produce the result.

2.3 Technology update

The Cyber-Trust TMS extends the current TMS paradigms and implementations by considering additional dimensions in the computation of the trust scores, notably the status of the devices and the associated risk. For the computation of the associated risk, the business value of assets can be considered where available. The TMS implementation will be able to adapt to its runtime environment: in resource-rich environments the full capabilities of the TMS will be included, which necessitate extensive computations and ample resources, while in constrained environments some features will not be realized, with the respective functionalities being consumed as services offered by corresponding, trusted, feature-rich installations.

2.4 Application architecture

Figure 2.1 illustrates the conceptual view of the Trust Management Service. Its architecture is designed to allow for exposing a coherent API, enabling any adaptation aspects to be implemented internally considering all the appropriate contexts (network & resource availability, situation criticality etc.). Reception of information needed to recompute the trust and risk scores - including device status, behaviour and associated risk aspects - are mainly intercepted through asynchronous messaging, through a dedicated communication channel, following the pub/sub paradigm. In this way, the TMS is decoupled from event producers and their timings; however, content consumption via APIs can be also used. Reciprocally, the TMS publishes events regarding notable changes of trust and risk levels, while also offering the same information under REST APIs. Adaptation, where needed, will be supported by an adaptation component to be developed and maintained separately from the computational aspects, promoting separation of concerns.

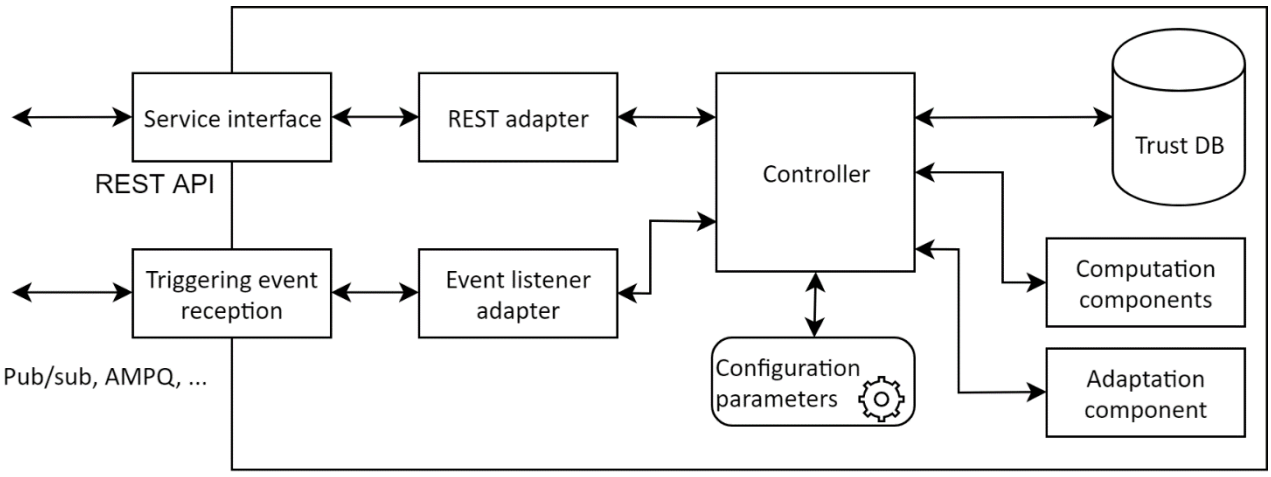


Figure 2.1. TMS high-level design

The overall high-level architecture of the TMS is depicted in Figure 2.1, while Figure 2.2 depicts the data view of the TMS, indicating:

(a) the data maintained internally in the TMS database;

(b) the messages that the TMS subscribes to in order to obtain the necessary information to compute trust and risk levels, as well as the sources of these messages, according to the overall Cyber-Trust architecture;

(c) the information that the TMS receives directly from the users (typically, through a UI);

(d) the messages that the TMS makes available to the asynchronous communication infrastructure, for the perusal of other Cyber-Trust components.

Trusted Peer TMS are curated directly by users. Users additionally provide information regarding other trusted entities in the platform: this pertains to modules that generate asynchronous messages to the information bus, and are expected to be consumed by the TMS. Each trusted entity specification provides the data needed by the TMS to verify the authenticity and integrity of received messages, i.e. the name of the peer and its certificate. While users are not commonly expected to be proficient with such data, automated procedures upon the setup of the platform are expected to relieve the user of the task of manually setting up this information. Should updates to this information be needed, automations, configuration assistants and wizards may also ease the task of the users, similarly to the case of Bluetooth device pairing, where the exchange of cryptographic data are masked beneath a simple PIN exchange [2].

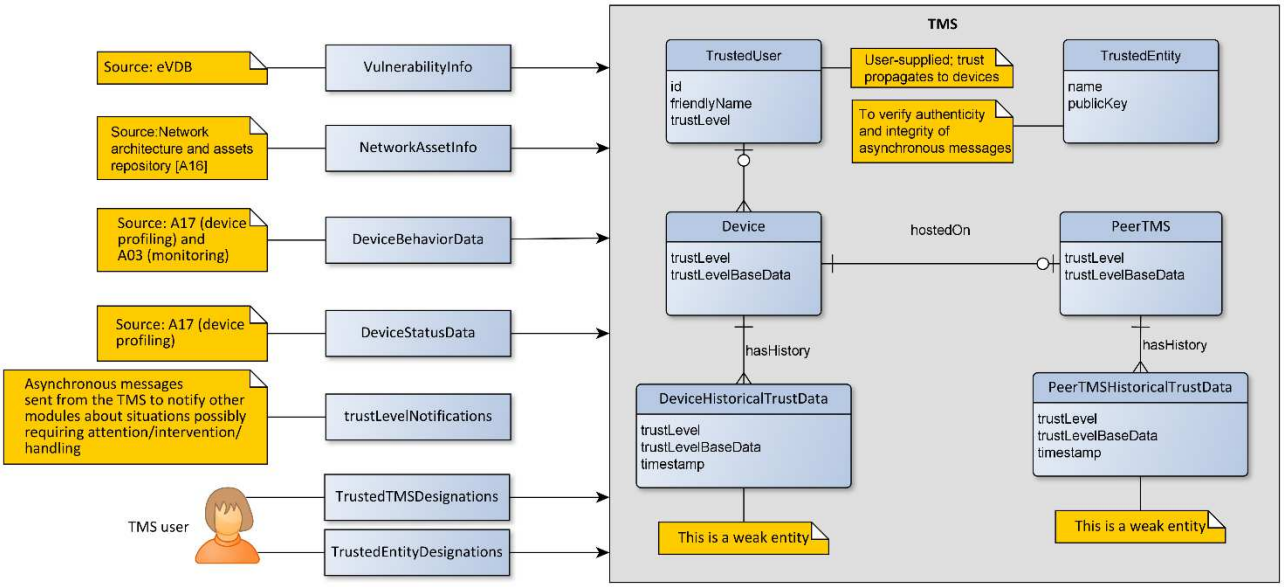


Figure 2.2. TMS data view

2.5 Application programming interfaces

The TMS exposes the REST APIs listed in the following subsections for direct invocation by other Cyber-Trust modules. As noted in subsection 2.4, and further elaborated in subsection 4.3, the TMS additionally employs a loose coupling communication pattern, through the exchange of messages via the message bus; the respective messages consumed through the message bus will be elaborated on in the context of WP8. The REST API presented in this section refines and extends the initial API design presented in D5.3 [3], mainly the incorporation of trusted entity management, which will facilitate the operation of the platform in a production environment, facilitating the dynamic incorporation of additional component instances from which the TMS may receive input.

2.5.1 REST APIs for managing device trust

Table 2.4 depicts the operations available for managing device trust, along with a brief description of each one.

Table 2.4. REST APIs for managing device trust

API URL specification	Description
GET /trust /info /{deviceId}	Returns the trust level for a device. The client may designate the desired trust dimensions. The information whether the reported trust level is explicit or implicit, is always returned.
PUT /trust/explicitLevel/{deviceId}	Explicitly specify the trust level of the device.
DELETE /trust/explicitLevel/{deviceId}	Delete the explicitly specified trust level of the device, returning to automatic computation.
GET /trust	Returns trust level for a set of devices. The client may designate the desired trust dimensions. The information whether the reported trust level is explicit or implicit, is always returned.

2.5.2 REST APIs for managing peer TMSs

Table 2.5 depicts the operations available for managing peer TMSs, along with a brief description of each one.

Table 2.5. REST APIs for managing peer TMS instances

API URL specification	Description
GET /peerTMS/{peerTMSId}	Returns information for a registered peer TMS
DELETE /peerTMS/{peerTMSId}	Deletes/unregisters a peer TMS.
PUT /peerTMS/{peerTMSId}	Creates or modifies a peer TMS.
GET /peerTMS	Returns information for a designated set of TMS.
GET /peerTMS/list/all	Returns information for all registered TMS.

2.5.3 REST APIs related to risk management

Table 2.6 depicts the operations available for risk management, along with a brief description of each one.

Table 2.6 REST APIs related to risk management

API URL specification	Description
GET /risks/prioritize	Returns the top risks, prioritized. The number of risks to return is described in the (optional) numRisks parameter. If missing, a default number is inserted.

2.5.4 REST APIs related to trusted user management

Table 2.7 depicts the operations available for trusted user management, along with a brief description of each one. Trust to users reflects on trust to the devices owned by them.

Table 2.7. REST APIs related to trusted user management

API URL specification	Description
GET /trustedUser/{trustedUserId}	Returns information about the designated trusted user.
DELETE /trustedUser/{trustedUserId}	Deletes/unregisters a trusted user.
PUT /trustedUser/{trustedUserId}	Creates or modifies a trusted user.
GET /trustedUser	Returns information for a designated set of trusted users.
GET /trustedUser/list/all	Returns information for all registered trusted users.

2.5.5 REST APIs related to managing trusted entities

Table 2.8 depicts the operations available for adding trusted devices; these operations enable the introduction of new trusted entities from which asynchronous messages can be received.

Table 2.8. REST APIs related to trusted user management

API URL specification	Description
POST /trustedEntity/	Adds information about a new trusted entity, specifying the name and the public certificate of the entity.

2.6 Technology stack

The technology stack and tools used for the implementation of the TMS are listed in Table 2.9. The technology stack has only slightly been modified since D4.4 [4] and D5.3 [3], through the addition of the Hibernate Object-Relational Mapping framework.

Table 2.9. Technology stack and applied tools used for the implementation of the TMS

Tool	Description
Swagger	Employed for prototyping the REST APIs of the TMS.
Java	The TMS functionality is coded in Java.
Spring framework	The Spring framework is employed to intercept and serve REST API requests.
MariaDB/MySQL	DBMS for managing the TrustDB.
Hibernate	For providing object-relational mapping, aligning the object-oriented view at the application level with the relational view at the database level.
Javax.Persistence	For managing database connections and persistent entities.
AMPQ/Asynchronous message protocol	For realizing pub/sub-based communications.

2.7 Physical architecture

In terms of physical architecture, the TMS is deployed as a single VM, running both the TMS and the data store (MariaDB/MySQL).

Furthermore, the following deployment options exist, to best suit the particular characteristics of the deployment platform/context:

1. run the TMS as one single Docker container, running both the TMS and the data store exists. Taking into account that Docker containers are ephemeral, provisions should be made upon deployment to map the filesystem of the Docker container that holds the data to stable storage.
2. run the TMS is deployed as one single Docker container, running both the TMS and the data store exists. Again, provisions should be made upon deployment to map the filesystem of the Docker container that holds the data to stable storage.
3. The TMS is deployed two interoperating Java application within a non-virtualized environment. This option is expected to be used (a) in environments not supporting virtualization and (b) in restricted environments where the overhead introduced by virtualization is not tolerable.

2.8 User Interface

The TMS runs as a service in Cyber-Trust platform and therefore it does not provide a dedicate own user interface (UI). However, certain UI elements are included in Cyber-Trust platform (e.g. information about the trust score of devices) to allow meaningful information to be provided to the user.

3. State of the art review

In this section we review the state of the art in the domain of trust management for the implementation of security, with a special focus on IoT environments. More specifically, we review both models and architectures for trust management (subsection 3.2) as well as concrete implementations (subsection 3.3). Models and architectures provide the generic context and mode of operation of trust management systems, whereas TMS implementations are examined for possibility of existing code reuse. Before models, architectures and implementations are presented, a brief overview of trust management concepts and dimensions is given in subsection 3.1. Subsection 3.4 draws conclusions and reviews aspects considered in the design and implementation of the TMS, taking into account the overall operation of the TMS in the Cyber-Trust environment.

3.1 Trust management concepts and dimensions

The concept of Trust management has emerged as a solution to the issue of managing and controlling access to information hosted in contexts where it is not feasible or desirable to use traditional authentication and authorization methods to that effect. This is particularly true in the IoT domain, where the number of devices render a case-by-case definition of access control specifications extremely laborious and practically infeasible, whereas the dynamic nature of the device population and the device-to-device interactions do necessitate an approach based on automated management, since human-mediated approaches would lead to non-timely and/or inaccurate specifications.

Trust management is defined in [5] as an aid the automated verification of actions against security policies. According to this definition, an action is allowed if the credentials presented are deemed sufficient, without the need to state or verify the actual identity of the interacting party; in this respect, symbolic representation of trust is separated from the actual person (or the person's digital agent). Later research has replaced the examination of credentials (which could be considered as *pseudonymized identities*, limiting hence the benefits of introducing trust management [6]) to the examination of a *set of properties*, which can be proven by an interacting party through the presentation of a set of digital certificates [7]–[9]. Under this scheme, the original set of trust management system elements identified in [6] is modified as follows:

1. Trusted unconditionally.
2. *Trust-related properties*, which represent aspects of interacting parties that are relevant to the application of security policies; typically, such properties are examined as antecedents of rules that comprise a security policy. Trust-related policies are safeguarded through digital signatures or other prominent means.
3. *Trust relationships*, which are a special kind security policy.

Note that the interaction between peers may involve multiple interacting parties, extending beyond the paradigm of a service/information requestor submitting a request to a server and receiving a reply. Under this view, a decentralized model is more prominent. In such a model, trust-related properties are generally provided and testified for by third parties, in the same fashion that TTPs provide digital certificate validation services. Furthermore, the computation of trust may be based on information collected from disparate sources within the system; this information may include trust-related properties, trust relationships, policies or trust assessments made by other peer systems. Notably, the sources of information bear themselves a level of trust, which should be taken into account in the information gathering and exploitation process.

Similar to the concept of security, the concept of trust between entities is a composite notion involving multiple aspects, including:

(a) the *status of each entity*, spanning across the entity's health level (especially whether the firmware, operating system kernel and critical configuration files are tampered with or integral; and whether the software operating on the system contains vulnerabilities or not).

(b) any *security controls* that are in effect and may limit the exploitability level of vulnerabilities or the impact that the exploitation of such vulnerabilities may have. Such security controls may be firewalls, IDS/IPS, backup procedures etc. [10]
(c) the *behavioural aspects* of the entity, which includes (i) the inspection whether the entity has been involved in malicious activities, (ii) the examination whether the entity complies with a prescribed behaviour which is known to be benign and (iii) the inspection whether the entity exhibits "normal" behaviour, i.e. behaviour that is known to be within the entity's operation history, or is found to exhibit exceptional behaviour.

Finally, contemporary attack methods entail complex multi-stage, multi-host attack paths, where each path represents a chain of exploits used by an attacker to break into a network [11]. Attack graphs enable the comprehensive risk analysis within a network, considering cause-consequence relationships between different network states; furthermore, the likelihood that such relationships would be exploited can be also taken into account [12].

The implementation of the Cyber-Trust TMS employs a trust- and risk-based approach to security, within which trust establishment and risk assessment encompasses all the above listed aspects, providing thus a comprehensive trust management and risk assessment view.

3.2 TMS models and architectures

Trust management models target at enabling nodes that participate in the trust management system to determine a trust metric value for other nodes within the system. Approaches to how trust models approach trust computation vary regarding numerous aspects, including the input used to compute trust, the way that trust values are updated, the consensus sought for trust value computation, the scale at which trust is measured, their resilience against attacks and so forth. Furthermore, trust management models vary with respect to architectural paradigm they follow, i.e., the way that the components participating in the trust management system are deployed in the target network, the relationships between the components and the information flows.

In the following subsections we survey existing trust models and their architectures, commenting on their merits and demerits.

3.2.1 Review of existing trust models

This section overviews the trust models that have been proposed by the literature trying to find an effective and efficient trust computation method. In service-oriented networks, an IoT device acting as a service requester needs a way of evaluating which of its peers can be trusted to provide it with the requested service, while taking into consideration the energy demands of carrying out such evaluation. This is the challenge that trust management models are aiming to solve. We present trust management models as seen in the literature and we categorize each model by trust dimensions, resiliency against certain attacks and qualitative characteristics.

3.2.1.1 Trust dimensions

Trust models are composed of several trust dimensions which can vary between them depending on the approach followed. In this section we present the five most essential trust dimensions, namely, trust composition, trust propagation, trust aggregation, trust update and trust formation [13].

Trust composition. Refers to the components the given model takes into consideration. The components include Quality of Service (QoS) and Social trust.

- QoS trust refers to the evaluation of a node based on its capability to deliver the requested service. It is considered as the "objective" evaluation of trust. In order to compute QoS trust, models use various trust properties including competence, cooperativeness, reliability, task completion etc.

- Social trust refers to the social relationship between owners of IoT devices. Social trust is used in systems where IoT devices must not be evaluated only on a QoS basis but also on a social basis, which is the device's commitment and willingness to cooperate. It can also be derived from similarity of devices. Social trust properties include connectivity, honesty, unselfishness etc.

Trust propagation. Refers to the way trust values are disseminated between entities. In general, there are two approaches, namely distributed and centralized.

- In distributed trust propagation, each device acts autonomously by storing trust values and disseminating them as recommendations to other devices as needed.
- In centralized trust propagation a central entity exists, which is responsible for storing trust values of the monitored network and disseminating them as needed.

Trust aggregation. Refers to the computation techniques used by a model to combine trust obtained from direct observation with indirect trust coming from recommendations. Main aggregation techniques include weighted sum, Dempster-Shafer theory, Bayesian inference, fuzzy logic and regression analysis.

- Weighted sum is a technique where weights are assigned on the participating values either statically or dynamically. For example, one model could use a trust property, e.g., competence, in order to assign higher or lower weights.
- Dempster-Shafer theory is based on belief function and plausible reasoning. It is a framework for reasoning with uncertainty related to other frameworks like probability, possibility and imprecise probability theories.
- Bayesian inference considers trust to be a random variable which follows a probability distribution. It is a simple and statistically sound model.
- Fuzzy logic uses approximate reasoning meaning that it doesn't use a binary evaluation variable but rather a variable whose values range between 0 and 1 for example, or even linguistic limits like High and Low which are translated using a membership function.
- Regression analysis is basically a prediction model which predicts the probability of an event happening or not happening (binary). In trust computing it is used to estimate relationships between environmental conditions, e.g., how much computing resources a node needs, which are treated as variables and used to predict the trustworthiness of an object.

Trust update. Describes when trust values are updated. There are two approaches: event-driven and time-driven.

- Event-driven is the approach in which trust values are updated when an event occurs.
- Time-driven is the approach in which trust values are update periodically.

Trust formation. Refers to how the overall trust is formed out of the trust properties considered. Trust can be formed by considering only one trust property (Single-trust) or many properties (Multi-trust).

- Single-trust is when only one property is taken into consideration when computing trust and it is usually a property of QoS. It is considered as a narrow approach because trust is multi-dimensional, but it is useful in cases with limited resources.
- Multi-trust is the multi-dimensional approach in computing trust, because it uses more than one trust properties to form the overall trust evaluation of a device.

We also used the following properties to classify the trust managements models: [14]

Trust scaling. Trust is represented by either discrete or continuous numerical values.

- *Binary discrete values*: Represented with 0 or 1, distrust or trust respectively.
- *Multinomial discrete values*: Sometimes binary values are not sufficient, so more scaled metrics are used, e.g., "very trust", "trust", "distrust", and "very distrust".
- *Continuous value*: For example [0,1].

- *Interval*: Instead of using one value to represent trust, an interval is used in order to introduce the uncertainty property of trust.

Semantic meaning. In different models and scenarios trust can have various semantic meanings. Some semantic meanings include:

- *Evidence- or experience-based trust*: Trustors build their trust based on their own observations and past interactions. This can be done using probabilities, mean average, mode average or difference.
- *Application-specific behaviour-based trust*: This means that trust is calculated based on specific monitored behaviours.
- *Similarity-based trust*: This approach assumes that devices that are similar to each other, will probably trust each other.
- *Reputation*: Reputation is a type of trust which isn't relative to the trust between two specific devices but instead each device has a trust value representing how much it is trusted by the whole community.
- *Fuzzy logic-based trust*: Trust is considered to be nondeterministic and because of this, fuzzy logic is suitable for evaluating it.
- *Comprehensive trust*: Many approaches take into consideration trust as seen in human relationships. In this case, trust is seen as a sum of complex human interactions. On this basis, social metrics are introduced in trust evaluation, like social similarity, social contact, friendship, etc.

Trust inference. In IoT networks, nodes are not always directly connected with another and in these cases trust evaluation cannot be done by direct observation. Therefore, trust recommendations are introduced. There are two operators to be considered for trust inference: transitivity and aggregation:

- Transitivity operator refers to the way the recommendations are combined by building a transitivity "chain" to the trustee node and it is based on the transitive relation from mathematics.
- Aggregation operator refers to the way the recommendations are combined to calculate the overall indirect trust.

Table 3.1. Overview of different trust models

Model	Composition		Propagation		Aggregation			Update	Formation	
	QoS	Social	Distrib	Central	Weigh	Fuzzy	Bayes	E/T	Sin	Mul
[15]–[18]	X	X	X		X			E/T		X
[19], [20]	X	X	X		X		X	E/T	X	
[21]	X		X		X	X		T	X	
[22]	X		X		X	X		T	X	
[23]	X		X		X			E	X	
[24]	X			X	X			T	X	
[25]	X	X	X	X	X			E		X
[26]	X			X	X			E/T	X	
[27]	X			X	X			E/T	X	
[28]	X		X		X			T	X	
[29]	X		X	X				T	X	
[30]	X		X	X	X			E	X	

[31]		X	X	X	X	X		E		X

3.2.1.2 Trust-based attacks

Inside an IoT network, every node wants to have a high trust value. A high trust value means the node will be selected more times over nodes with lower trust value, thus increasing their gains and influence over the network. Malicious nodes will try a variety of attacks in order to gain more trust among their peers. There are a lot of attacks that can be executed in an IoT network, such as, jamming attacks, replay attacks, eavesdropping attacks, DoS attacks, etc. However, there are some attacks that are especially used to disrupt trust and reputation systems. These attacks fit better into the scope of this work and the most common ones are shortly presented below [31].

- **Self-promotion attacks (SPA)** [13]**.** The malicious node provides good recommendations for itself.
- **Bad-mouthing attacks (BMA)** [13]**.** A malicious node provides bad recommendations for a "good" node in order to decrease its trust value and probability of being chosen as a service provider.
- **Ballot-stuffing attacks (BSA)** [13]**.** A malicious node boosts the trust of another malicious node in order to increase the possibility of the malicious node being chosen as a service provider.
- **Opportunistic service attacks (OSA)** [13]**.** When the trust of a malicious node starts dropping, it starts acting as a "good" node in order to regain its trust.
- **On-off attacks (OOA)** [13]**.** A malicious node is behaving randomly, sometimes performs well sometimes bad, so that it won't get labelled as malicious.
- **Whitewashing attacks** [14]**.** When a malicious node has very low trust, it discards its identity by leaving the network and re-entering it.
- **Discriminatory attacks** [15]**.** A malicious node attacks non-friends or nodes without strong social ties.
- **Sybil-Mobile attacks** [18]**.** A malicious node creates one or more fake identities in order to manipulate recommendations, promote itself and gain influence over the network.
- **Selective Behaviour attacks** [27]**.** A malicious node is behaving well and bad between different services. For example, well for simple services, but bad for more complex ones.

Table 3.2. Overview of trust-based attacks

Attack Resiliency	SPA	BMA	BSA	OSA	OOA	White-washing	Discrimi-natory	Sybil-Mobile	Selective Behaviour
[16]	X	X	X						
[15]	X	X	X			X	X		
[17]	X	X	X						
[18]	X	X	X					X	
[19]	X	X	X	X					
[20]	X	X	X	X					
[21]	X								
[22]									
[23]					X				
[24]									
[25]	X	X	X	X					

[26]									
[27]	X	X	X		X				X
[28]									
[29]									
[30]		X							
[31]									

3.2.1.3 Trust management models

In this section we survey the different trust models proposed in the literature. For each model, the approach adopted for trust computation is presented, while salient features of the models are summarized in Table 3.3, within subsection 3.2.1.4 below.

Bao, 2012 [16]**.** This model considers Community of interest (Coif) based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner keeps a friends list. Nodes belonging to similar communities are more likely to have similar interests or capabilities. Both QoS and Social trust composition are considered, including three trust properties: honesty (QoS), cooperativeness (QoS) and community-interest (Social); please refer to Table 3.3 for further details. The trust value is a real number in the range [0,1] where 1 indicates complete trust, 0.5 ignorance, and 0 distrust. The trust values can occur from direct observations and when such observations are not available, from recommendations. It follows a distributed scheme, while for trust aggregation the weighted sum technique is used. It is worth mentioning that the weights that were used for past experiences can be dynamically adjusted when new evidence occurs to rebalance the trust convergence rate and trust fluctuation rate. In the simulation results, the effect that changing weights have is observed, but a way to dynamically adjust them is not mentioned.

Chen, 2016a [15]**.** This model is very similar to Bao, 2012. Main differences include: 1. A general approach for the computation of overall trust is not discussed. Instead, overall trust computation for specific scenarios is discussed. 2. The friends (nodes) lists exchanged between nodes upon interaction are encrypted with a one-way function in a way that nodes can identify only common friends. Hashing is cost-efficient. 3. The model is tested in two real-world scenarios, namely, “Smart City Air Pollution Detection” and “Augmented Map Travel Assistance”.

Bao, 2013 [19]**.** This model considers Community of interest (CoI) based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner keeps a friends list. Nodes belonging to similar communities are more likely to have similar interests or capabilities. Both QoS and Social trust composition are considered. The trust value is a real number in the range [0,1] where 1 indicates complete trust, 0.5 ignorance, and 0 distrust. The trust properties considered are honesty, cooperativeness and community-interest; please refer to Table 3.3 for more details. The trust propagation is distributed. For trust aggregation, Bayesian inference is used for direct trust and weighted sums are used to aggregate recommendations into indirect trust. Most importantly, this model introduces an efficient storage management strategy suitable for large-scale IoT systems.

Chen, 2016b [20]**.** This model is an extension of Bao, 2013 [19]. Extensions include: 1. In the evaluation of recommenders, it introduces two additional properties, namely, friendship and social contact, which are further analysed in Table 3.3. 2. In trust aggregation it combines the direct with the indirect trust to form the overall trust. 3. Its simulations outperform Eigen Trust [32] and PeerTrust [33] in trust convergence, accuracy, and attacks resiliency.

Chen, 2011 [21]**.** This model considers only QoS metrics for evaluating trust, namely, end-to-end packet forwarding ratio (EPFR), energy consumption (EC), and package delivery ratio (PDR). Each node maintains a data forwarding transaction table which includes the values: 1. Source: the trust and evaluation evaluating nodes, 2. Destination: the evaluated destination nodes, 3. $RF_{i,j}$: the times of successful transactions made between nodes i and j, and 4. $F_{i,j}$: positive transactions. It follows a distributed scheme in terms of trust

propagation. In trust aggregation, a fuzzy trust model is used, and the overall trust is formed using a weighted sum of direct and indirect trust based on recommendations. The direct trust is computed by first aggregating the aforementioned QoS metrics, then labelling the results as a positive or negative experience based on a threshold and then a fuzzy membership function computes the direct trust based on the number of positive and negative experiences. Additionally, the model was tested on simulations and achieved better performance from the Bio-inspired Trust and Reputation System for Wireless Sensor Network (BTRM-WSN) [34] and the Distributed Reputation-based Beacon Trust System [35] in both packet delivery ratio and detection probability of malicious nodes.

Mahala, 2013 [22]**.** This model considers three QoS metrics: Experience (EX), Knowledge (KN) and Recommendation (RC) ratings. It follows a distributed scheme, as every device considers the ratings of its neighbours for the calculation of the trust score. Trust is calculated periodically using Mamdani-type fuzzy rules (representing If-Then relationships between their input variables) from the linguistic values of the three aforementioned metrics. Trust scores (as linguistic values) are then mapped to a set of access control permissions. Experience (EX) is the weighted sum of a number of previous interaction ratings between two devices (+1 for a successful interaction and -1 for an unsuccessful interaction), Knowledge (KN) is the weighted sum of direct and indirect knowledge ratings, and Recommendation (RC) is the weighted sum of RC ratings from a number of devices about the device to be trusted. The three metrics are mapped to their linguistic variables using predefined numeric (crisp) ranges. The model was tested in a simulated environment of wireless sensors with communication between sensors being controlled by trust ratings, resulting in more energy efficient communications, and proving to be scalable.

Prajapati, 2013 [28]**.** This model considers the service satisfaction at a given time from a specific service provided by a node (a QoS property). Trust is defined as: the Direct Trust value, the Recommended Trust value if the node calculating the trust value had no interaction with the rated service/node and thus the Direct Trust value can't be calculated, or as a predefined Ignorance Value if the rated node is joining the cloud environment for the first time. Direct Trust is defined as the weighted sum of the rated service satisfaction ratings over time (with the weights decreasing over time, thus favouring newer ratings). Recommended Trust is defined as the weighted sum of the Direct Trust values of the other nodes. The weights assigned for each Direct Trust value are calculated using the number of positive interactions between the node calculating the trust value and the node whose rating is considered in the weight calculation, and the Satisfaction Level –a factor dependent on availability, recovery time, connectivity and peak-load performance as defined in the service agreement. All nodes maintain a Direct Trust Table and a Recommended Trust Table containing the respective trust values with both tables being updated periodically. This model follows a distributed model as in the case of Recommended Trust, the trust values of all network nodes are considered.

Saied, 2013 [27]**.** This model considers ratings given to a specific node and service at a given time while also taking into consideration its state (e.g., age, resource capacity, etc.). It follows a centralized scheme with a Trust Manager (TM) node receiving reports from the network and calculating the trust values on demand. This leads to reduced communication overheads: a) since trust values are calculated and transmitted on demand, less memory usage for each node; b) since the trust values can be requested again from TM, and thus being energy efficient. The model operates in five phases: 1) TM receives reports from the network nodes, 2) TM calculates the trust values of a number of candidate nodes and sends a list of trustworthy nodes to the requesting node, 3) the requesting node receives the list and interacts with a chosen trustworthy node, 4) the requesting node rates the service provided by the chosen trustworthy node and sends the rating to the TM, and finally 5) TM updates its trust values accordingly. Trust is calculated as the weighted average of the scores given to a node while taking into consideration the reputation of the node providing the score, the contextual similarity of all the reports concerning the same node, and the age of the report –favouring the most recent reports. Contextual similarity is calculated from the node capabilities between two nodes –to locate similar nodes, and/or from the difference of required resources between two services –to locate nodes able to run a similar service. Initially all nodes of the network are deemed trustworthy.

Mendoza, 2015 [23]**.** This model is a distributed version of the model proposed by Saied et al. [27]. It is noted that centralized schemes may not be suitable for IoT systems as server installation and server costs may be prohibitive. The model considers ratings given to a specific node and service. The model operates in three

phases: 1) every node announces its presence to its neighbours while also keeping a record of its neighbours, 2) a node requests a service from a neighbouring node and rates the response as positive or negative, and 3) the node calculates and stores the trust value of its neighbour. The response rating is defined as the fixed value of the provided service weighted by an adjusting factor, with the negative response rating being equal to two times the positive response rating. The provided service value is proportional to the processing requirements of the service, as more processing power or energy is required to run a service the higher the service value will be. The trust value of a node is calculated as the sum of all interaction ratings. The model was tested against On-Off Attacks (OOA) and it is noted that a large number of neighbours can cause delays in the assignment of the maximum distrust score to the malicious nodes.

Namal, 2015 [24]**.** This model considers four parameters: availability of resources to its users, reliability of produced information, response time irregularities, and capacity. It follows a centralized scheme with a Trust Manager (TM) module, hosted on the cloud, receiving filtered data from Trust Agents (TA) distributed on the network which in turn receive raw data and monitor the state of the network nodes. The TM implements a Monitor, Analyse, Plan, Execute, Knowledge (MAPE-K) feedback control loop and calculates the trust using the weighted sum of the trust parameters for all parameters considered. The trust parameter is also a weighted sum of the current value and the previous value calculated. This model shows advantages in: availability and accessibility –as the TMS is hosted on the cloud and is accessible from the internet, scalability –as the TMS utilizes TAs filtering the raw data, and flexibility –as the TAs can be deployed in a flexible manner.

Khan, 2017 [26]**.** This model considers ratings given to a node by its neighbours, these ratings are the combination of three variables: belief, disbelief and uncertainty –as defined in Jøsang's Subjective Logic. This model is proposed as part of an extension of the RPL routing protocol [36], utilizing the proposed model to isolate malicious nodes. It follows a centralized scheme with a central node (e.g., RPL border router or cluster-head) calculating trust values for all network nodes and deciding to isolate malicious nodes. Each node of the network is assumed to be able to detect and therefore rate the performance of its neighbouring nodes; each of the three aforementioned variables is defined as follows: belief is the number of positive interactions divided by the total number of interactions and a constant *k*, disbelief is defined similarly but instead of the positive interactions the number of negative interactions is used, and uncertainty is also defined similarity but with the constant k used instead of the number of positive/negative interactions. The central node calculates the trust value of each network node by combination of the trust values regarding the node to be trusted and using a threshold the central node isolates malicious nodes from the network.

Djedjig, 2017b [37]**.** This model considers two QoS parameters: selfishness and energy, and one social parameter: honesty as ratings given about a node from its neighbours. This model is a proposed extension of the RPL routing protocol, as in Khan et al. [21], to isolate malicious nodes. It follows a distributed scheme with each node calculating the trust values of its one-hop neighbours while also considering the trust values of its one-hop neighbours. Trust calculation is performed as follows: 1) each node calculates the direct trust values of its one-hop neighbours as a weighted sum of the honesty, energy and unselfishness metrics (definitions of which are not discussed in detail) with each metric being the weighted sum of the current value of the metric and the previous value of the metric, 2) each node receives the direct trust values calculated by its one-hop neighbours concerning the node to be rated, and 3) the indirect trust is then calculated by each node as the average of the direct trust calculated by the node itself and its neighbours. All nodes are assumed to be equipped with Trusted Platform Module (TPM) chips.

Medjek, 2017 [13]**.** This model is based on the one proposed by Djedjig et al. [37] with the difference in the metrics considered: honesty, energy and mobility. The main difference is the network architecture as this model applies to RPL networks consisting of a Backbone Router (BR) that federates multiple 6LoWPAN networks, each consisting of a 6LoWPAN Border Router (6BR) connected to the BR and the rest of the network nodes. This model follows a distributed scheme with each network node calculating the trust of its one-hop neighbours, as in [37], with the added steps of notifying its 6BR if a node is found to be untrustworthy and with the 6BR in turn notifying the BR of the malicious node. All nodes are assumed to be equipped with a Trusted Platform Module (TPM) and all nodes are registered with the BR at installation time, with every node having a unique ID assigned by the BR. Several lists are maintained by the various network nodes; the BR maintains two lists: one of potential malicious nodes and one of all nodes and their states; the

6BR maintains three lists: one of all 6BR area nodes, one of all the mobile nodes, and one of the potential malicious nodes; finally the remaining nodes also maintain three lists: one of potential malicious nodes, one of suspicious nodes and a copy of the mobile node list from the 6BR. Three modules operate on the various network nodes: IdentityMod controls access to the network and ensures that every node has a unique ID, MobilityMod ensures that both the BR and the 6BRs are aware of mobile nodes and of their status, and IDSMod is responsible for attack detection and mitigation. Trust is calculated in a similar fashion to [37] with the values of the honesty metric supplied by the IDSMod and the values of the mobility metric supplied by the MobilityMod; the three metrics are not discussed in detail.

Nitti, 2014 [25]**.** The two proposed models, subjective and objective, consider seven parameters: service ratings, number of transactions per node –to detect nodes with an abnormal number of transactions, node credibility, transaction factor –separating important transactions to avoid trust to be built solely by many small transactions, computation capacity –as "smarter" nodes can be better suited to become malicious, relationship factor –the type of relation between two nodes, and finally the notion of centrality –as a node with many connections or involved in many transactions takes a central role in the network.

The subjective model follows a distributed scheme where each node stores the necessary information to calculate the trust values locally. Two situations are covered relating to the social relationship between nodes: when the rating node has a social relationship with the rated node and when the two nodes have no direct social relationship. In the first situation trust depends: on the centrality of the rated node in relation to the rating node –by count of the common friends out of all the neighbouring nodes, the direct experience of the rating node –further defined as the weighted sum of both short-term and long-term opinions, and the indirect experience of the rating node's friends –defined as the weighted average of the trust values assigned to the rated node by the rating node's friends, weighted by their credibility. In the second situation, trust depends: on the opinions of the chain of common friends connecting the two nodes, again weighted by their credibility. Generally, after each transaction a rating (positive/negative) is given to the node providing the service and to the nodes whose opinion was considered in calculating the trust value. Negative recommendation ratings are given to both malicious nodes and to nodes in their neighbourhood, thus isolating the malicious nodes and their influence further.

The objective model follows a more centralized scheme where each node reports its feedback to special nodes, referred to as Pre-Trusted Objects (PTO), responsible solely for maintaining the distributed storage system, in this case a Distributed Hash Table (DHT) and more specifically one following the Chord architecture. Trust is calculated in a similar fashion as in the subjective model; node centrality is defined as the total number of transactions performed by the node to provide a service divided by the total number of transactions performed to either provide or request a service, and both short-term and long-term opinions consider the ratings of every network node weighted by their credibility. Nodes with few social relations, high computation capabilities and nodes involved in a large number of transactions between them are assigned low credibility, as they are more likely to become malicious.

Wu, 2017 [29]**.** The system model consists of four entities with three trust relationships among them. The four entities are defined: RFID tags, RFID readers, authentication centres and one administration centre, with the first three being grouped in domains. A domain has multiple RFID readers connected with the domain authentication centre which authorizes the readers to interact with the RFID tags, and the domain authentication centres are connected with the administration centre. The trust relationships of this system model are defined as: intra-domain trust –trust relationship between RFID tags and readers of the same domain, inter-domain trust –trust relationship between authentication centres, and cross-domain trust –trust relationship between RFID tags and readers belonging to different domains.

The trust management model consists of two layers: the authentication centre trust layer –a centralized trust management system managing the trustworthiness of authentication centres, and the reader trust layer –two proposed trust management schemes managing the trustworthiness of RFID readers. The RFID tags are always assumed to be trusted.

The first reader trust management layer scheme proposed uses the Dempster-Shafer evidence theory and consists of four steps: 1) the interaction of an RFID reader is recorded by its neighbours, 2) the neighbours

calculate the local trust values which are then transmitted to the authentication centre, 3) the authentication centre calculates the global trust of the RFID reader by using the Dempster knowledge rule, and finally 4) it the RFID reader is malicious or malfunctioning the administration centre is notified. Possible RFID reader interaction events are identified and marked as: malicious behaviour, malfunctioning behaviour and normal behaviour by the neighbouring RFID readers, each counting the number of events within a specified time frame. Using the number of recorded events, the neighbouring RFID readers can calculate the local trust value for each type of interaction events as: the number of events marked as malicious/malfunctioning/normal divided by the total number of recorded events. The final value of the local trust value is then chosen from the event-specific local trust values using a threshold. The authentication centre calculates the global trust of the RFID reader by aggregating the event-specific local trust scores calculated by the neighbouring RFID readers and then choosing the final integrated event-specific score using a threshold.

The second reader trust management layer scheme proposed, considers the fact that events may not be detected by neighbours of the RFID reader and thus the first reader trust management layer scheme may not be applicable to certain situations. Each RFID tag keeps record of the last interaction with an RFID reader, more specifically the RFID reader ID, a timestamp and the rating assigned to the RFID reader by the tag. This record is sent at the next time the RFID tag interacts with any RFID reader (and is then deleted from the RFID tag), with the RFID reader forwarding the record to its authentication centre which checks for abnormalities and if any problem arises, it notifies the administration centre as well as the authentication centre the previous RFID reader belongs.

The proposed authentication centre trust layer scheme considers abnormal event reports by RFID readers and affects the trust value of the domain authentication centre the readers are part of. Calculation of trust in this case can be performed by either of the two methods proposed for the reader trust management schemes.

Mahmud, 2018 [31]**.** This model considers three social trust metrics for a pair of nodes, namely: relative frequency of interaction, intimacy and honesty, and the deviations of generated data from the historical data of the node that generated the trust metric and its neighbours. Two trust dimensions are defined: node behavioural trust and data trust; both calculated by combination of direct (from the rating node) and indirect (from the rating node's neighbours) interactions, with indirect interactions being weighted by the distance of the neighbour to the rated node. Node behavioural trust is calculated using an Adaptive Neuro-Fuzzy Inference System (ANFIS), a fuzzy system using back propagation to tune itself. The three inputs to ANFIS are defined as: relative frequency of interaction is defined as the ratio of interactions with the rating node out of all interactions of the rated node in a given time period, intimacy is defined as the ratio of time amount spent interacting with the rating node out of the total time spent interacting with all nodes except the rating node, and honesty is defined as the ratio of successful interactions out of the total number of interactions of the rated node with its rating node. Three linguistic terms are used in ANFIS for each of the three inputs: Low, Medium and High. Deviations of generated data, used to calculate the data trust, are defined as follows: direct data trust is defined as the deviation of instantaneous data from the historical data generated by the rated node, and indirect data trust is defined as the deviation of instantaneous data from the historical data from the historical data generated by the rated node's neighbours.

Arabsorkhi, 2016 [38]**.** The work of Arabsorkhi et al. presents the general principle behind many proposed trust management models considering ratings given to network nodes for the quality of the services provided over a specific time period. If the rating node has enough information to determine the trust value from its own ratings over the specified time period (by direct observation) it can proceed to calculate the trust value of the node to be rated. If not, then the rating node can query the rest of the network and aggregate the trust values assigned by the other network nodes to the rated node.

Yuan, 2018 [30]**.** This model considers ratings given after node interaction for the quality of provided services. The network model consists of IoT edge nodes being part of a domain federated by an edge broker node, which in turn contact a central cloud server responsible for the final calculation of trust values. Three trust values are calculated: the direct trust about a device to another device (D2D direct trust), the feedback trust

about a node by an edge broker (B-to-D feedback trust), and the overall trust (the final trust value) about a device. D-to-D direct trust is updated and based on the history of direct interaction between nodes, it is defined as the ratio of positive interactions and the number of total interactions between the two nodes. B-to-D feedback trust is updated by the edge broker periodically and is based on all the D-to-D direct trust values concerning an edge node (except self-ratings); the edge broker aggregates the D-to-D direct trust values using weights derived by use of object information entropy theory, overcoming the limitations of assigning the weights manually. The overall trust value is calculated as the weighted sum of the D-to-D direct trust and the B-to-D feedback trust, thus considering the opinion of the rating node as well as the opinion of the whole network about the rated node.

3.2.1.4 Qualitative characteristics

Table 3.3 summarizes the qualitative characteristics of the surveyed trust models. The following characteristics are included in this summary:

- *Inference:* which mechanisms are employed for inferring trust values based on recommendations?
- *Trust scaling*: which is the range of the trust computation function?
- *Advantages*: which are the strong points of the model?
- *Complexity*: comments on space, time, processing, memory and communication complexity of the model.
- *Limitations*: aspects that constrain the effectiveness or the applicability of the model.
- *Monitored behaviour*: which activities and evidence are collected to support the calculation of the trust metric?
- *Trust metric*: lists the dimensions expressed within the trust metric, such as honesty, reputation etc.
- *Context*: refers to the environment for which the model has been developed for.
- *Semantic meaning:* lists how the approach to trust computation is interpreted at a high level of abstraction. For instance, some could be experience-based or reputation-based, while some others could be application-specific or application-agnostic. Note that multiple orthogonal dimensions can be involved here.

Table 3.3. Overview of qualitative trust characteristics

Model	Inference	Trust Scaling	Advantages	Complexity	Limitations	Monitored behaviour	Trust metric	Context	Semantic meaning
[16]	Multiplication for transitivity and weighted sum of trust values for aggregation.	Continuous [0,1].	Its trust-based service composition outperforms random service composition and approaches the maximum achievable performance from ground truth.	Node storage needed to keep trust values.	Hostility is considered to be increasing only over time in the simulations. When ground trust changes dynamically, recommendations don't contribute to convergence speed.	Honesty: estimated by keeping a count of suspicious dishonest experiences observed over a time interval using a set of anomaly detection rules such as high recommendation discrepancy as well as interval, retransmission, repetition, and delay rules. Cooperativeness trust: of node i towards node j is the ratio of the number of common friends over the number of node i's friends. Community-interest trust: of node i towards node j is the ratio of the number of common community/group interests over the number of node i's community/group interests.	Honesty, Cooperativeness and community-interest.	Community of interest (CoI) based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner keeps a friends list. Nodes belonging to similar communities are more likely to have similar interests or capabilities.	Comprehensive
[15]	Multiplication for transitivity and	Continuous [0,1].	Its trust-based service composition outperforms	Node storage needed to	The storage needs can be	Honesty: estimated by keeping a count of	Honesty, Cooperativene	Community of interest (CoI)	Comprehensive. Although a

	weighted sum of trust values for aggregation.		random service composition and approaches the maximum achievable performance from ground truth.	keep trust values. The storage cost per node is $O(N_T N_X)$, where N_T is the number of IoT devices and N_X is the number of trust properties.	excessive for IoT devices with limited memory space.	suspicious dishonest experiences observed over a time interval using a set of anomaly detection rules such as high recommendation discrepancy as well as interval, retransmission, repetition, and delay rules. Cooperativeness trust: of node i towards node j is the ratio of the number of common friends over the number of node i's friends. Community-interest trust: of node *i* towards node *j* is the ratio of the number of common community/group interests over the number of node i's community/group interests.	ss and Community-interest.	based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner keeps a friends list. Nodes belonging to similar communities are more likely to have similar interests or capabilities.	general approach for overall trust formation is not discussed.
[19]	Multiplication for transitivity and weighted sum of trust values for aggregation.	Continuous [0,1].	It introduces a storage management strategy suitable for large-scale IoT systems. Newly joining nodes can build their trust very quickly through available recommendations.	The storage management strategy is very efficient, "find medium, maximum and minimum operations have a	Trust recommendations can be biased when the recommender is from a different CoI.	Honesty: estimated by keeping a count of suspicious dishonest experiences observed over a time interval using a set of anomaly detection rules such as high recommendation discrepancy as well as	Honesty, Cooperativeness, Community-interest	Community of interest (CoI) based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner	Comprehensive.

			The simulations considering the limited storage method, where storage management was used, achieved similar performance level with the unlimited space simulation and even better trust convergence time.	complexity of O(1) by using the max-min-median heap and all other operations (find, insert, delete) can be performed in O(log(n)) time.	The case in which a new node joins when the systems hasn't converged yet is not tested.	interval, retransmission, repetition, and delay rules. Cooperativeness trust: of node *i* towards node *j* is the ratio of the number of common friends over the number of node i's friends. Community-interest trust: of node *i* towards node *j* is the ratio of the number of common community/group interests over the number of node i's community/group interests.		keeps a friends list. Nodes belonging to similar communities are more likely to have similar interests or capabilities.	
[20]	Multiplication for transitivity and weighted sum of trust values for aggregation.	Continuous [0,1].	It introduces a storage management strategy suitable for large-scale IoT systems. The weights for combining social similarities are adjusted dynamically and this leads to credible trust feedback and minimized trust bias. It outperforms EigenTrust [32] and PeerTrust [33] in trust convergence, accuracy, attacks resiliency.	The storage management strategy is very efficient, "find medium, maximum and minimum operations have a complexity of O(1) by using the max-min-median heap and all other operations (find, insert,	Only persistent attack patterns considered, i.e., malicious nodes perform attacks with a probability of 1 or whenever there is a chance. The determination of the optimal trust decay parameter by means of	User feedback (binary: satisfied/not satisfied). Friendship similarity: the cosine similarity of the two users' friends lists. Social contact similarity: cosine similarity of the two users' locations lists. Community of interest similarity (CoI): cosine similarity of the two users' devices lists.	User satisfaction based on service completion, Friendship, Social- contact, Community-interest.	Service oriented architecture (SOA) based social IoT (SIoT) systems. Devices have owners and owners have many devices. Each owner keeps a friends list. Nodes belonging to similar communities are more likely to have similar	Comprehensive.

			Simulations considering limited storage had same performance as the ones with unlimited storage.	delete) can be performed in O(log(n)) time.	convergence and accuracy trade-off based on environment conditions is left for future work.			interests or capabilities.	
[21]	Multiplication for transitivity and weighted sum of trust values for aggregation.	Continuous [0,1].	Fast convergence. The model reduces energy consumption caused by the presence of malicious nodes. Better performance from BTRM-WSN [34] and DRBTS [35] in both packet delivery ratio and detection probability of malicious nodes.	-	-	End-to-end packet forwarding (EPFR): the ratio between the numbers of packets received by the destination nodes to the number of packets sent by the source node. Annual energy consumption (AEC): the nodes' energy consumption ratio. Package delivery ratio (PDR) calculated by packet loss and packet retransmissions.	End-to-end packet forwarding (EPFR), Energy consumption (AEC), Package delivery ratio (PDR).	Wireless sensor networks of IoT and cyber-physical systems (CPS). Highly dynamic topology.	Fuzzy logic-based trust.
[22]	Multiplication for transitivity and fuzzy membership function mapped	Continuous [-1,1] mapped to three linguistic values "Low",	The framework is scalable in terms of number of nodes and number of trust linguistic terms,	-	-	Experience (EX) metric calculation is based on past interactions. When interaction is successful it has a value of +1 and a	Experience (EX), Knowledge (KN),	A fuzzy trust-based access control (FTBAC) framework for IoT is discussed.	Fuzzy logic-based trust.

	to three linguistic values which are in turn used to create rules that form overall trust	"Average" and "Good"	without affecting performance. The simulations show that energy consumption is less in access control with FTBAC than without. Furthermore, residual energy is higher in access control with FTBAC than without.			value of –1 otherwise. The final value is relative to past interactions values sum. Knowledge (KN) is calculated based on "direct and indirect knowledge" but the monitored behaviour to obtain these values is not discussed. Recommendation (RC) metric is calculated with the use of RC values from other devices for the trustee.	Recommendation (RC)	It focuses on permissions that are assigned to a device based on the service provider's trust towards this device.	
[28]	Multiplication for transitivity with no aggregation between direct and indirect trust.	Continuous [0,1].	-	-	-	Service satisfaction for direct trust is not discussed in detail. For recommended trust a satisfaction level is defined which depends on availability, processing capacity, recovery time. Connectivity and peak-load performance.	Service satisfaction.	This trust model is defined in the context of Software as a Service (SaaS) in cloud environments. It is perceived, that a consumer will ensure the trustworthiness of the relevant service providers before accessing a service.	Evidence, experience and reputation-based trust.
[27]	Multiplication for transitivity and	Continuous [-1,1].	Its context-aware multi-service approach	-	-	Service satisfaction: the service for which the node	Service satisfaction	A trust management	Evidence, experience,

	weighted average for aggregation but with centralised propagation (only recommendations are used).		introduces a high level of sophistication in trust management.			was evaluated, the resources-based capability of the node at the time of the service request, the time at which the service was requested. How the service satisfaction is evaluated depends on the service.	(negative/positive).	system for the IoT which takes into consideration that an IoT network can contain different kinds of devices providing different kinds of services. On this basis it proposes a context-aware and multi-service approach.	application-specific and similarity-based trust.
[23]	Only direct trust.	Continuous [-1,1].	A multi-service approach is followed.	-	The model convergence time is relative to the number of nodes and simulations show it doesn't scale well.	Service satisfaction is relative to the service for which the node was evaluated. Services are valued based on their processing and energy requirements. More demanding services have a higher weight value.	Service satisfaction (negative/positive). The positive/negative follows an award/punishment logic of the trustor is weighted based on the above.	A trust management system for the IoT which takes into consideration that an IoT network can contain different kinds of devices providing different kinds of services. On this basis it proposes a multi-service approach.	Evidence, experience, and application specific based trust.

[24]	No recommendations. Calculations are done on the central TMS which receives raw data from sensors.	Continuous [-1,1].	The MAPE-K control feedback loop improves trust level consistence over time in a highly dynamic environment as opposed to the simulations run without feedback.	-	-	The IoT sensors send raw data that they collect, and the representation of the data differs based in the trust metric they are referring to. Examples include: A sensor that senses availability would send the number of successful ping requests. A sensor that senses reliability would send the Bit Error Rate (BER) of the target environment. Response time could be evaluated based on the round-trip time and capacity based on the current sessions of a device and maximum number of connections to a device.	Availability: availability of resources, Reliability: a reliable system always produces correct information, Response time: irregularities in response time can mean a device is compromised, Capacity: accessibility and scalability	This model proposes a framework for integrating cloud and IoT in order to develop a cloud-based autonomic TMS which evaluates the level of trust in an IoT cloud ecosystem.	Evidence and experience-based trust.
[26]	Multiplication for transitivity and weighted mean of evidence for aggregation	Continuous [0,1].	He achieves better performance than both resilient-RPL (rRPL) and classical-RPL (cRPL) in various tests. When the network size increases the number of bad paths is reduced as opposed to the other implementations where it increases.	-	A small false positive rate is associated with bad nodes detection.	Nodes monitor the activity of their neighbours. So, a node *x* sends a packet to a node *y* to be forwarded. If the *y* forwards the packet correctly and timely, *x* increases the value of positive experiences, otherwise the value of negative experiences.	Trust metrics are belief, disbelief and uncertainty. They are relevant to the number of positive and negative experiences.	This model proposes a trust-based extension of the RPL routing protocol.	Evidence and experience-based trust.

			In the simulation where the number of bad nodes varied the proposed trust RPL (tRPL) has less bad paths than the other two. Better packet delivery ratio. tRPL can detect 80% of bad nodes successfully.						
[37]	Indirect trust is not weighted, and aggregation is done through average of direct and indirect sum.	Continuous [0,1].	Simulations show that it avoids malicious paths better than classical RPL.	-	This model uses additional hardware embedded in every device for security computations and processing.	For the ERNT metric computations nodes monitor their neighbours for selfishness, energy, and honesty.	Extended RPL Node Trustworthiness (ERNT)	This model proposes an alternative scheme for the RPL protocol.	Comprehensive.
[13]	Indirect trust is not discussed.	Continuous [0,1].	It achieves high levels of security.	The use of T-IDS is resource-demanding in both storage and communication overhead.	This model uses additional hardware embedded in every device for security computations and processing.	For the ERNT metric nodes are monitored for honesty, energy and mobility.	Extended RPL Node Trustworthiness (ERNT)	This model considers the work of [37] and extends it by proposing a trust-based IDS (T-IDS).	Comprehensive.
[25]	Subjective: Multiplication for transitivity and weighted	Continuous [0,1].	In the simulations with Class 1 malicious objects, both Subjective and Objective models	-	-	Subjective: Feedback: each node evaluates the service received with a value in [0,1].	Subjective/Objective: Feedback, Total number of	Trustworthiness for the social IoT. Two separate models are	Comprehensive.

	sum for aggregation. Objective: The computation is done by a dedicated node based on feedback by the other nodes and nodes retrieve trust values from it. The feedbacks are weighted based on the credibility and transaction factor metrics.		outperform TVM/DTC [34] and TidalTrust [39] models.			Total number of transactions between two nodes. Credibility: the credibility of the recommender is based on the direct trust of the recommendation receiver towards the recommender and the centrality of the recommender. Transaction factor: the relevance of a transaction considered between two nodes to discriminate relevant from irrelevant ones. Relationship factor: based on the nature of the relationship between two nodes different values are assigned. The relationships include but are not limited to ownership object relationship and co-location object relationship. Notion of centrality is based on the sequence	transactions, Credibility, Transaction factor, Relationship factor, Notion of centrality, Computation capability.	proposed, namely, Subjective and Objective Trustworthiness.	

						of social links that form the path between the two nodes. Computation capability: Objects with greater computation capabilities are considered as more capable of malicious activities. Objects are divided in two classes. Class 1 includes objects with great computational capabilities, such as smartphones. Class 2 includes objects with only sensing capabilities, such as a sensor. Objective: Feedback: same as in Subjective. Total number of transactions: same as in Subjective. Credibility: depends on relationship factor, computation capabilities and total			

						number of transactions. Transaction factor: same as in Subjective. Relationship factor: same as in Subjective. Notion of centrality: is based on the number of times the node requested a service, the number of times it acted as an intermediate node in a transaction, and how many times is has provided a service. Computation capability: same as in Subjective.			
[29]	Dempster-shafer: based on an algorithm. Verification of interaction proof (VIP): based on a ratio of the positive/negative interactions a reader had.	Dempster-Shafer: {Trusted, Malfunctioning, Malicious} Verification of interaction proof: Continuous [0,1].	Fast trust convergence. Able to support large scale RFID applications. Both Dempster-Shafer and VIP outperform the Bayes-based scheme in convergence speed, malicious event detection rate.	-	-	Behaviour: Discarding data, tampering with data, Replaying or forging data. Positive interactions: RFID tag rates a reader with 0 for negative and 1 for positive.	Dempster-Shafer: Behaviour. Verification of interaction proof (VIP): Ratio of positive interactions.	This model proposes a trust management system for multi-domain RFID systems. The RFID system model consists of one or more domains and each domain	Evidence, experience and reputation based.

								includes RFID tags, RFID readers, authentication centres and an administration centre. A centralized trust propagation approach is followed.	
[31]	Adaptive neuro-fuzzy inference.	Not clarified. (pg. 8-9)	The proposed model TMM outperformed TRM [21] in both packet forwarding ratio and energy efficiency. It also outperformed AODV (Ad hoc On-Demand Distance Vector) [40] and trusted-AODV in throughput. TMM has higher accuracy and f-measure than a model using a fuzzy inference system instead of the adaptive neuro-fuzzy inference system.	-	-	Behavioural trust: Relative frequency interaction: relative to the number of interactions between the nodes and the total number of interactions with other nodes over the same period of time. Intimacy: relative to the time of interaction between two nodes and the cumulative time of interactions with other nodes. Honesty: based on the numbers of successful and unsuccessful interactions.	Behavioural trust: Relative frequency interaction, Intimacy, Honesty. Data trust: Direct, Indirect.	This model proposes a Neuro-Fuzzy based Brain-inspired trust management model for cloud based IoT architectures security and data reliability.	Comprehensive.

						Data trust: deviation of node's current data from the historical data of the node. In both direct and indirect evaluations.			
[30]	Recommendations are computed and provided in a centralized manner. Objective information entropy theory is used for transitivity. Weighted sum for aggregation.	Continuous [0,1].	It achieves better global convergence time and task failure ratio than PSM and Distributed Reputation Management (DRM) [41]. It's lightweight in terms of complexity.	Space complexity (communication overhead): 3*m*n*δ, m: number of clusters, n: size of clusters, δ: the maximum number of trust computing for a given Δt. Time complexity: the total time complexity of overall trust evaluation is $O(n^2)$.	-	After each transaction, each participating node evaluates the other node based on service completion and sends the value to the broker.	Service satisfaction.	This model proposes a trust computing mechanism specifically designed for IoT edge computing.	Evidence, experience and reputation-based trust.

3.2.2 Trust management architectures

In section 3.2.1 we have reviewed the existing trust management systems and their features. In this section, we survey the relevant trust management architectures, which dictate how the TMS components are deployed in the target network, the relationships between the components and the information flows. A trust management system involves a number of different components that are involved in the various activities taking place within the system; taking into account the aspects of trust models identified in section 3.2.1, we can identify the following types of components:

1. *Data collection components*. These components collect the necessary data for performing trust assessment, which range from consumer satisfaction, QoS aspects, suspicious/dishonest behaviour and so forth.
2. *Data storage components*. These components store the data collected by data collection components and make them available for trust calculations.
3. *Trust calculation components*, which extract the data from data storage components and calculate trust. In this process, they may query other trust calculation components regarding their trust assessments and use the replies in their computation.
4. *Trust consumers*, which query trust calculation components regarding trust assessments and use the obtained values for implementing security policies.

While data collection and storage components as well as trust consumers are typically dispersed across the network, trust calculation components are laid out according to different paradigms, and these layouts characterize the trust management system architecture. Overall, the following categories are identified for trust management architectures: (a) centralized, (b) hierarchical, (c) distributed/Peer to peer. In the following subsections we elaborate on each of the categories, presenting its features and prominent application cases.

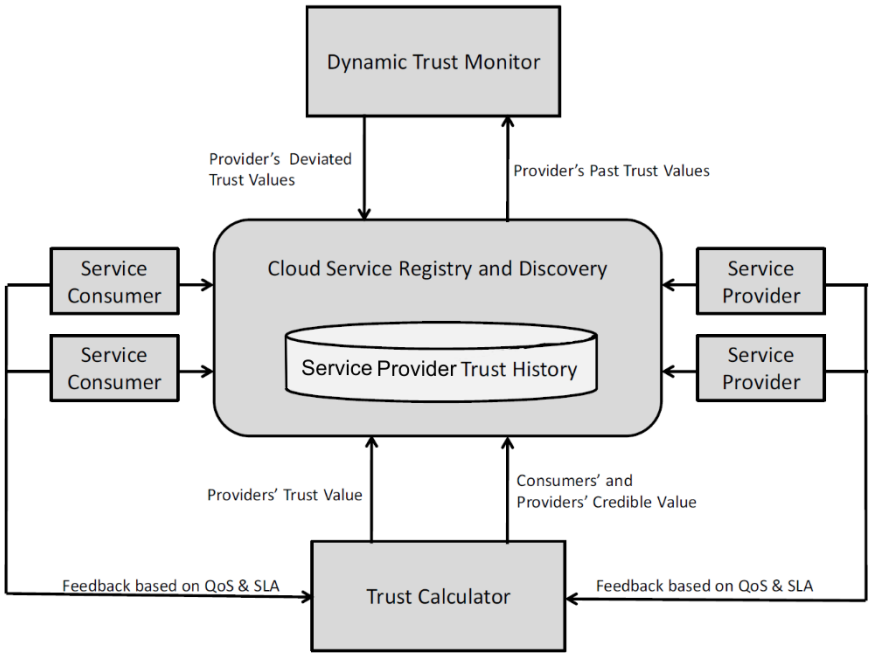


Figure 3.1. A centralized trust management system architecture

3.2.2.1 Centralized

The centralized architecture paradigm involves a unique trust management authority, which collects all the information necessary for trust calculation and computes the trust score for entities. Then, interested parties can query the trust score of entities, subject to suitable authorization.

Centralized systems are known to have scalability and reliability issues, hence only few systems have been reported in the literature to follow this paradigm. Figure 3.1 presents a reference centralized TMS architecture from [42]. In this architecture, a single trust calculator collects all the metrics related to trust computation and computes the trust metric for entities. Both service providers and consumers are assigned a trust score (credibility): the trust score assigned to service consumers moderates the weight of the trust metrics they contribute to the system.

3.2.2.2 Hierarchical

The hierarchical architecture paradigm identifies clusters of nodes, where each cluster elects a coordinator. Nodes within a cluster liaise with the coordinator, exchanging observations, metrics and trust values; the coordinator is responsible for synthesizing the trust assessments of the nodes within the cluster it coordinates into a comprehensive trust score and for communicating with other coordinators to exchange trust values. Hierarchical architectures are well-suited for IoT infrastructures, where nodes with limited resources are placed under the coordination of the corresponding gateway node, which is more resource-rich and can host resource-intensive operations. Respectively, trust assessments are generally performed having available more detailed local data (measurements and observations obtained and collected at cluster level) whereas inter-cluster communications are limited to the exchange of either trust assessments or data summaries, rather than detailed data.

Nodes in hierarchical systems may be organized across multiple levels of hierarchy. In this line, [43] describes an architecture where nodes are clustered into autonomic nodes, and autonomic node contains multiple autonomic Decision Entities (DEs). In turn, A DE is introduced in the Generic Autonomic Network Architecture (GANA) designed to follow hierarchical, sibling and peering relationships with other DEs within a node or network. It collects information from peering DEs or sibling DEs, makes decisions and manages Managed Entities (MEs) at a lower level i.e., the level of abstracted networking functions. DE is the element that drives the control-loop over the MEs and implements the self-* functionalities e.g., self-configuration, self-monitoring, self-healing.

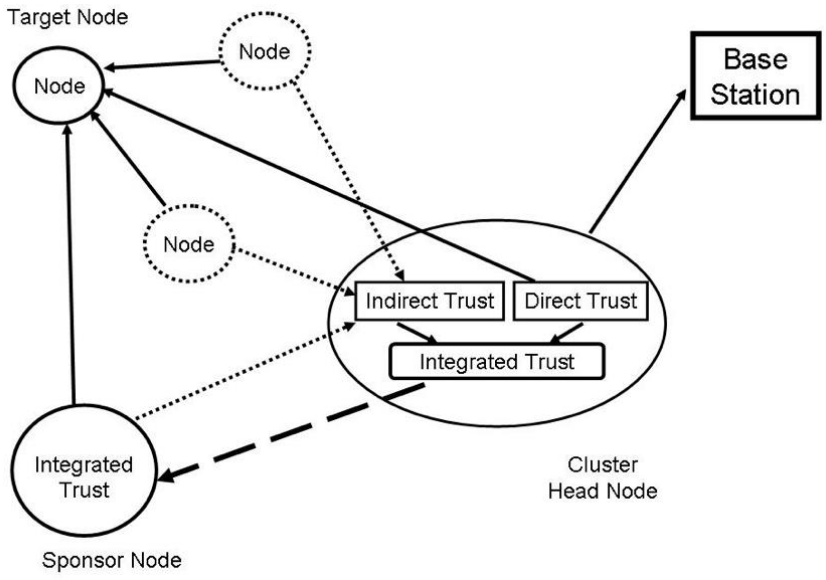


Figure 3.2. Internal structure of a cluster [44]

Figure 3.2 presents the internal structure of a cluster, as per the design reported in [44], whereas Figure 3.3 depicts the organization of multiple clusters into a hierarchical trust management system [43]. Karame et al. [45] is also an example of a hierarchical architecture, applied in peer-to-peer nodes having super-peers.

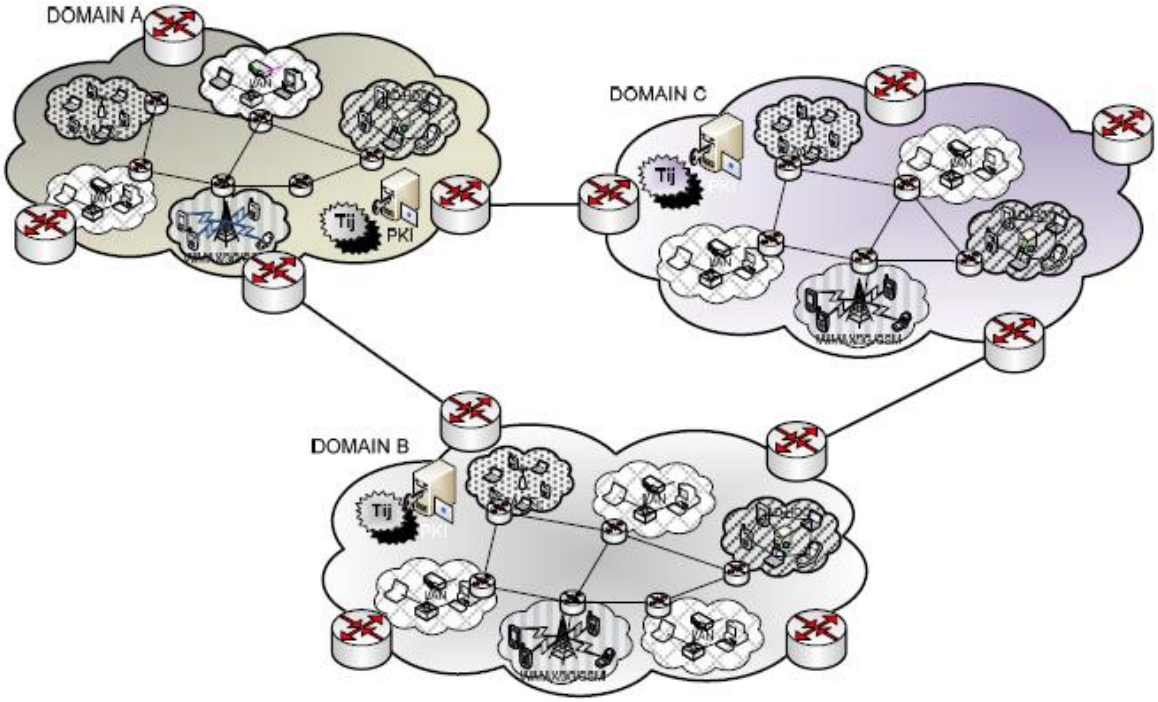


Figure 3.3. Integration of multiple clusters into a hierarchical trust management system [43]

3.2.2.3 Distributed/Peer to peer

In this architectural paradigm trust management components are dispersed across the network and operate autonomously. Each node makes its own observations and measurements and maintains them into a local database. Nodes may also request from other peer nodes either detailed measurements and observations or synopses of measurements of measurements and observations, or trust assessments; then, they compute a trust score for other entities, synthesizing their own data and the data they have received.

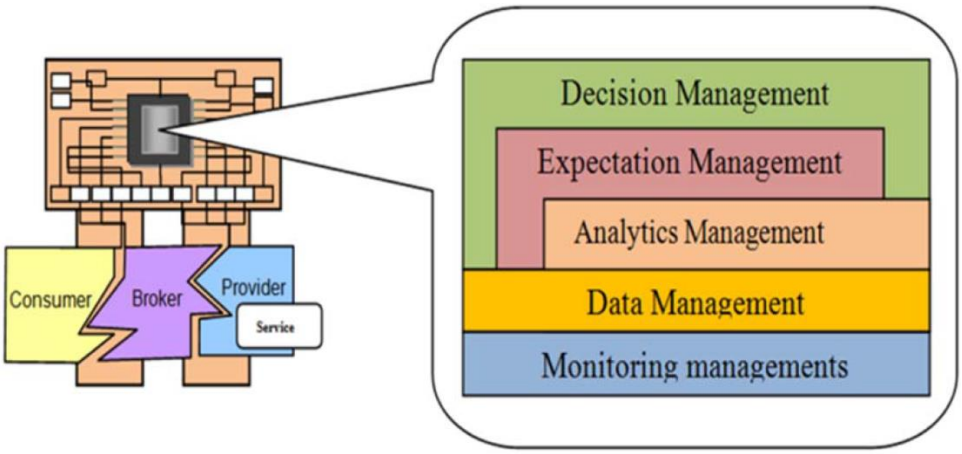


Figure 3.4. Components within a peer node participating in a TMS [46]

Figure 3.4 illustrates the components within a peer node participating in a TMS. The node itself responds to requests from other nodes for trust assessments (and provisionally other data), while it can become itself a client to other nodes, requesting trust assessments (and provisionally other data).

The distributed/peer to peer paradigm is the most widely used in the literature: [46]–[50] are typical cases where this paradigm is used.

3.3 Trust Management System Implementations

In the previous subsections, we have surveyed of trust management methods, protocols, algorithms and architectures; in this section, we examine trust management implementations, focusing on the open source implementations, which could be used as a basis for the implementation of the Cyber-Trust TMS. To this end, we have performed extensive searches in the major open source repositories, namely GitHub[1] and SourceForge[2]. For each of the trust management systems located, we performed an initial assessment, dropping those repositories that were incomplete or not adequately populated (i.e., contained only a few files or only documentation with no concrete implementations). Subsequently for the remaining implementations, we considered the following aspects:

1. *Domain of use*. The intended domain of use for the software was assessed, examining whether the system was oriented to the computer security trust domain or other trust domains with different concepts and requirements; in particular, many systems were oriented towards *financial trust*, not being thus suitable for use within the Cyber-Trust project.
2. *Functionality*. The functionality offered by the system was analysed, considering whether REST/web service/remotely invokable APIs are offered, the existence of UIs and in particular web-based UIs and the algorithms implemented.
3. *Extensibility, modifiability, active support and documentation*. These properties are required for adaptation of a system to the needs to Cyber-Trust to be accommodated. Extensibility and modifiability are contextualized for the Cyber-Trust project, considering the expertise of the consortium in the implementation language and environment.
4. *Deployability to Cyber-Trust target platforms*. The Cyber-Trust system specifications dictate that TMS instances will be running on data centres, smart gateways and smartphones; each deployment target has its own runtime environments and resource capabilities and TMS implementations should be able to run efficiently on top of all these deployment targets.

In the following paragraphs, we present the open source TMS implementations surveyed. Systems designed for use in other domains, and therefore not being useful for the context of Cyber-Trust are briefly described in subsection 3.3.8.

3.3.1 Soutei

Soutei (https://sourceforge.net/projects/soutei/) is a trust-management system for access control in distributed systems. Soutei policies and credentials are written in a declarative logic-based language. Soutei policies are modular, concise, readable, supporting conditional delegation. Policies in Soutei support verification, and, despite the simplicity of the language, express role- and attribute-based access control lists, and conditional delegation. They support policy verification, and, despite the simplicity of the language, express role- and attribute-based access control lists, and conditional delegation.

Soutei provides a number of interesting concepts, and may model, among others, role-based access control, capabilities, policies predicated on time and lists, trees, organizational charts & partial orders. It provides a TCP server hence functionalities can be remotely invokable. Its documentation however is limited, hindering installation, configuration and maintainability. The Cyber-Trust consortium does not have adequate experience with the Haskell language, hence the extension and maintenance of the software will be further hindered. The provided implementation is almost 10 years old and in many aspects it is incompatible with the recent developments of the Haskell language; Soutei does not compile and run successfully under the recent versions of the Haskell compiler; it has been found to compile successfully under version 6.8.3 of the Haskell compiler, which is severely outdated and may contain functionality or security issues. Finally, running Haskell on Android devices, which is a significant target for the Cyber-Trust project, has only been reported

[1] https://github.com/

[2] https://sourceforge.net/

in 2018, and requires the use of low-level techniques, such as Java Native Interface (JNI) or the Native Development Kit (NDK) [51] which introduces an additional set of required programming skills and another level of mapping which increases the probability of errors. The requirement to use version 6.8.3 of the Haskell compiler will probably introduce additional compatibility issues with the Android platform.

Considering the above, Soutei is not a prospective candidate for use in the Cyber-Trust project.

3.3.2 Trust guard

Trust Guard (https://github.com/blatyo/trust_gaurd) is an implementation of an algorithm for countering vulnerabilities in reputation management for decentralized overlay networks, introduced in [52]. This implementation dates back 9 years and is reported by the author as "untested". It includes some code on how trust values are computed/updated according to newly arriving information and old estimates decay with time in favour of new information. The implementations are simple and can be directly derived from the reference paper [52]. No server is provided, hence no remote invocation is possible; only running through the command-line is supported. Obviously, the implementations can be wrapped within web service containers, however considering the simple nature of the implementations and the fact that the project is untested, the benefits from using the Trust Guard are minimal and additionally introduce the need for testing and binding to the Ruby platform.

Considering the above, Trust Guard is not a prospective candidate for use in the Cyber-Trust project.

3.3.3 pyKeynote/keynote library

pyKeynote (https://github.com/argp/pykeynote) is a Python extension module for the KeyNote trust management system [5]. It provides a high-level object-oriented interface to the KeyNote trust management API. The implementation is very outdated, with its last update dating back 12 years, and it relies on the keynote library (http://www1.cs.columbia.edu/~angelos/keynote.html) which dates back at an even older timepoint (2000). Some concepts of the keynote library –which is written in C-, including assertions and grants could be usable. The C library accommodates provisions for local invocations, it could however be wrapped under remotely invokable containers. Porting though to the TMS implementation language (Java) would necessitate much effort, since supporting libraries are not directly available in Java.

Considering the above, the Keynote library is not a prospective candidate for use in the Cyber-Trust project.

3.3.4 SAFE

SAFE (https://github.com/wowmsi/safe) is an integrated system for managing trust using a logic-based declarative language. Logical trust systems authorize each request by constructing a proof from a context---a set of authenticated logic statements representing credentials and policies issued by various principals in a networked system. Two informal publications ([53] and [54]) describe the theoretical and practical basis of the system. SAFE aims to address the problem of managing proof contexts: identifying, validating, and assembling the credentials and policies that are relevant to each trust decision. The approach of SAFE to managing proof contexts is using context linking and caching. Credentials and policies are stored as certified logic sets named by secure identifiers in a shared key-value store. SAFE offers language constructs to build and modify logic sets, link sets to form unions, pass them by reference, and add them to proof contexts. SAFE fetches and validates credential sets on demand and caches them in the authorizer. We evaluate and discuss our experience using SAFE to build secure services based on case studies drawn from practice: a secure name service resolver, a secure proxy shim for a key value store, and an authorization module for a networked infrastructure-as-a-service system with a federated trust structure [53], [54].

The SAFE implementation is distributed under the Apache 2.0 license, which is permissive, so it can be reused, either as a whole or at the level of selected portions. Scala implementations can be run on Android (https://scala-android.org/) with a small footprint, however more extensive tests should be made to determine the footprint of the particular implementation.

The code implementing SAFE has some high-level documentation regarding the architecture, however the documentation on compiling and running the code is lacking. No executable commands or relevant sources are present in the default distribution, hence execution procedures cannot be determined. The code implementing the SAFE TMS lacks comments, therefore code modifiability and extensibility is low. Porting though to the TMS implementation language (Java) would necessitate much effort, since supporting libraries are not directly available in Java.

Considering the above, SAFE TMS is not a prospective candidate for use in the Cyber-Trust project.

3.3.5 TMLib

The *tmlib* system (https://github.com/pchapin/tmlib) is a library of functions that allow applications to support trust management style distributed authorization. The TMlib library is reported to provide an administrative application that can be used to create and manually verify certificates in multiple certificate formats. In addition, this library provides functions for performing a proof of compliance computation that can be used in any application that wishes to use trust management services.

TMLib assumes that the participating nodes specify local policies and encrypts the communication between nodes. Node identity is proved by means of certificates. Fundamentally TMLib is a library of functions that can be called by an application that is interested in trust management services. However, there are a number of administrative tasks that any node must support in order for the system to be usable. Accordingly, TMLib comes with an administrative application that allows its user to perform certificate creation, managing of public key and policy databases, as well as executing test queries and setting policy dissemination rules.

TMLib has an undocumented dependency on an ACO project (presumably *ant colony optimization*) written in ADA; however, no such open-source project or relevant files could be located. Hence. It was not possible to compile and test the project. Furthermore, consortium expertise with ADA is very limited, hindering thus modifiability and extensibility.

Considering additionally the lack of documentation, TMLib is not a prospective candidate for basing the Cyber-Trust TMS implementation.

3.3.6 Cloud trust protocol daemon

The Cloud Trust Protocol Daemon (CPTD - https://github.com/CloudSecurityAlliancePublic/ctpd) is a prototype server implementing the Cloud Security Alliance's Cloud Trust Protocol. The Cloud Trust Protocol (CTP) is designed to be a mechanism by which cloud service customers can ask for and receive information related to the security of the services they use in the cloud, promoting transparency and trust. This prototype called *ctpd* is a Unix-style server written in Go with *mongodb* as a database backend. It has been tested on Ubuntu/Debian Linux and Mac OS X. The code of *ctpd* is reported to be still in 'beta' stage and is mainly intended for testing and research purposes.

ctpd aims to fully implement the CTP data model and API [55], [56] as well as the non-official CTP 'back office' API [57]. This API includes provisions for managing the concepts of:

- *service views*: represents a service offered to a specific customer under the responsibility of a single provider. This service is usually described in an SLA or service interface. A service-view encompasses a set of assets.
- *Assets*: used to represent any tangible or intangible element of a cloud information system, such as for example simple API URLs, storage, processor cores or compute instances, databases, full blown platforms, etc. A set of attributes is attached to an asset.
- *Attributes:* used to represent characteristic of an asset that can be evaluated quantitatively or qualitatively and is identified with a distinct name (e.g., “availability”, “incident response”, etc.). Associating a value with a security attribute requires the specification of a measurement.

- *Metrics:* a standard of measurement, which will be referenced in measurements. A metric is typically specified in an external document that describes in human readable form the conditions and the rules for performing the measurement and for understanding the results of a measurement.
- *Measurement,* which describes how a specific attribute is evaluated, using a specific metric.
- *Triggers*, which enable cloud service customers to receive notifications when specific conditions are met. A trigger is a conditional expression on the measured value of a security attribute.
- *Log entries*, which are generated by triggers.
- *Dependencies*, which are used to describe relationships between different cloud services that are associated together to form a cloud supply chain.

Notably, the Cloud Trust Protocol Daemon does not compute any trust or risk metric: its purpose is to manage the concepts listed above, which can be used (among others) in the formulation of trust scores, the delivery of notifications (through triggers) and the creation of persistent log entries. However, it should be noted that triggers are not fully implemented (trigger deletion is lacking), and XMPP-based notifications (through which notifications raised by triggers are delivered) are not implemented at all (c.f. [56]).

Under this view, the utility of CPTD for the implementation of the Cyber-Trust TMS is limited, and will not be further considered towards this direction.

3.3.7 Retrust

Retrust (https://github.com/liamzebedee/retrust) is a work-in-progress protocol for decentralized reputation/trust, based on Evidence-Based Subjective Logic (EBSL) [58]. The model is based on capturing interactions between nodes in the form *(source, target, value)*, with *value* being > 0 for trusted interactions and < 0 for negative interactions. In this model, trusted friends/seeds need not be specified or explicitly maintained, since this information is automatically derived from interactions. An application-agnostic mode is also considered, in which reputation is a subjective-logic opinion of (belief, disbelief, uncertainty) that can model any quality of reliability in interaction. Naturally, the implementation should provide implementations of the methods computing the reputation of entities: the reputation for an entity is *perspective-specific*, i.e. a single entity may be assigned multiple reputation scores, depending on the perspective under which it is evaluated. The perspective may be a self-view, the individual view of another entity or the view of a group of other entities (e.g. all entities belonging to a specific entity category such as servers within the demilitarized zone, or entities bound together with any arbitrary criterion).

The implementation is command-line based, therefore it is oriented towards single runs that retrieve interactions/evidence for entities, compute the results and display them and/or generate graphical representations for them. This means that substantial development effort should be put into modifying the code so as to provide server-type operation, i.e. modify the code to run as a background service and render it capable to responding to REST calls as well as receiving from third parties and maintaining trusted notifications regarding evidence, on which reputation computation will be based. Considerations also exist regarding the efficiency of the code: a simple simulation involving 10 “good” nodes, 20 “bad” nodes and 20 “Sybil” nodes, and having few interactions between nodes, 15.5 seconds were needed to run it on a Linux server with one 6-core Xeon E5-2420@1.90GHz CPU and 8GB of memory, and requiring a virtual memory size of 1.15GB (albeit the resident set was only 80MB; the required size of virtual memory can pose problems in mobile or –more generally- resource-constrained devices).

Concepts used in *Retrust* (evidence-based assessments, multiple perspectives) have been included in the Cyber-Trust TMS implementation.

3.3.8 Systems in other domains of use

The *Linux SGX Trust Management Framework* (https://github.com/IBM/sgx-trust-management) is a system for supporting the Software Guard Extension technology, available in Skylake and later processors. SGX technology supports the creation of *enclaves*, i.e., secure memory regions that are protected with hardware

encryption in the system-on-chip (SoC). In more detail, according to the SGX framework, the data exists in unencrypted format only inside the processor. Before being written to the main memory it is encrypted by the SoC and then decrypted by the SoC when fetched from the main memory.

TrustApp (https://github.com/dedicatedvivek/TrustApp) is a financial risk assessment application, with its core model involving banks and expenses. Furthermore, it lacks documentation and the code does not readily run on Unix due to some non-portable conventions.

TrustFeatures (https://github.com/hashinclude-co-in/kamban.org) is oriented towards non-governmental organisation (NGO) members and volunteer management; to this end it includes features such as contact management, survey management, event management, inventory management and task management. These aspects do not intersect with the functionalities needed in the context of Cyber-Trust.

Imob (https://github.com/zeqing-guo/imob) claims to implement "an identity and trust relationship management on blockchain for IoT", however no relevant functionalities or documentation are implemented in the code.

Trust Management System (https://github.com/shiwenbo/Trust-Management-System) accommodates the concept of nodes that are verified by credentials and assume roles, however there is no notion of trust and risk metric computation.

The *Tennessee Risk Management Trust* (https://github.com/lindseyemaddox/tnrmt) is oriented towards economic insurance, and its core concepts are loss control, property and liability, tort liability etc. In this respect, its scope does not intersect with the functionalities needed in the context of Cyber-Trust.

The *CA* system (https://github.com/pontiflex/trustme/tree/master/CA) is a web-based application, appearing to suite the management of certificates. Its functionality is limited, with a very limited overlap with the functionalities needed in the context of Cyber-Trust, while issues exist in the setup process and no updates have been provided for 6 years.

Trust composer (https://github.com/ricktobacco/trust-composer) is a web application for demonstrating a secure trust composer on blockchain using Hyperledger composer. The model realized by *Trust composer* involves claims for *service/resource accesses* issued by *users*; these claims are supported by *proofs*, whereas *assessors* are a specific subclass of users that provide guarantees to support a user's claims. Finally, *services* maintain a trust balance of users they manage and consider claims, together with associated proofs and assessor-issued guarantees to accept or deny requests. In more detail:

- *Users* are the super class of all other participants. They have a name and may create claim request assets, which may be further used to build trust for exchange against access balances with services.
- *Issuers* issue claim receipts based on users' requests, upon examination of proofs therein. They are the super class of services, subclass of assessors, and have a list of assessors operating on their behalf.
- *Services* upload resources with their associated access costs, and maintain a list of balances for the users that have requested access and also been granted trusts by assessors operating on behalf of the service.
- *Assessors* operate on behalf of services and issuers to package claim receipts, and assign levels of trust according to their own weights for various claim definitions (based on their verification specializations).

The example usage listed in the *TrustComposer* distribution is oriented towards the economics domain, persons submitting claims for damages they sustained from disasters or hospitals (as service providers) considers level of guarantee from assessors (e.g., insurance companies) to grant resources, some concepts might be used in the Cyber-Trust TMS; however, implementations that are closer to the Cyber-Trust domain, in particular ReTrust, provide more direct analogies, hence *TrustComposer* will not be further considered.

3.4 Conclusions and directions

In this section we have reviewed the aspects regarding trust management and risk assessment, which have been taken into account in the development of the Cyber-Trust TMS component.

In particular, the Cyber-Trust TMS will exploit behavioural aspects and status assessments to determine the trust status of individual devices, while trust levels, combined with environmental aspects (e.g. threat agents and security controls) can derive risk levels. We have also performed a review of the state-of-the-art trust models, which describe how independent entities can operate in the context of a distributed system to exchange, synthesize, propagate and update trust metrics. From these models, we have extracted the trust relationships that are pertinent to the Cyber-Trust context (user-to-user, user-to-device), and these relationships are included in the Cyber-Trust TMS.

A number of contemporary trust management models have been examined: some of them are specifically designed for the IoT domain, however a number of them have not be proven to be resilient to attacks [31], [24] or have been shown to resist only very few attacks [21], [23], [30]. The trust models described in [25] and [27] have been shown to be capable of withstanding most attacks: [27] includes a centralized propagation aspect which could introduce bottleneck problems in a highly populated network, albeit it could fit the domain of a smart home, or an enterprise. From the non IoT-specific models, the one proposed by Chen, [15] encompasses many defences against attacks and has been designed in a generic fashion; however, it should be tested in the context of large-scale IoT. The properties of the models that (a) contribute to the resilience of the system and (b) do not impose high level of expertise by the users or disproportional resources and (c) allow a performing implementation that can be deployed in the scope of the Cyber-Trust domain will be exploited. In particular, social relationships between users; user-to-device relationships; and cooperation between trusted TMSs, in a peer-to-peer fashion.

The data needed by each model for its operation have also been considered. Some models require data that carry application-level semantics, such as competence, cooperativeness and honesty, and -in order to have such data available- applications should be *trust-aware* i.e. are programmed to (a) assess the trust level of their peers and (b) compute the trust assessment based on the observable behaviour of their peers and (c) provide trust feedback to the entity computing the trust (i.e. the TMS). Since applications at this stage do not operate in this fashion, neither such operations are facilitated by existing libraries and underpinnings, a design decision was taken to not use such data, and limit the data to be used to those that can actually be obtained from existing technologies, such as signature-based and behaviour-based IDS and IPS systems, as well as health attestation systems. Nevertheless, the TMS is designed in an extensible fashion, to enable the incorporation of additional information into the trust computation process.

Regarding trust management system implementations, none of the identified implementations was assessed as usable in the context of Cyber-Trust: many of them are out of context (pertaining to other domains), while other implementations present different challenges, including lack of features, implementation languages that are not appropriate for mobile platforms or for which the consortium lacks experience, large footprints or performance issues. Taking these into account, a Java-based implementation was opted for, since Java provides an omnipresent and portable environment, capable to run on multiple devices ranging from high-end desktop computers to Android devices or even Arduino processors [59].

4. Cyber-Trust TMS design

In this section, we present in detail the design of the Cyber-Trust TMS. In this context, we first describe the generic model for trust computation (subsection 4.1), as well as on the relationship between the TMS and other components of the Cyber-Trust architecture. Subsequently, in subsection 4.2, we elaborate on the methods used for the computation of trust. Finally, in subsection 4.3, we provide details on the internal structure on the TMS.

4.1 Generic model

Figure 4.1. The entities in the IoT and SOHO environments and their relationships depicts the entities considered by the Cyber-Trust trust model in the IoT/Smart home/SOHO environments and the relationships between these elements. The elements are as follows:

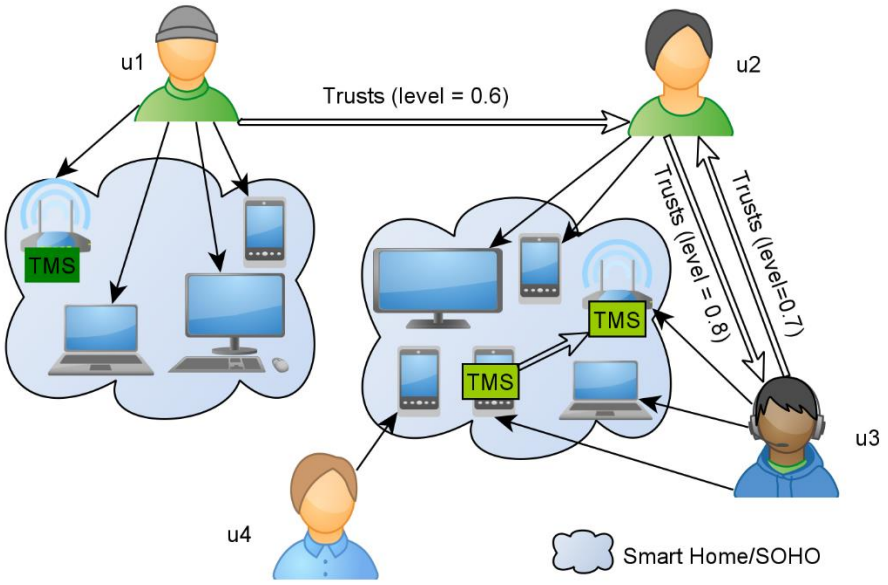


Figure 4.1. The entities in the IoT and SOHO environments and their relationships

- *Devices*, that operate within the considered environment.
- *Users*, who own devices. A single user may own multiple devices. Users may develop trust relationships between them; trust relationships between users are *directed*, *not necessarily symmetrical, not transitive* and *weighted*, i.e.:
  - Some user u_1 declares to trust some other user u_2, providing a *trust level*, expressing u_1's confidence that u_2 will not act in a way that is harmful for u_1 -or even will act in a way that is beneficial for u_1.
  - The assertion of trust towards u_2 made by u_1 does not imply in any way that u_2 *also trusts* u_1, expressing the fact that trust may not be reciprocated [23]. It is still however possible that u_2 makes a separate, independent assertion that s/he trusts u_1; such an assertion may express a different trust level than the respective assertion made by u_1.
  - Trust is not transitive: if u_1 trusts u_2 and u_2 trusts u_3, no assumption is made that u_1 trusts u_3. An explicit assertion by u_1 is required to establish any trust relationship to any other user in the domain of discourse.
- *Trust Management System instances* (TMS): these are software agents operating within the considered environment and implement functionalities for computing trust levels towards devices. To compute the value of trust towards a device, the TMS synthesizes multiple pieces of information, either explicitly provided or gathered through observations. These pieces of information are:
  - *the status of the device*: this encompasses (1) information regarding the *device integrity*, i.e. the extent to which the device is known to run legitimate firmware/operating

system/software under a validated configuration, as contrasted to the case that the device firmware/operating system/software/configuration files have been tampered with; and (2) information regarding the *device resilience*, i.e. the extent to which the device firmware/operating system/software/configuration are known to have security vulnerabilities, as contrasted to the case that no such known vulnerabilities exist.

- *the behaviour of the device*: this includes information regarding whether:
    1. the device has been detected to launch attacks, or be target of attacks.
    2. the device's resource usage metrics are within a pre-determined range which is considered to be "normal" or deviate from it. The metrics can pertain to any observable aspect of resource usage metrics, e.g. CPU load, network usage or disk activity. Practically, any class of system metrics that can be quantified, and for which baseline metrics can be created so as to allow computation of deviations from the baselines is eligible for incorporation within this dimension. Similar practices are widely employed in monitoring infrastructures, such as Nagios [60], and may include metrics such as number of connected users, amount of free disk, total number of processes, number of processes corresponding to some specific service instance, etc.
    3. the device complies with pre-specified behaviour which has been whitelisted as benign. MUD specification [61] files are the most prominent source of such information, albeit their adoption and manufacturer support is lagging behind expectations.
- *the risk associated with the device*. Devices within the IoT may be attacked, and some attacks may be successful. The probability that each device is finally compromised can be computed taking into account only technical information, such as the reachability of the device and the vulnerabilities present on it, and attack graphs are a prominent tool for supporting such computations [62]. However, not all compromises have the same level of impact on the organization/person owning the device: the level of impact is moderated by the perceived value of the device. The perceived value of the device in turn is moderated by (a) the assets that the device hosts (e.g. a database) or the value/criticality of the dependent processes that the device supports (e.g. a temperature sensor may support a simple temperature reading application or the automated cooling system of a nuclear reactor).

  Furthermore, in the context of sophisticated, multi-staged attacks, a compromised device *d* may be used as a stepping-stone, enabling the attackers to launch attacks against other devices which are reachable from *d* and may be otherwise unreachable (or harder to reach), if *d* were not compromised. Notably, the devices that are reachable from *d* contain themselves assets that have a business value, and the technical probability that these devices are compromised in the context of multi-staged attacks can be jointly considered with the respective business values to provide an additional aspect of the risk associated with device *d*.

- The associated risk dimension combines the above-mentioned aspects i.e. (i) the technical probability that the device is compromised with the perceived value of the device, and (ii) the probability that the device is used as a stepping stone to attack other devices, in conjunction with the business values of the assets associated to these devices, to synthesize a single, comprehensive metric expressing the business risk applicable to a device.
- The trust relationship between the owners of the device running the TMS and the owner of the device, for which the trust evaluation is conducted. This aspect allows the propagation of the trust between users to the level of the devices they own.

The three separate trust dimensions, i.e. (i) status-based, (ii) behaviour-based and (iii) associated risk-based trust, are synthesized by the TMS instances into a single overall trust assessment.

Furthermore, *directed*, *not necessarily symmetrical, not transitive* and *weighted* trust relationships can be established between TMS instances, in the same fashion that trust relationships are established between users. The trust relationships between TMS instances are explicitly provided by the users owning the devices

on which TMS instances are run. Once a trust relationship stating that TMS instance T_1 trusts TMS instance T_2 is established, T_1 will source trust assessments for devices from TMS T_2, and take them into account when computing the respective devices' trust levels.

Finally, users are allowed to set explicitly the trust level of the devices they own, overriding the computations made by the TMS. This provision is accommodated to handle false positives mainly related to network attacks (an attack is flagged by relevant modules but was not actually performed), network anomalies (e.g. excessive traffic was detected but this was due to a user-initiated backup or a software/firmware update) and compromises (e.g. some software on the device was misclassified as malware). The TMS will be able to provide both the automatically computed and the explicit trust level of the device, so that relevant applications will be able to detect devices where major discrepancies exist and keep the users informed about such deviations, promoting awareness and facilitating intervention, as needed.

According to the description listed above, the TMS composes the trust score in a hierarchical fashion, as depicted in Figure 4.2. Trust score composition dimensions and aspects, undertaking a holistic view towards trust assessment. To perform this composition, the TMS necessitates different types of information for each device. The TMS operates in the broad context of the Cyber-Trust platform and sources the required information from other Cyber-Trust modules, as depicted in Figure 4.3. Cyber-Trust platform elements providing information to the TMS.

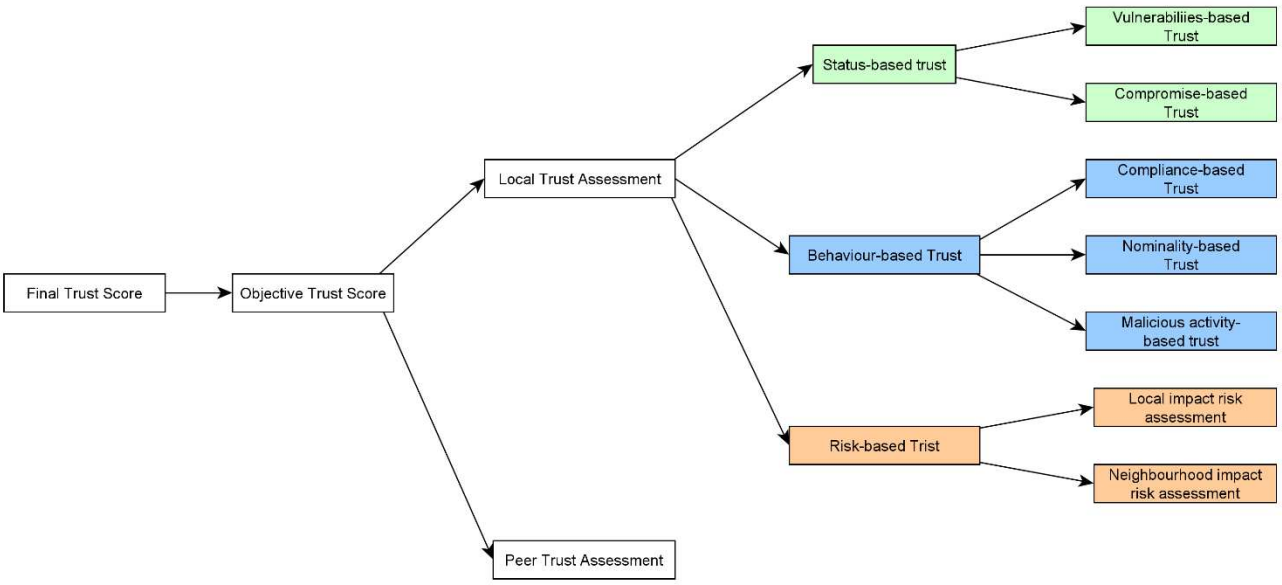


Figure 4.2. Trust score composition dimensions and aspects

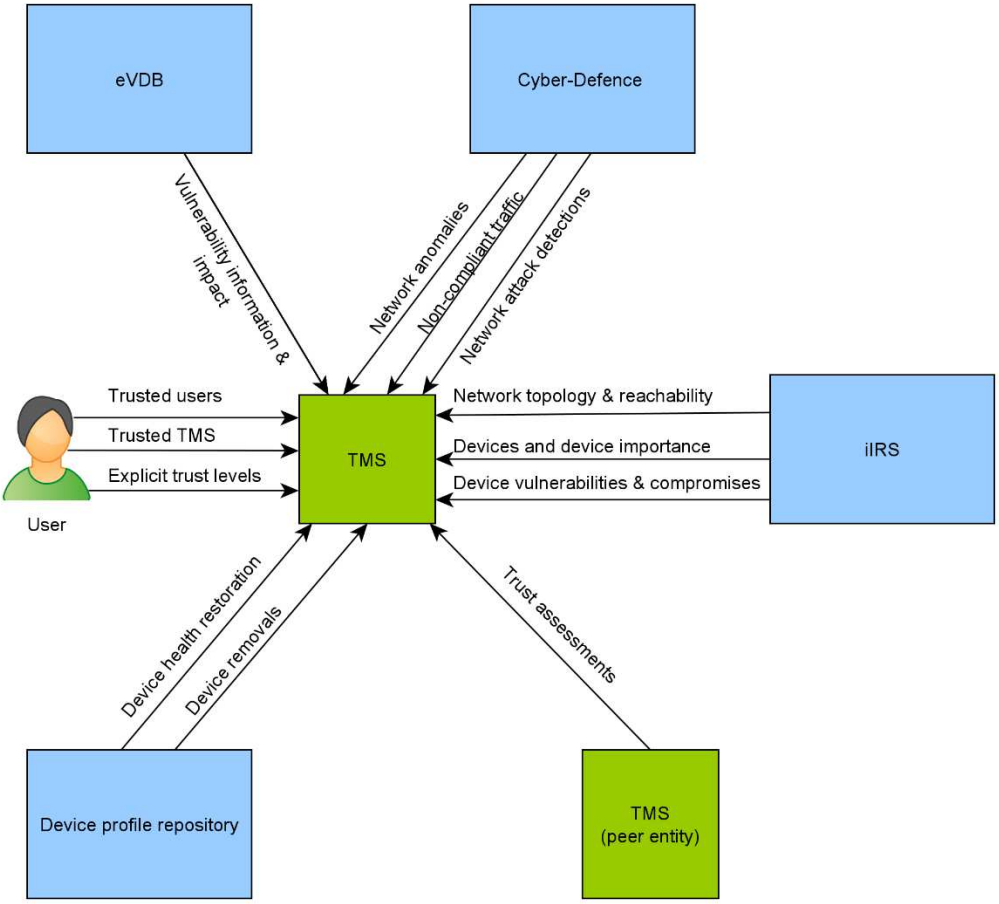


Figure 4.3. Cyber-Trust platform elements providing information to the TMS

In more detail, the information sourced from other Cyber-Trust platform elements is as follows:

- *Cyber-Trust platform users* provide information regarding the peer users they trust, the peer TMSs that are trusted and explicit device trust specifications. Naturally, user interaction with the TMS is mediated through an appropriate application.
- *The CyberDefense module* provides data regarding the network anomalies detected (deviations from the nominal device and network behaviour), the non-compliant traffic (traffic flows that have not been whitelisted as "acceptable behaviour" for the device) and network attacks (primarily in the context of signature-based detection), either originating from some device or targeted against it.
- *The iIRS module* provides information regarding the devices that are in the scope of the TMS, their importance, the vulnerabilities existing on devices, events of device compromises, as well as network topology and reachability information. Notably, some of these information elements could be sourced from other components, especially the Device profile repository, however the iIRS module synthesises individual information elements served by the Device profile repository into more comprehensive representations, hence it was chosen to retrieve the data from the iIRS module for optimization purposes.
- *The eVDB module* provides information on the detected vulnerabilities, including their impact, underpinning the assessment of the impact that vulnerabilities may have on the trust level of the affected device.
- *The Device profile repository* provides information on the cases that a device is removed from the system and when the device health is restored after a compromise (i.e. the malware is removed or "clean" versions of the operating system/firmware are installed).
- *The TMS*, acting as a trusted peer entity, provides trust assessments which are combined by the receiving TMS instance with the own device trust estimations, to synthesize a comprehensive trust score.

The TMS, in turn, publishes information regarding changes in the trust level of the devices through the Cyber-Trust information bus. This information can be exploited as follows:

- Cyber-Trust operator and end-user interfaces may use this information to generate alerts, especially in the cases of noteworthy trust demotion.
- Defence mechanisms, and in particular the iIRS can exploit this information to apply or disable restrictions in network traffic.
- The Device repository updates its own database, guaranteeing information consistency and dissemination of the trust level to any other interested component.
- Peer TMSs can use this information to update their trust assessments.

Figure 4.4. TMS: Outgoing information flows depicts the TMS outgoing information flows.

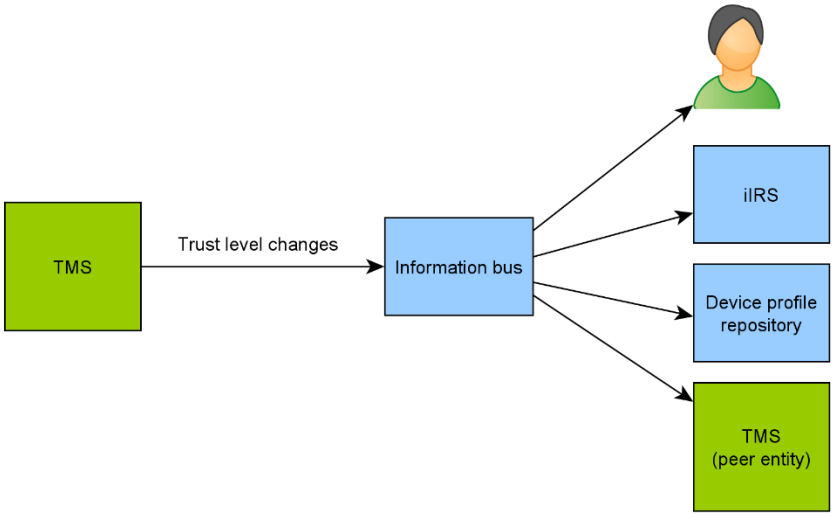


Figure 4.4. TMS: Outgoing information flows

4.2 Trust computation

As described in subsection 3.1, the TMS synthesizes a comprehensive trust score, taking into account the following aspects for a device:

- Its *status*, i.e. the health state of the device and the existence of vulnerabilities.
- Its *behaviour*, i.e. the observed elements of network traffic involving the device, as well as data regarding in-device activity (number of processes, disk I/O and so forth)
- The *risk* associated with the device, i.e. the impact of any value demotion of the device, both towards the loss of assets (data and services) hosted on the device as well as towards the potential use of the device as a stepping stone for further attacks against the infrastructure, after some compromise permitting code execution.
- The *peer TMS trust assessments* for this device.
- The trust relationships between the owner of the device hosting the TMS (which coincides with the owner of the protected infrastructure, e.g. smart home or SOHO) and the owner of the device whose trust is assessed.

In this subsection, we describe in detail the methods used for computing the different dimensions of the device trust, and synthesizing these dimensions into a comprehensive trust score.

4.2.1 Computation of the status-based trust score

The trust-based score of a device *D* comprises the following two aspects:

- the *integrity* aspect, which relates to whether the software components of the device (firmware, operating systems and generic software applications) are integral or have been tampered with; this

status aspect is denoted as $SBT_I(D)$. When some device has been detected to be compromised, the TMS sets $SBT_I(D)$ to zero. $SBT_I(D)$ is restored to one when the health of a device is explicitly designated to be restored (typically through manual intervention).

- the *vulnerability* aspect, which relates to whether the software bears weaknesses which can be exploited to compromise the device. In this context, only vulnerabilities having a *network* or *adjacent* attack vector [63] (i.e. vulnerabilities that can be exploited remotely) are considered. Each vulnerability has an associated impact score, expressing the impact of the vulnerability, taking into account the effect that it may have on the value of the device as well as the exploitability of the vulnerability [63]. Therefore, the overall vulnerability impact metric for device *D*, denoted as *OVIM(D),* can be calculated as

$$OVIM(D) = \sum_{v \in vulnerabilities(D)} ne(v) * \frac{im(v)}{10} \quad (1)$$

where:

  - *vulnerabilities(D)* is the set of vulnerabilities present on device *D*;
  - *ne(v)* is equal to 1 if vulnerability *v* is remotely exploitable or zero, otherwise and
  - *im(v)* is the impact metric for vulnerability *v*; the value of *im(v)* is divided by 10 to normalize its range into [0, 1], since the CVSS specification [63] designates a range [0, 10].

The value of *OVIM(D)* can be arbitrarily high, depending on the number of vulnerabilities present on D. Its value can be normalized in the range [0, 1] to produce the status score related to the aspect of vulnerabilities for device *D*, denoted as $SBT_V(D)$, by employing equation (3).

$$SBT_V(D) = 1 - e^{-OVIM(D)} \quad (2)$$

The effect of the value of *OVIM(D)* on $SBT_V(D)$ can be regulated by multiplying the value of *OVIM(D)* in the exponent of equation (3) by an amortization factor *saf*; higher values of *saf* will lead to a faster convergence of $SBT_V(D)$ to the value of 1.

The value of SBT_V is modified when new vulnerabilities are associated with the device and when vulnerabilities are mitigated (e.g. by installation of a patch, removal of the vulnerable software components etc.).

Finally, the partial status-based scores $SBT_I(D)$ and $SBT_V(D)$ are combined to formulate an overall status-based trust assessment for device *D*, which is denoted as *SBT(D)*. This is accomplished using equation (3).

$$SBT(D) = SBT_I(D) * SBT_V(D) \quad (3)$$

According to equation (3), if the device is compromised, its status-based trust score will be equal to zero (since factor $SBT_I(D)$ will be equal to zero); if the device is not compromised, its status-based trust score will be determined by the overall impact of vulnerabilities on the device.

4.2.2 Computation of the behaviour-based trust score

The behaviour-based trust score for a device *D* comprises three distinct aspects:

- *Compliance*, corresponding to whether *D* in accordance to some rules which describe benign behaviour for the particular device; the score for this behavioural aspect is denoted as $BBT_C(D)$. This aspect mainly applies to network traffic, and in this context the MUD specification is the prevalent approach [61], defining compliance through a set of rules designating the allowed traffic flows. When *D* sends traffic that does not adhere to such rules, it is flagged as *non-compliant* and $BBT_C(D)$ is set to zero. However, the non-compliance penalty should not remain indefinitely, since the deviation may be coincidental: for instance, the system administrator could issue a command initiating a non-compliant traffic flow, in an otherwise benign system. To guard against cases of indefinite demotions of compliance-related trust scores, the TMS restores $BBT_C(D)$ at some specific rate, which is moderated through a respective system parameter, $TSRR_{compliance}$. Should the device continue to exhibit non-compliant behaviour, $BBT_C(D)$ will be again set to zero.

- *Normal behaviour*, corresponding to whether the observable aspects of resource usage metrics exhibited by the device fall in the range that is typically exhibited by the device, as determined by the collection and classification of historical device behaviour data; the score for this behavioural aspect is denoted as $BBT_N(D)$. When abnormal behaviour is detected, the TMS decreases $BBT_N(D)$; the deduction made to $BBT_N(D)$ is equal to the degree of deviation from the nominal metrics: in particular, the degree of deviation is computed as

$$deviationFactor = \frac{detectedMaxMetricValue - nominalRangeHighEnd}{detectedMaxMetricValue} \quad (4)$$

  In the case that the deduction results to a negative value for $BBT_N(D)$, the value of $BBT_N(D)$ is reset to zero. Similarly to the case of compliance, the value of $BBT_N(D)$ is gradually restored at some specific rate, to guard against coincidental deviations (e.g. an unanticipated update operation or a backup operation producing higher bulks of data transfers than nominal volumes; or when a device is under a DDoS attack, the network metrics -and probably CPU metrics- may deviate from the respective nominal values), the TMS restores $BBT_N(D)$ at some specific rate, which is moderated through a respective system parameter ($TSRR_{nominality}$). Should the device continue to exhibit deviant behaviour, $BBT_N(D)$ will be repetitively reduced and thus maintained at low levels.

- *Malicious activities*, corresponding to the detection of attacks being launched from *D*; the score for this behavioural aspect is denoted as $BBT_M(D)$. When launching of attacks is detected, the TMS sets $BBT_M(D)$ to zero. Contrary to the cases of $BBT_M(D)$ and $BBT_M(D)$, the TMS *does not* restore the value of $BBT_M(D)$, since the launching of an attack is deemed improbable to be coincidental. $BBT_M(D)$ can only be restored when the health of a device is explicitly designated to be restored (typically through manual intervention).

The TMS synthesizes the values pertaining to the different aspects of the behaviour-based trust dimension into a single, comprehensive score for behaviour-based trust, which is denoted as $BBT(D)$. The value of is computed according to formula (5):

$$BBT(D) = BBT_C(D) * BBT_N(D) * BBT_M(D) \quad (5)$$

According to formula (5), a major demotion the score of any of the behavioural aspects leads to a low value for the behaviour-based trust dimension.

4.2.3 Computation of the associated risk-based trust score

The risk-based trust score dimension combines the technical probability that a machine is compromised with the level of the damage that would be sustained to the owner of the machine/infrastructure as a result of this compromise, to accommodate a business-oriented security aspect, in line with the information system risk assessment model [64], [65].

To this end, in order to compute the associated risk-based trust score for machine *D* the TMS employs the following algorithm:

1) It combines the probability that machine *D* is compromised with the perceived impact of the machine compromise, as explicitly entered by the user. This is accomplished using the widely used risk assessment matrix (adapted from [65]) shown in **Error! Reference source not found.**.

Table 4.1. Risk level computation

Severity level	Probability of occurrence				
	Highly probable	Probable	Medium	Remote	Improbable
Catastrophic	Catastrophic	Catastrophic	Catastrophic	Serious	Medium
Severe	Catastrophic	Catastrophic	Serious	Medium	Low
Normal	Catastrophic	Serious	Medium	Low	Negligible
Minor	Serious	Medium	Low	Negligible	Negligible
Negligible	Medium	Low	Negligible	Negligible	Negligible

If the compromise probability is given using a numeric value 0 ≤ p ≤ 1, it is converted into a fuzzy label as shown in equation (6):

$$FL_{compromise}(p) = \begin{cases} Improbable & if\ 0 \le p \le 0.1 \\ Remote & if\ 0.1 < p \le 0.3 \\ Medium & if\ 0.3 < p \le 0.6 \\ Probable & if\ 0.6 < p \le 0.85 \\ Highly\ probable & if\ 0.85 < p \le 1 \end{cases} \quad (6)$$

The use of the table in **Error! Reference source not found.** results in the computation of a fuzzy risk labels, constituting the *fuzzy label singular risk assessment for device* which is denoted as $SRA_{FL}(D)$. Fuzzy labels can be converted to numeric ratings by dividing the range [0,1] in a number of strata (0.0; 0.25; 0.5; 0.75; 1.0) and mapping fuzzy labels to the corresponding stratum value. This constitutes the *numerical singular risk assessment for device D*, and is denoted as $SRA_{L}(D)$, i.e.:

$$SRA_L(D) = \begin{cases} 0 & if\ SRA_{FL}(D) = Negligible \\ 0.25 & if\ SRA_{FL}(D) = Low \\ 0.5 & if\ SRA_{FL}(D) = Medium \\ 0.75 & if\ SRA_{FL}(D) = Serious \\ 1 & if\ SRA_{FL}(D) = Catastrophic \end{cases} \quad (7)$$

2) Furthermore, if *D* is compromised in a fashion that allows remote code execution, the machine can be used by the attacker as a stepping-stone to commit attacks against other machines within the protected infrastructure, leading thus to the potential of additional impact being incurred on the organization and, consequently, higher risk levels. To accommodate this dimension, the TMS considers (a) the probability that *D* is compromised in a fashion that allows remote code execution, (b) the neighbouring devices of *D*, (c) the vulnerabilities of each of the neighbouring devices that would permit remote exploitation and the severity of each one of them and (d) the perceived value of each of the neighbouring devices. To compute this dimension, the TMS first computes the cumulative effect on the risk on neighbouring infrastructure stemming from the potential compromise as

$$CCEN(D) = \sum_{n \in neighbours(D)} SRA_L(n) \quad (8)$$

The *CCEN(D)* quantity computed according to equation (8) may be arbitrarily high, while it does not consider the base probability that device *D* is compromised. To normalize the *CCEN(D)* in the range [0, 1] and accommodate the probability that *D* is compromised, TMS uses the *CCEN(D)* quantity com and the probability that *D* is compromised (PRC(D)) to compute the *amortized cumulative compromised effect on neighbouring infrastructure* as depicted in equation (9):

$$ACCEN(D) = PRC(D) * (1 - e^{-CCEN(D)}) \quad (9)$$

The effect of the value of *CCEN(D)* on *ACCEN(D)* can be regulated by multiplying the value of *CCEN(D)* in the exponent of equation (9) by an amortization constant *af*; higher values of *af* will lead to a faster convergence of *ACCEN(D)* to the value of *PRC(D)*.

3) Finally, the values of $SRA_N(D)$ and *ACCEN(D)* are combined to compute the overall risk assessment for device *D*. This is performed using the formula in equation (10):

$$ABT(D) = 1 - \left(1 - SRA_N(D)\right) * \left(1 - ACCEN(D)\right) \quad (10)$$

The rationale behind equation (10) is that factor $\left(1 - SRA_N(D)\right)$ represents the "risk-freeness" directly associated with *D*, whereas factor $* \left(1 - ACCEN(D)\right)$ represents the "risk-freeness" indirectly associated with *D*; Thus, the overall "risk-freeness" is the product of the two factors, while the complementary value (1 minus the product) is the total risk, comprising both the direct and indirect dimensions.

4.2.4 Synthesizing the status-based, behaviour-based and associated risk-based scores

The three dimensions of trust, whose calculation was presented in sections 4.2.1-4.2.3 are synthesized, in order to produce a comprehensive trust score, which considers all trust-related aspects of the device. This comprehensive score reflects the *local view* of the TMS computing the trust level, and will be referred to as *local trust assessment* (LTA).

Several methods for the combination of individual trust scores can be employed, as reported in the bibliography. The most widely used ones are:

1. *Simple additive weighting* [66]: according to this method, a weight is attached to each of the dimensions, with the sum of weights being equal to 1. Effectively, for the case of the TMS, three weights w_s, w_b and w_a would need to be defined, associated with the status, behaviour and associated risk, respectively, with $0 \leq w_s, w_b, w_a \leq 1$ and $w_s + w_b + w_a = 1$. Then, the local trust assessment for device D, LTA(D) would be:

$$LTA(D) = w_s * SBT(D) + w_b * BBT(D) + w_a * ABT(D) \quad (11)$$

Considering the values of w_s, w_b and w_a, we may note that the behaviour trust score is based on evidence on the activity of the device; on the other hand, the presence of vulnerabilities on a device, while undesirable, may or may not lead to its compromise (depending on a number of factors such as the reachability of the device or the perceived value of the device for attackers). Consequently, we expect that $w_b > w_s$. Similarly, the associated risk dimension pertains to events that may occur, and correspondingly $w_b > w_a$.

2. Fuzzy simple additive weighting [67]. This is similar to simple additive weighting, however explicit values of trust dimension assessments are first converted to fuzzy labels, e.g. "Low", "Medium", "High" and subsequently synthesized accordingly. A variant of this method is *stratified trust* [68], where each individual trust score is first rounded to a "close" stratum value using the formula

$$stratify_s(t) = \lceil t * s \rceil \quad (12)$$

where *s* is the number of strata used for the stratification.

3. *Multiplicative* [67], where the values of the different trust dimensions (being in the range [0, 1]) are multiplied together to compute the final score; in the case of the TMS this is formulated as

$$LTA(D) = SBT(D) * BBT(D) * ABT(D) \quad (13)$$

Formula (13) considers all dimensions equally, e.g. the demotion to the overall *LTA(D)* score incurred when *SBT(D)* decreases by some factor (say 10%), is equal to the demotion incurred by an equal decrement to the *BBT(D)* score. This however is not in-line with the remark made above, according to which behaviour-based scores should be taken into account more strongly than status-based scores. This can be tackled using mapping functions that map the range [0, 1] of score dimensions to ranges of the form *[lowerBound, 1]*, where $0 \leq lowerBound \leq 1$, attenuating thus the effect of trust demotions in a specific dimension on the overall trust assessment. For instance, the status-based

score could be mapped to the range [0.5, 1] using the formula $SBT_{mapped}(D) = 0.5 + \frac{SBT(D)}{2}$, and subsequently the $SBT_{mapped}(D)$ value would be used in formula (13) instead of *SBT(D)*; under this arrangement, any demotion of the *SBT(D)* score would have half the effect on the value of *LTA(D)* as compared to an equal demotion of *BBT(D)*.

4.2.5 Incorporating the trusted peer TMS-source based trust score

The user *U* owning a specific TMS *T* may have designated a set of trusted TMSs, *TTMS={TTMS₁, TTMS₂, …, TTMSₙ}*, which can be consulted in order to combine the view of an individual TMS instance towards the trust level of the device with the respective views of its peers, in order to synthesize a comprehensive trust assessment. The rationale behind this synthesis is that the view of *T* is only based on the information that *T* has observed (through its notifications from other Cyber-Trust components, as detailed in subsection 4.1), and this information may be incomplete: for example, other TMS instances may have observed attacks or deviations from nominal metrics that *T* has not observed, or, inversely, *T* may have only witnessed some suspicious behaviour whereas other TMS instances may testify that the device's behaviour is typically normal. This can be particularly true in situations entailing mobility, where some wireless communications may be missed due to the distance between the device and Cyber-Trust components (e.g. the Cyber Defence module), or where the device may move across different Smart Home/SOHO environments.

When assessing the trust level of some device *D*, T will retrieve the trust assessments for *D* offered by the elements of TTMS. However, some of them may not be able to offer an assessment for *D*, e.g. because they have no information regarding *D*. Without loss of generality, we will assume that only the first *k* elements of TTMS are able to offer a trust assessment for device *D*; the notation $TA_{TTMS_i}(D)$ will denote the trust assessment offered by $TTMS_i$ for device *D*, where $1 \leq i \leq k$. Furthermore, recall from subsection 4.1 that for each trusted TMS peer $TTMS_i$, a trust level is provided, denoting the degree confidence that the trust assessments of $TTMS_i$ are accurate. This trust level will be denoted as $TL(TTMS_i)$.

The trust assessments offered for *D* by $TTMS_1, TTMS_2, \ldots, TTMS_k$ are synthesized into a comprehensive peer TMS assessment score using the following equation:

$$PTA(D) = \frac{\sum_{i=1}^{k} TA_{TTMS_i}(D) * TL(TTMS_i)}{\sum_{i=1}^{k} TL(TTMS_i)} \quad (14)$$

which moderates the strength that each peer TMS assessment is taken into account in the final score by the trust level of the offering TMS.

Equation (14) considers the recommendation provided by each peer TMSs with a weight equal to the one specified for the particular TMS, without taking into account the probability that the device on which it is hosted is compromised (this includes the case that the TMS itself is compromised, since in this case the machine will run tampered software, demoting at least the device's status-based trust). If the device on which the peer TMS is compromised, the peer TMS may provide falsified trust assessments to serve the attacker's purposes. To defend against such cases, an *effective trust level* computed and considered for each peer TMS $TTMS_i$, which takes into account (a) the trust level assigned to the peer $TTMS_i$ and (b) the trust assessment of the device hosting $TTMS_i$. The effective trust level of $TTMS_i$ will be denoted as $ETL(TTMS_i)$ and is computed as shown in equation (15):

$$ETL(TTMS_i) = TL(TTMS_i) * LTA(Dev(TTMS_i)) \quad (15)$$

where $Dev(TTMS_i)$ denotes the device on which $TTMS_i$ runs on, while *LTA(D)* is the local trust assessment for this device (c.f. subsection 4.2.4). Under this view, equation (14) is rewritten as

$$PTA(D) = \frac{\sum_{i=1}^{k} TA_{TTMS_i}(D) * ETL(TTMS_i)}{\sum_{i=1}^{k} ETL(TTMS_i)} \quad (16)$$

For the particular device *D*, *T* will have computed a score *LTA(D)* based on its own observations (c.f. subsection 4.2.4). *LTA(D)* is then be combined with the peer TMS assessment score formulating the community trust assessment *CTA(D)* using the following equation:

$$CTA(D) = LW * LTA(D) + (1 - LW) * PTA(D) \quad (17)$$

where *LW* is a weight assigned to the local trust assessment (0 ≤ LW ≤ 1): higher values of *LW* indicate that the local trust assessment is taken more strongly into account for the computation of the community trust assessment, whereas lower values of *LW* attenuate the importance of the local trust assessment in favour of the peer TMS trust assessment.

However, equation (17) does not take into account the fact that peer TMSs may either not offer any trust assessment on *D*, or the case that only few trust assessments may be received, which are offered by peer TMS instances with a low trust score (in which case the confidence towards the $PTA(D)$ score will be low). To tackle this issue, *LW* may be computed in an adaptive fashion, depending on the trust levels of the peer TMSs that have offered a trust assessment on D. Under this view, *LW(D)* will computed as:

$$LW(D) = \begin{cases} LW_{min} & if \sum_{i=1}^{k} ETL(TTMS_i) \geq PT_{threshold} \\ (1 - LW_{min}) * \frac{\sum_{i=1}^{k} ETL(TTMS_i)}{PT_{threshold}} + LW_{min} & if \sum_{i=1}^{k} ETL(TTMS_i) < PT_{threshold} \end{cases} \quad (18)$$

Equation (18) considers a minimum weight LW_{min} that will be assigned to the local trust assessment; this value will be used when the cumulative trust level of peer TMSs offering trust assessments on *D* exceeds some threshold $PT_{threshold}$. If, however, the cumulative trust level of peer TMSs is below the value of $PT_{threshold}$, the value of *LW(D)* increases, decreasing correspondingly the weight assigned to peer TMS assessment score, since the latter is deemed to be of low confidence. Under this view, equation (17) is modified as:

$$CTA(D) = LW(D) * LTA(D) + (1 - LW(D)) * PTA(D) \quad (19)$$

where *LW(D)* is computed as listed in equation (18).

4.2.6 Incorporating user-to-user trust relationships and computing the final trust score

The trust assessment of a device, as computed in subsection 4.2.5 is an objective measure, synthesizing the status, behaviour and associated trust dimensions observed by the TMS, as well as the views of trusted peer TMSs[3]. In the final step, the TMS takes into account the trust relationships between the users of the Cyber-Trust platform, and in particular between the owner of the TMS *U* and the owner of device *D*, who will be denoted as *Owner(D)*. The trust level between two users U_1 and U_2 of the Cyber-Trust platform is denoted as $UT(U_1, U_2)$ and is computed as follows:

$$UT(U_1, U_2) = \begin{cases} 1 & if\ U_i = U_2 \\ ETS(U_1, U_2) & if\ U_1\ has\ explicitly\ established\ a\ trust\ relationship \\ & towards\ U_2\ with\ a\ trust\ level\ equal\ to\ ETS(U_1, U_2) \\ UT_{default} & in\ all\ other\ cases \end{cases} \quad (20)$$

$UT_{default}$ is a parameter of the TMS, which can regulate the trust level assigned to devices for which the user is unknown, and transitively to moderate the access levels granted to this class of devices.

The user trust level computed by equation (20) is finally used by the TMS to moderate the community trust assessment computed by equation (19) and produce the final trust score as follows:

$$TS(D) = CTA(D) * UT(Owner(T), Owner(D)) \quad (21)$$

Equation (18) may be further refined to distinguish between devices that are registered to the Cyber-Trust platform and devices that are not; this is based on the rationale that registered devices are associated with

[3] Actually, the community trust assessment entails a degree of subjectivity, stemming from (a) which TMSs have been chosen to be trusted (b) the degree of confidence to those TMSs.

a physical person who is accountable for the activities of the device, and therefore the probability that such a device is deliberately launching attacks is limited. Under this view, equation (20) can be rewritten as:

$$UT(U_1, U_2) = \begin{cases} 1 & \text{if } U_i = U_2 \\ ETS(U_1, U_2) & \text{if } U_1 \text{ has explicitly established a trust relationship towards } U_2 \text{ with a trust level equal to } ETS(U_1, U_2) \\ UT_{default.registered} & \text{if no explicit trust relationship is established between } U_1 \text{ and } U_2 \text{ and } U_2 \text{ is registred to the Cyber-Trust platform} \\ UT_{default.unregistered} & \text{If } U_2 \text{ unknown (not registered to the Cyber-Trust platform)} \end{cases} \quad (22)$$

The result of equation (22) is then used as a user-to-user trust metric *UT(U_1, U_2)* in equation (21).

4.3 Detailed TMS architecture

This section reiterates on the TMS architecture, presenting a detailed view of its components and elaborating on the tasks that each component undertakes and the functionalities it delivers.

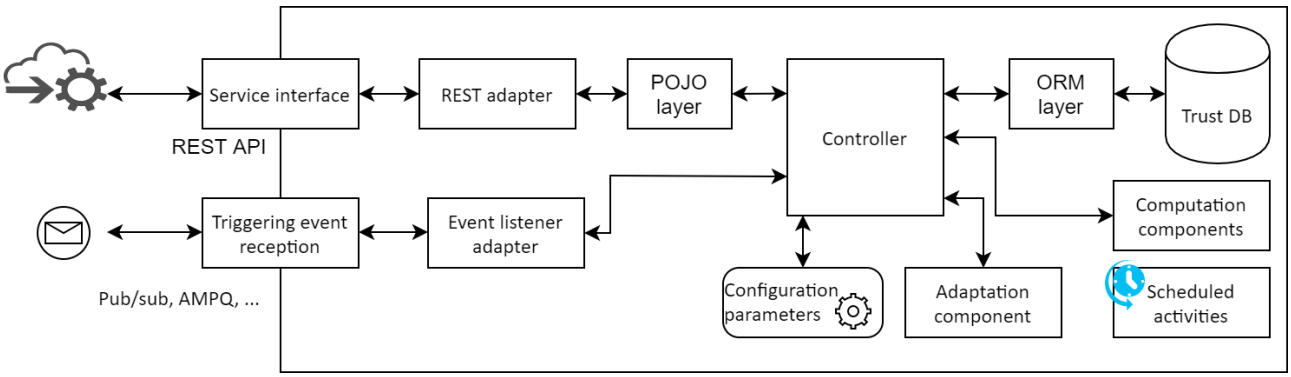


Figure 4.5. Detailed view of the TMS architecture

Internally, the TMS comprises the following modules:

1) The *TMS REST adapter:* this module arranges for intercepting requests according to the REST protocol. This layer arranges for ensuring that (a) requests adhere to the specifications of the REST protocol, (b) are compliant with the implementations realized within the TMS (c) appropriately extracting parameter values from requests so as to forward them to the appropriate implementation of the requested functionality and (d) intercepting responses from the functionality implementations, and formatting response elements according to the specifications of the REST protocol, ensuring proper delivery of response data to the clients.
2) The *triggering event reception component*, which arranges for (a) subscribing to the proper channels of the information bus (b) intercepting messages from channels to which subscriptions have been established, (c) ensuring that intercepted messages are compliant with the Cyber-Trust asynchronous message specifications (d) extracting and verifying message digital signatures and (e) locating the appropriate implementation that is associated with the handling of the relevant message category and delegate the handling of the message to the identified implementation.
3) The *trust database* (*Trust DB)* realizes persistent storage for the TMS; this component stores information about the devices and their attributes, as well as any other piece of information needed for the operation of the TMS (peer TMSs, trusted users, trusted devices and so forth). This module also hosts the historical data of trust computations.
4) The *controller* component is the collection of code elements that realizes the handling of functionalities that either realize functionalities requested through the REST adapter or functionalities related to the handling of asynchronous messages. Effectively, the controller component implements the business logic behind the TMS, and arranges for maintaining the

contents of the *trust* database up to date, by updating elements stored therein according to the information flows intercepted by the TMS.

5) The *computation components*: the computation components implementing the trust score computation procedures described in subsection 4.2. Computation components mainly process input stemming either from incoming information flows or retrieved from the trust DB, however in certain cases they necessitate information from other Cyber-Trust platform components, notably from the eVDB; this information is fetched as needed.
6) The *adaptation components*, which arrange that the TMS operation is tailored to the particular capabilities and particularities of its environment. While at an initial design stage this component was mainly planned to adjust the trust computation procedures according to the resource capabilities of the hosting platform (e.g. use more lightweight trust computation algorithms, as well as offload computational procedures to more resource-rich trusted peers in device environments with limited processing power, memory or power), this was found to be unnecessary, since the resources needed by the trust computation algorithms are generally low). The only element of the TMS that could necessitate adaptive behaviour is the internal caching of vulnerability information, sourced from the eVDB, to avoid re-fetching of the same information and consequently improve performance. While the amount of memory needed for the storage of data for each vulnerability is low, in environments where the number of vulnerabilities present in all devices observed by the TMS is very high and the available memory is constrained, the amount of memory allocated to the caching of vulnerability information should be accordingly limited. This task is undertaken by the adaptation component, which manages the vulnerability data cache accordingly.
7) The *scheduled activities* component performs tasks of periodic nature, which are triggered when specific periods of time have elapsed. The TMS entails four periodic tasks:
    a. The task replenishing the $BBT_C(D)$ score of devices, when some devices have been found to exhibit non-compliant behaviour (c.f. 4.2.2).
    b. The task replenishing the $BBT_N(D)$ score of devices, when some devices have been found to deviate from their nominal behaviour (c.f. 4.2.2). Notably, when the period at which this task should be performed coincides with the periodicity of the task replenishing the $BBT_C(D)$ score of devices, the two tasks may be merged to promote efficiency.
    c. The *peer TMS assessment refresh* task, which arranges for fetching updated device assessment from trusted peer TMSs.
    d. The *historical data purging and anonymization task*, which arranges so that historical trust computation data are purged or anonymized after the expiration of the data retention period.

    Notably, periodic tasks (a)-(c) listed above are performed *frequently*, i.e. at intervals ranging from minutes to a few hours, while task (d) is of considerably lower frequency, being performed at weekly/monthly basis. Consequently, different techniques have been employed to implement task scheduling, with tasks (a)-(c) being realized as process threads, where task (d) being realized as a separate process, which is scheduled using the relevant operating system facilities (e.g. *cron* in Unix or *task scheduler* in Windows).

8) The *configuration parameters* which provide two types of information:
    a. *Information about the environment that the TMS runs in*; notable pieces of this type of information are the data used for connecting to the information bus and the eVDB (endpoint and credentials), as well as information about the trusted Cyber-Trust components, from which messages will be received and processed through the information bus (component names and certificates, needed for validation of the digital signatures).
    b. *Information about aspects of the TMS operation*, which mainly pertains to parameters used by TMS components to realise trust score computation (e.g. weights of different trust dimensions or amortization factors – c.f. subsection 4.2), or periods at which scheduled activities should be performed (only for period tasks realized as threads, since the related information for the historical data purging and anonymization task is specified to the operating system facility employed for process scheduling).

9) *The POJO*[4] *component* is a collection of utility classes supporting the interaction of the TMS with the REST adapter. POJO component elements effectively realize the *view* element of the Model-View-Controller framework [69] and present REST clients with plain-data limited views of the internal Java objects, which entail the whole information required by the model, appropriately encapsulated through methods.
10) *The ORM layer*, which intervenes between the controller and the trust database, arranging for aligning the object-oriented view at the application level with the relational view at the database level. The ORM layer is implemented using the Hibernate Object-Relational Mapping framework (c.f. 2.6).

[4] POJO is an acronym for *Plain Old Java Object*

5. Attack scenarios

In this section we review attack scenarios against which the TMS will be evaluated. These attack scenarios are classified under two broad categories:

1. Attacks against the TMS trust and risk computation mechanisms, in the sense that malicious devices attempt to lead the TMS to the formulation of inaccurate assessments. For these attacks, the evaluation will focus on the capability of the TMS to be resilient against them and achieve to discard falsified input and discern suspicious behaviour, and ultimately produce accurate assessments.
2. Attacks against the infrastructure protected by the Cyber-Trust architecture, i.e. devices within the target environment (smart home, SOHO, industry and so forth). In this context, the evaluation will assess the degree to which the trust and risk assessments offered by the TMS can support the Cyber-Trust infrastructure and its users towards the effective mitigation of these attacks.

The two categories of attacks are described in the following subsections.

5.1 Attacks against the TMS trust and risk computation mechanisms

The attacks against the TMS trust and risk computation mechanisms include either the provision of falsified data to the TMS or actions that are specially chosen to avoid the assignment of a low trust/high risk score. The literature on trust management systems reports on the following attacks (c.f. section 3.2), which are relevant to the Cyber-Trust project:

- **Self-promotion attacks (SPA)** [13]**.** The malicious node provides good recommendations for itself, in order to increase its trust level and thus gain more access privileges.
- **Bad-mouthing attacks (BMA)** [13]**.** A malicious node provides bad recommendations for a "good" node in order to decrease its trust value and therefore block the trusted device's access to services that it would legitimately be entitled to. Additionally, bad mouthing attacks can be used to flood the Cyber-Trust system with unjustified demoted trust recommendations, with the ultimate goal of forcing the administrators to disable the trust-based security controls, considering them as untrustworthy.
- **Ballot-stuffing attacks (BSA)** [13]**.** A malicious node M_1 attempts to boost the trust of another malicious node M_2 in order to increase the level of trust of M_2 and thus allow it to gain more access privileges.
- **Opportunistic service attacks (OSA)** [13]**.** When the trust of a malicious node starts dropping, it starts acting as a "good" node in order to regain its trust.
- **On-off attacks (OOA)** [13]**.** A malicious node is behaving randomly, sometimes performs well sometimes bad, so that it won't get labelled as malicious.
- **Whitewashing attacks** [14]**.** When a malicious node has very low trust, it discards its identity by leaving the network and re-entering it.
- **Sybil-Mobile attacks** [18]**.** A malicious node creates one or more fake identities in order to manipulate recommendations, promote itself and gain influence over the network.
- **Selective Behaviour attacks** [27]**.** A malicious node is behaving well and bad between different services. For example, well for simple services, but bad for more complex ones.
- **Denial of service** [70]. Attackers may attempt to disrupt the mechanisms underpinning the trust system, through a denial of service attack. Agents performing such attacks typically classified as malicious and nonrational, a fact that significantly encumbers the defence against such attacks. This type of attack is of particular importance in the context of systems that employ trust assessments to make timely decisions, in which case the unavailability of trust assessments, coupled with the need to maintain normal system operation and availability of services offered by the protected

infrastructure to the benign nodes, may lead to the adoption of a loose access control policy, permitting thus the execution of operations that would otherwise be blocked.

5.2 Attacks against the infrastructure protected by the Cyber-Trust architecture

The range of attacks that can be launched against any IoT infrastructure is vast, ranging from simple attempts to use default access credentials or well-known weaknesses to complex multi-stage and/or multi-agent attacks. Project deliverable D2.1 [10] has extensively surveyed the current threat landscape in the IoT environment cataloguing more than 130 types of attacks that may take place in IoT environments. In [10], these attacks are classified under the following ENISA categories [71].

Table 5.1. ENISA threat taxonomy branches relevant to Cyber-Trust

Type of attack
Abuse of authorizations
Abuse of Information Leakage
Compromising confidential information (data breaches)
Denial of Service
Failure or disruption of communication links
Generation and use of rogue certificates
Hoax
Identity theft (Identity fraud/account)
Intercepting compromising emissions
Interception of information
Interfering radiation
Malicious code/ software/ activity
Man-in-the-middle
Manipulation of hardware and software
Manipulation of Information
Misuse of audit tools
Misuse of information/information systems (including mobile apps)
Network reconnaissance and information gathering
Network traffic manipulation
Receiving unsolicited E-mail
Remote activity (execution)
Replay of messages
Session hijacking
Social Engineering
Targeted attacks
Unauthorized activities
Unauthorized installation of software
War driving

Out of these types of attacks, the interception of compromising emissions and the (passive) interception of information, are performed at a very low level or in a totally passive fashion, and consequently cannot be detected by typical security and surveillance tools. Moreover, social engineering is performed at human-to-human communication level, hence no traces are collectable at the network or even the application layer. All other types of attacks generate one or more *observables*, i.e. items that can be observed in the context of an attack related to the specific threat. Notably, observables may occur either when an attack is underway

or after an attack has been successful: for instance, attempts to exploit the threat related to missing or weak implementations of security mechanisms may not generate any observables, however if such a security mechanism is violated, this will allow the attackers to penetrate one or more devices, and after the penetration it is highly probable that observables will be generated, e.g. as traces in log files, or detection of unauthorized programs that have been installed, or high traffic volumes owing to data exfiltration attempts, and so forth. Table 5.2 lists the observables that can be detected for the threat types listed in Table 5.1, as these observables are recorded in D2.1 [10] [5]. According to the above, in section 6.1.2, the capability of the TMS to intercept information regarding the presence of the observables listed in Table 5.2 and suitably incorporate them in trust scores will be examined.

Table 5.2. Observables related to the presence of attacks/security hazards

Observables
abnormal device behaviour
abnormal/large network traffic flows and delays
abnormal physical channel behaviour
abnormal transmission patterns
activation of devices
data encrypted with invalid certificates
degradation/loss of a service
degraded network performance
detection of falsified checksums
detection of MAC/IP/identity conflicts
detection of vulnerable software
device/network instability
emails or pages with malicious code
installation/presence of unauthorized software and libraries
installation/presence of untrusted/vulnerable software and libraries
known malicious payloads
log files contain suspicious entries
malware inflows
presence of malware
resource depletion
resource measurements deviating from historical patterns
specific network packet payloads
specific traffic patterns
unencrypted network packets
unusual device/system actions and behaviour
use of weak cipher suites in network traffic

[5] D2.1 lists observables in a higher level of detail, e.g. “numerous replacements of files in short time”, “abnormal regulation of device”, “abnormal system behaviour”, “unusual behaviour of an IoT device”; in Table 5.2

6. TMS evaluation, tuning and validation

In this section, we report in the evaluation, tuning and validation of the TMS. Firstly, in subsection 6.1 we assess the resilience of the TMS to attacks against the TMS trust and risk computation mechanisms, as well as its ability to contribute to the mitigation of attacks against the infrastructure protected by the Cyber-Trust architecture. Subsequently, in subsection 6.2 the effect that different parameters of the trust computation algorithm have on the trust scores, and consequently on the level of protection that can be offered on the basis of trust assessment is surveyed. Finally, in section 0 the TMS is validated against the relevant KPIs defined in D8.1 [72].

6.1 TMS evaluation

6.1.1 Resilience to attacks against the TMS trust and risk computation mechanisms

In an IoT environment, malicious nodes may directly attack the trust and risk computation mechanisms, aiming to manipulate the trust assessments produced by the TMS and therefore assume a level of control on the operations relying on the trust/risk assessments, notably including access control and notification of users/administrators/security officers. These attacks typically take two forms:

1. Malicious nodes generate falsified data, which they try to feed to the TMS as input; this data may either be high-level assessments that will be fed to the TMS as peer TMS trust/risk assessments, or lower-level information, such as notifications reporting that some device's behaviour is found to be malicious/benign, that its health status is promoted or demoted, or any other data that the TMS is known to expect and process.
2. Malicious nodes adopt behaviour patterns that aim to conceal their malicious activities and therefore increase the probability that they are assigned a higher trust level score than the one that would otherwise be assigned to them.

TMS system proposals and reviews have reported on different types of attacks belonging to the two abovementioned categories [13], [14], [18], [27], [70]. In the following paragraphs, we elaborate on the resilience of the TMS algorithms presented in section 4 against each of these attacks.

6.1.1.1 Self-promotion attacks

Self-promotion attacks (SPA) [13] fall under the category of falsified data provision attacks; in such an attack the malicious node attempts to feed the TMS with data that would boost its own trust levels, either reporting on own benign behaviour, or providing trust assessments for itself that report high trust levels.

The TMS architecture and algorithms presented in section 4 are resilient to this type of attacks since:

1. High-level trust assessments are only sourced from TMSs that are designated as *trusted*, therefore trust assessments provided in any way by arbitrary malicious devices are not accepted or processed by the TMS. Furthermore, in case that a trusted peer TMS $TTMS_c$ is compromised and it provides falsified trust assessments for the device $Dev(TTMS_c)$ that it runs on as an effect of this compromise, these trust assessments will be taken into account with a demoted weight (c.f. subsection 4.2.5), significantly limiting thus the effect that these falsified trust assessments will have on the final trust assessment for $Dev(TTMS_c)$.
2. Lower-level information that is used for trust calculation (e.g. notifications reporting that some device's behaviour is found to be malicious, or regarding the device's health/integrity status) are received through the message bus, encapsulated in digitally signed messages, whose source, authenticity and integrity is verified. The only acceptable sources are the authoritative components of the Cyber-Trust platform for the generation/provision of relevant data (e.g. the iIRS, the CyberDefense module and the Device profile repository); therefore, messages that have been injected by some malicious node *MN* aiming to boost its own trust will be rejected and will not be considered for the computation of the trust assessment for device *MN*.

6.1.1.2 Bad-mouthing attacks

Bad-mouthing attacks (BMA) [13] fall under the category of falsified data provision attacks; in this type of attacks, malicious node *MN* provides falsified trust assessments and/or low-level data that support trust computation for a benign node *BN* in order to decrease the trust value of *BN* and therefore block *BN*'s access to services or resources that it would legitimately be entitled to. Additionally, BMAs can be used to flood the Cyber-Trust system with demoted trust recommendations for benign devices, aiming to lead the administrators to disable the trust-based security controls, considering them as untrustworthy.

The TMS architecture and algorithms presented in section 4 are resilient to BMAs since:

1. High-level trust assessments are only sourced from TMSs that are designated as *trusted*, therefore trust assessments provided in any way by arbitrary malicious devices are not accepted or processed by the TMS. Furthermore, in case that a trusted peer TMS $TTMS_c$ is compromised and it provides low trust assessments for some benign device *BD*, these trust assessments will be taken into account with a demoted weight (c.f. subsection 4.2.5), significantly limiting thus the effect that these falsified trust assessments will have on the final trust assessment for *BD*.
2. Lower-level information that is used for trust calculation (e.g. notifications reporting that some device's behaviour is found to be malicious, or regarding the device's health/integrity status) are received through the message bus, encapsulated in digitally signed messages, whose source, authenticity and integrity is verified. The only acceptable sources are the authoritative components of the Cyber-Trust platform for the generation/provision of relevant data (e.g. the iIRS, the CyberDefense module and the Device profile repository); therefore, messages that have been injected by some malicious node *MN* aiming to lower the level of trust for some benign device *BD* will be rejected and will not be considered for the computation of the trust assessment for *BD*.

6.1.1.3 Ballot-stuffing attacks

Ballot-stuffing attacks (BSA) [13] fall under the category of falsified data provision attacks; in this type of attacks, M_1 attempts to boost the trust of another malicious node M_2 (potentially an accomplice of M_1) in order to increase the level of trust of M_2 and thus allow it to gain more access privileges.

The TMS architecture and algorithms presented in section 4 are resilient to BSAs since:

1. High-level trust assessments are only sourced from TMSs that are designated as *trusted*, therefore trust assessments provided in any way by arbitrary malicious devices are not accepted or processed by the TMS. Furthermore, in case that a trusted peer TMS $TTMS_c$ is compromised and it provides falsified elevated trust assessments for some malicious node M_2, these trust assessments will be taken into account with a demoted weight (c.f. subsection 4.2.5), significantly limiting thus the effect that these falsified trust assessments will have on the final trust assessment for M_2.
2. Lower-level information that is used for trust calculation (e.g. notifications reporting that some device's behaviour is found to be malicious, or regarding the device's health/integrity status) are received through the message bus, encapsulated in digitally signed messages, whose source, authenticity and integrity is verified. The only acceptable sources are the authoritative components of the Cyber-Trust platform for the generation/provision of relevant data (e.g. the iIRS, the CyberDefense module and the Device profile repository); therefore, messages that have been injected by some malicious node M_1 aiming to promote the level of trust for some malicious device M_2 will be rejected and will not be considered for the computation of the trust assessment for M_2.

6.1.1.4 Opportunistic service attacks

Opportunistic service attacks (OSA) [13] fall in the category of behaviour pattern adoption, aiming to the deceive the TMS into the calculation of higher trust levels. In the context of an OSA attack, malicious nodes observe their trust assessments and, when that starts dropping, they adopt a benign node behaviour so as to reinstate their trust to a higher level.

The TMS architecture and algorithms can successfully mitigate opportunistic service attacks since, firstly, malicious nodes are not able to observe their trust assessments. Furthermore, if a malicious node is detected

to launch attacks, the TMS demotes its behaviour-based trust level to zero, and this trust level is not reinstated, until the system administrator manually declares that the node health has been restored. Obviously, the system administrator will not take such action for a malicious node. Should the node exhibit abnormal activities, such as excessively high network traffic its trust level is again demoted. While the anomaly-based demotion is temporary (as noted in subsection 4.2.2, the $BBT_N(D)$ score is replenished at some rate R), an appropriate adjustment of the trust demotion and trust replenishment rates will maintain the behaviour-based score dimension of malicious nodes at low levels. Notably any mixture of detected attacks and detected anomalies will demote the behaviour-based trust to zero, with no replenishment provisions.

6.1.1.5 On-off attacks

On-off attacks (OOA) [13] fall in the category of behaviour pattern adoption, aiming to the deceive the TMS into the calculation of higher trust levels. In the context of such an attack, a malicious node is behaving randomly, switching between benign and malicious behaviours, aiming to avoid the assignment of low trust level.
The TMS architecture and algorithms can successfully mitigate on-off attacks since, if a malicious node is detected to launch attacks, the TMS demotes its behaviour-based trust level to zero, and this trust level is not reinstated, until the system administrator manually declares that the node health has been restored. Similarly to the case of OSA (c.f. 6.1.1.4), the system administrator is not expected to take such action for a malicious node. If the malicious node refrains from committing known attacks, but rather it simply creates excessive load in the network, this will be flagged as an anomaly and will lead to the demotion of the node's trust score. Again, demotion levels in the case of anomalies and anomaly-based score replenishment may be appropriately tuned so that behaviour-based scores remain at low levels.

6.1.1.6 *Whitewashing attacks*

Whitewashing attacks [14] fall in the category of behaviour pattern adoption, aiming to deceive the TMS into the calculation of higher trust levels. In the context of these attacks, a malicious node realizing that its trust value has dropped to a very low-level attempts to increase it by assuming a new identity; this is typically accomplished by leaving the network and re-joining it.

To tackle whitewashing attacks, the TMS relies on the information obtained from other components of the Cyber-Trust architecture. Firstly, for devices that are registered to the Cyber-trust platform, a strong identity mechanism is present, and therefore the device identity is maintained, preventing thus the execution of whitewashing attacks. This however mainly applies to the case of registered devices that are compromised, since it is possible that a registered Cyber-Trust platform user takes actions to de-register a device from the Cyber-Trust platform and register it back again, in which case a new identity is indeed assumed. Nevertheless, the fact that the device will be correlated to the physical person performing these purposeful activities supports accountability, and this is expected to deter users from performing such actions.

For devices that are not registered to the Cyber-Trust platform, a number of measures can be taken to protect the infrastructure against whitewashing attacks. Firstly, in environments with high security requirements, it is possible to assign a very low trust level to devices unknown to the Cyber-Trust platform, considerably limiting thus the extent of activities that are allowed for these devices within the protected infrastructure. This can be performed by configuring the user-to-user trust relationship computation procedure to use a very low default trust moderation value for unknown devices, when no known user-to-user relationship exists (c.f. subsection 4.2.6, parameter $UT_{default.unregistered}$). Secondly, measures can be taken by Cyber-Trust platform modules to detect cases that some device is re-joining the platform; indicative such measures are the comparison of the device's MAC address and/or CPE attribute with a list of recently MAC address/CPE attribute values that recently appeared within the protected infrastructure. A positive result provides indications that the device has departed from the network and re-joined it, and therefore suitable correlations, possibly tagged with a confidence degree, can be established.

6.1.1.7 *Sybil-Mobile attacks*

Sybil-Mobile attacks (SMA) [18] fall under the category of falsified data provision attacks. In the context of SMAs, a malicious node *MN* creates a number of fake identities FI_1, FI_2, …, FI_n; then, each of the fake identities attempts to provides false data to the TMS, aiming and deceive it into the calculation of a high trust score for *MN*, allowing it thus to gain more access privileges..

The TMS architecture and algorithms presented in section 4 are resilient to SMAs since:

1. High-level trust assessments are only sourced from TMSs that are designated as *trusted*, therefore trust assessments provided in any way by any of the fake identities FI_1, FI_2, …, FI_n are not accepted or processed by the TMS. Furthermore, in case that a trusted peer TMS $TTMS_c$ is compromised, it can only provide a single recommendation for the device it is hosted on, and this is resolved as described in the case of self-promotion attacks, through a demoted value of the weight that such assessments are considered (c.f. subsection 6.1.1.1).
2. Lower-level information that is used for trust calculation (e.g. notifications reporting that some device's behaviour is found to be malicious, or regarding the device's health/integrity status) are received through the message bus, encapsulated in digitally signed messages, whose source, authenticity and integrity is verified. The only acceptable sources are the authoritative components of the Cyber-Trust platform for the generation/provision of relevant data (e.g. the iIRS, the CyberDefense module and the Device profile repository); therefore, messages that have been injected by any of the fake identities FI_1, FI_2, …, FI_n are rejected and are not considered for the computation of the trust assessment for *MN*.

6.1.1.8 *Selective Behaviour attacks*

Selective Behaviour attacks [14] fall in the category of behaviour pattern adoption, aiming to the deceive the TMS into the calculation of higher trust levels. In the context of selective behaviour attacks, a malicious node *MN* adopts different behaviours towards different services/nodes, aiming to conceal its malicious behaviour towards selected services/nodes under a higher volume of good behaviour indications, stemming from its interactions with other services/nodes.

The TMS architecture and algorithms presented in section 4 are resilient to selective behaviour attacks since:

1. If *MN* launches known attacks against some service/node, when a single attack is detected, its behaviour-based trust will drop to zero, and benign behaviours towards other services/nodes will not affect in any way the level of trust computed for *MN*. Notably, in such cases, *MN's* behaviour-based trust level will only be reinstated if the system administrator manually takes action to state that *MN's* health has been restored, and such actions are not considered probable.
2. If *MN* uses high volumes of traffic towards certain services/nodes, e.g. in the context of denial of service attacks, while it creates low-volume traffic flows against other services/nodes, the CyberDefense modules of the Cyber-Trust platform will still flag the anomalous behaviour, since (a) the volume of data emanating from the device will still be high and (b) anomaly-based analysis may consider device-to-device data volume flow granularity, rather than collective device-to-network data volume flows. Once the anomalous behaviour is flagged, *MN*'s behaviour-based trust will be demoted, and the relevant Cyber-Trust modules will be notified accordingly to take prominent actions against *MN* and limit its access to the network and services.

6.1.1.9 *Denial of service*

Denial of service attacks [70], in this context, refer to the attempts made by attackers to disrupt the operation of the TMS by typically creating a high volume of requests to the TMS, aiming to deplete its resources.

As noted in [70], the predominant defence measure against denial of service attacks is the adoption of distributed calculation and dissemination algorithms, which are less vulnerable to attacks if enough redundancy is employed, such that misbehaviour or loss of one or a few TMS instances will not affect the operation of the trust management system as a whole. The Cyber-Trust TMS adopts a distributed, peer-to-

peer architecture for trust computation and dissemination (c.f. subsection 4.1), and therefore provides an elevated level of resilience against denial of service attacks.

6.1.2 Mitigation of attacks against the infrastructure protected by the Cyber-Trust architecture

In this subsection, we review how the TMS may contribute to the provision of an elevated level of security for the Cyber-Trust protected infrastructure. As discussed in subsection 5.2, the analysis of the defence potential is based on the degree to which the TMS utilises the information produced by other Cyber-Trust platform components (notably the iIRS, the CyberDefense, the device registry and the eVDB modules), to compute comprehensive trust and risk score levels that will be then used for the determination and enactment of appropriate defence measures. Taking these into account, it is clear that the accuracy, timeliness and quality of trust assessments produced by the TMS is highly dependent on the accuracy, comprehensiveness and timeliness with which the aforementioned Cyber-Trust platform components process data and produce notifications, to be intercepted and further processed by the TMS. In the remainder of this section, we will only focus on the utilization of the data/information by the TMS and the effect on the trust score, and not on the accuracy, comprehensiveness and timeliness properties of this information.

Table 6.1 presents details on the mapping between attack/security hazard observables, the relevant information produced by other CyberTrust components and utilised by the TMS, and the effect of this information on the scores computed by the TMS.

Table 6.1. Attack/security hazard observables, relevant information utilised by the TMS and its effect on the scores

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
1.	abnormal device behaviour	Deviations from nominal device behaviour	The TMS will reduce the behaviour-based aspect of the device's trust; the higher the deviation, the bigger the reduction.
2.	abnormal/large network traffic flows and delays	Deviations from the nominal metrics of network flows	The TMS will extract from the notifications it receives the sources of increased network flows and demote their trust level. Notably, large information flow volumes may be flagged for benign devices that are the targets of DoS/DDoS attacks; again, in this case, the irregularity will be flagged for relevant modules to take defence action and for system administrators/security officers to attend to the issue.
3.	abnormal physical channel behaviour	Deviations from the nominal metrics of network flows	The current ingress information flows of the TMS do not include some information about the behaviour of the physical channel; however, irregularities on the physical layer of communication always have an effect on all higher layers, hence deviations from the nominal metrics of network flows for devices operating on top of the affected physical channel will be flagged. The TMS will exploit this information to raise alerts on the trust level of affected devices; these alerts, coupled with the fact that all affected devices will be supported by the same communication medium, will allow system administrators/security officers to infer the root cause of the issue.
4.	abnormal transmission patterns	Attacks against devices	Abnormal transmission patterns are typically associated with known attacks, such as the selective forwarding [73], the SYN flood attack [74] and the Slowloris attack [75]. Once these attacks are identified, the TMS will utilize the information to decrease the

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
			attacker device's behaviour-based score to zero, notifying accordingly the Cyber-Trust platform components that are responsible for triggering the enactment of defence measures.
5.	activation of devices	Deviations from nominal device behaviour; Presence of unauthorized devices	Activation of devices is related mainly to either the unanticipated operation of devices at time points not compliant to their schedule or specifications [76] or the presence of unknown devices within the network perimeter. In the former case, the TMS will exploit the information received about the abnormal behaviour of these devices to demote their trust; in the latter case, the assignment of low trust levels to unknown devices/devices with no known owner is bound to provide a high level of protection against them.
6.	data encrypted with invalid certificates	Device vulnerability flagging or attacks against devices	Data may be encrypted with invalid certificates for different of reasons: firstly, a legitimate certificate may expire and therefore become invalidated; secondly, some malicious nodes may attempt to forge certificates, in order to deceive other devices/users into believing that they actually possess different identities. The first case resolves to a weakness or misconfiguration; when such issues are identified and reported by relevant Cyber-Trust components, the TMS will appropriately demote the status-based trust. On the other hand, the second case resolves to a deliberate attack, hence the flagging and reporting of such cases will trigger the demotion of the behaviour-based score of the device to zero.
7.	degradation/loss of a service or system	Deviations from nominal device behaviour	When a service is lost, or its quality is degraded, the behaviour of the relevant device will deviate from its nominal behaviour, exhibiting ping packet loss, high CPU loads, increased response time etc. Such observables are also exploited by infrastructure monitoring systems (e.g. [60]). Correspondingly, when such deviations are detected and reported by appropriate Cyber-Trust components, the TMS will degrade the corresponding device's trust level, increase the associated risk and publish notifications on these changes. The notifications will be intercepted by Cyber-Trust defence and awareness components, which will trigger appropriate actions.
8.	degraded network performance	Deviations from the nominal metrics of network flows	Degraded network performance is owing either to packet flooding, noise/irregularities on the physical channel layer or hardware faults. Packet flooding will be handled similarly to case (2) "abnormal/large network traffic flows and delays" above. Hardware faults are indistinguishable from noise/irregularities on the physical channel layer; both of these cases are

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
			handled as described in case (3) "abnormal physical channel behaviour" presented above.
9.	detection of falsified checksums	Attacks against devices; device compromises	Falsified checksums are used by malicious nodes either (a) to impersonate other devices or (b) to conceal device compromises. Case (a) (impersonation) constitutes an attack against the nodes/infrastructure, while case (b) (concealment of compromises) corresponds to demotion of device integrity. When the relevant cases are detected and reported by appropriate Cyber-Trust components, the TMS will reduce either the behaviour-based trust score (case a) or the status-based trust score (case b).
10.	detection of MAC/IP/identity conflicts	Misconfigurations or attacks against devices	IP conflicts may correspond to either misconfigurations of the network infrastructure or attempts of a malicious node to impersonate other devices; MAC and identity conflicts are typically correlated to attacks, due to the considerably limited probability that such conflicts occur coincidentally. Consequently, when MAC and identity conflicts are reported, the TMS will degrade the behaviour-based trust of the offending device. In the case of IP conflicts, the action of the TMS will depend on whether the reporting Cyber-Trust platform module will flag the conflict as a misconfiguration or as an attack: in the former case, the status-based trust score of the offending device will be demoted, while in the latter case the behaviour-based trust score of the offending device will be set to zero.
11.	detection of vulnerable software	Device vulnerability flagging	Data vulnerabilities are detected by the relevant Cyber-Trust modules and reported; upon reception of the relevant information, the TMS demotes the trust level of the device, as detailed in subsection 4.2.1.
12.	device/network instability	Device deviation from nominal metrics	Device and network instability may be owing to service misconfigurations, transient or periodic physical channel noise, IP/MAC conflicts and so forth. These phenomena are captured and flagged by network monitoring tools as *flapping hosts/services* (e.g. [60]). Correspondingly, when such deviations are detected and reported by appropriate Cyber-Trust components, the TMS will degrade the corresponding devices' trust level, increase the associated risk and publish notifications on these changes. The notifications will be intercepted by Cyber-Trust defence and awareness components, which will trigger appropriate actions.
13.	emails or pages with malicious code	Attacks against devices	Both cases constitute malicious activities, with the intent to cause harm to devices. When an email/web page containing malicious code is identified, the corresponding server's behaviour-based trust score will be demoted to zero. It is noted here however that the response to such cases should be more elaborate

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
			than simply blocking access to the respective server, since blocking e.g. access to the mail server will result in loss of ability to fetch additional e-mails; the same will be true for the cases that the user users webmail software to access his/her emails, and some e-mail is found to contain malicious code. Special handling of such servers can be achieved by setting an explicit trust level for the them (c.f. subsection 2.5.1).
14.	installation/presence of unauthorized software and libraries	Device vulnerability flagging	The installation or presence of unauthorized software and libraries will be flagged by Cyber-Trust platform components monitoring/attesting device health as a vulnerability; the TMS will therefore decrease the status-based trust of the device.
15.	installation/presence of untrusted/vulnerable software and libraries	Device vulnerability flagging	The installation or presence of unauthorized software and libraries will be flagged by Cyber-Trust platform components monitoring/attesting device health as a vulnerability; the TMS will therefore decrease the status-based trust of the device.
16.	known malicious payloads	Attacks against devices	When the CyberDefense modules detect and flag attacks against devices, the TMS will demote the behaviour-based trust of the originating devices to zero. Note that the trust level of the device in such cases is not automatically replenished and the system administrator/security officer should explicitly set that the corresponding device's health has been restored (c.f. subsection 4.2.2)
17.	log files contain suspicious entries	Attacks against devices; misconfigurations; device vulnerability flagging; presence of malware	Suspicious entries in log files may stem from many different reasons, with the most prevalent ones being attacks against the device (e.g. fail2ban entries [77]) and misconfigurations [78]. If the device runs automated vulnerability scanning software (e.g. password strength test [79]) or malware detection software, log file entries may signify the existence of vulnerabilities or presence of malware. In general, all these incidents may be reported by agents running on the device. Once such reports have been received by the TMS, the corresponding aspects of the device trust scores will be demoted as follows: • If attacks against devices are detected, the attacking device's behaviour-based trust score is demoted to zero (c.f. subsection 4.2.2). • In the event of detection of misconfigurations, device vulnerabilities or malware, the status-based trust level of the device is demoted (c.f. subsection 4.2.1).
18.	malware inflows	Attacks against devices	Malware inflows constitute attempts to compromise devices within the protected network scope. Once these attacks are identified, the TMS will utilize the information to decrease the attacker device's

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
			behaviour-based score to zero, notifying accordingly the Cyber-Trust platform components that are responsible for triggering the enactment of defence measures.
19.	presence of malware	Device vulnerability flagging	The presence of malware will be flagged by Cyber-Trust platform components monitoring/attesting device health as a vulnerability; the TMS will therefore decrease the status-based trust of the device (c.f. subsection 4.2.1).
20.	resource depletion	Deviations from the nominal device metrics	Resource depletion reports indicate that the CPU, disk channels or memory of the device are saturated; these events are identified and reported as deviations from the nominal device metrics, where typical resource device usage varies between a nominal minimum and a nominal maximum. Upon reception of such reports, the TMS will demote the behaviour-based trust of the device (c.f. subsection 4.2.2). The amount of trust demotion is reciprocal to the magnitude of the deviation.
21.	resource measurements deviating from historical patterns	Deviations from nominal device behaviour	When such deviations are flagged, The TMS will reduce the behaviour-based aspect of the corresponding device's trust; the higher the deviation, the bigger the reduction.
22.	specific network packet payloads	Attacks against devices	Detection of specific network packet payloads typically corresponds to positives in signature-based intrusion detection [80], [81], i.e. malicious activities against devices. When the CyberDefense modules detect and flag such attacks, the TMS will demote the behaviour-based trust of the originating devices to zero. Note that the trust level of the device in such cases is not automatically replenished and the system administrator/security officer should explicitly set that the corresponding device's health has been restored (c.f. subsection 4.2.2)
23.	specific traffic patterns	Attacks against devices	Specific traffic patterns are typically associated with known attacks, such as the selective forwarding [73], the SYN flood attack [74] and the Slowloris attack [75]. Once these attacks are identified, the TMS will utilize the information to decrease the attacker device's behaviour-based score to zero, notifying accordingly the Cyber-Trust platform components that are responsible for triggering the enactment of defence measures.
24.	unencrypted network packets	Device vulnerability flagging	Use of weak cipher suites signify weakness of the corresponding devices (actually, of the services run on the devices); when such issues are identified and reported by relevant Cyber-Trust components, the TMS will appropriately demote the status-based trust.
25.	unusual device/system	Deviations from nominal device behaviour	The detection of unusual device/system actions and behaviour may be rooted to a number of causes,

#	Attack/security hazard observables	Relevant information utilized by the TMS	Effect of this information on the scores
	actions and behaviour		including the following: (a) the device has been compromised, (b) the device is under attack and (c) the device is misconfigured; however, no specific root causes have been traced, and reports remain at a generic "unusual behaviour" level, in the form of deviations from nominal behaviour (e.g. high number of processes; high network load; flapping services [60]; and so forth). As a response to such reports, the TMS will decrease the behaviour-based trust score of the device.
26.	use of weak cipher suites in network traffic	Device vulnerability flagging	Use of weak cipher suites signify weakness of the corresponding devices; when such issues are identified and reported by relevant Cyber-Trust components, the TMS will appropriately demote the status-based trust.

6.2 TMS parameter tuning and performance

In this subsection we initially review the different parameters used in the trust computation algorithm and the selection of their values, while subsequently, we perform a performance analysis on the TMS implementation.

6.2.1 TMS parameter setting

The TMS trust computation algorithm described in section 4.2 entails the use of a number of parameters, regulating various aspects of trust score computation as follows:

- Compliance-based trust score restoration rate ($TSRR_{compliance}$),
- Nominal behaviour-based trust score restoration rate ($TSRR_{nominality}$),
- Status-, behaviour- and associated risk-based trust score weights (w_s, w_b, w_a),
- Minimum local weight (LW_{min}) and peer trust threshold ($PT_{threshold}$),
- Default user trust value for registered users ($UT_{default.registered}$),
- Default user trust value for unregistered users ($UT_{default.unregistered}$).

In this subsection we survey the effect of these parameters in the overall trust computation. Since different environments have different security requirements, aspects of the environment in which the Cyber-Trust solution will be deployed are also taken into account.

6.2.1.1 Setting the compliance-based trust score restoration rate parameter

The compliance-based trust score restoration rate parameter, denoted as $TSRR_{compliance}$, regulates the rate at which the compliance-based aspect of the trust score of some device is restored after the detection of non-compliant traffic transmitted from the device. Recall from section 4.2.2 that when non-compliant traffic is detected, the compliance-based trust score is set to zero, which correspondingly sets the overall behaviour-based trust score to zero (c.f. equation (5)), and this is further propagated to the overall trust score (c.f. section 4.2.4).

While in an ideal setting no non-compliant traffic should be detected from any benign device, in a real-world setting this may not hold for a number of reasons:

1. Firstly, non-malicious/coincidental human activities may lead to the generation of non-compliant traffic. A characteristic case is that of a user/system administrator that experiments with commands on a device, with one or more command leading to the generation of non-compliant traffic. Notably, this aspect is related to the level of computer skills possessed by the infrastructure owner: naïve

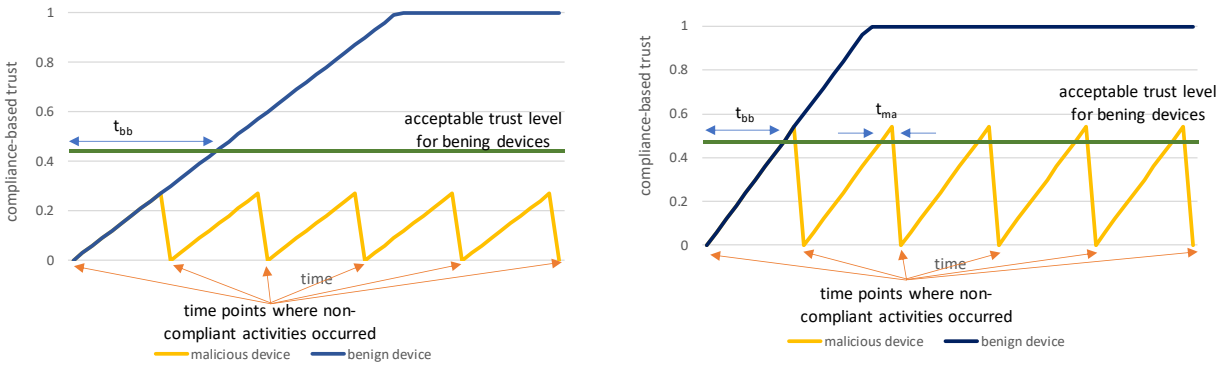
users are not bound to engage in such activities, while technically-aware users are more bound to seek experimentation.

2. Secondly, as noted in section 4.2.2, device compliance is expected to be mainly measured according to the specifications of MUD files [61], however, for most device types, such files are not provided by manufacturers. This may lead infrastructure owners or maintainers to craft their own MUD files, which are bound to be imperfect and thus cause false positives in the process of non-compliant behaviour flagging.
3. Thirdly, MUD files or other compliance specification means should be tailored to the particular installation. This includes allowed devices to communicate with (e.g. a wireless surveillance camera in a smart home should communicate its signals to the security software, and this is bound to be located in a different address in different installations); gateways for communicating to the external world (e.g. for fetching updates); proxies that may operate within the infrastructure, through which communication should mandatorily directed; and so forth.

Considering the specificities of a particular installation, an *acceptable rate/frequency of non-compliant traffic* should be sustained; this is accommodated through the replenishing of the compliance-based trust score, allowing occasional non-compliant functionalities to be "forgotten", however penalizing cases where non-compliant behaviours exceed this acceptable rate/frequency of non-compliant traffic. As noted above, the parameter $TSRR_{compliance}$ regulates the compliance-based trust score replenishment rate.

Figure 6.1 presents a trade-off between blocking time of benign devices and periods at which malicious devices are granted access. In Figure 6.1(a) we can observe that setting a relatively low value for $TSRR_{compliance}$ ensures that malicious devices (which exhibit non-compliant behaviours at a high frequency) always stay below a threshold above which only benign devices should be placed[6]; This is due to the fact that they frequently make non-compliant actions and their compliance-based trust is reset to zero at each occurrence of such actions. On the other hand, if a benign device happens to exhibit non-compliant traffic, the time t_{bb} that is needed until its compliance-based trust is restored above the acceptable level is prolonged.

Conversely, if $TSRR_{compliance}$ is set to a high value (Figure 6.1(b)), the trust restoration curve is steeper, ascertaining that benign devices quickly recover to the acceptable trust level after an accidental occurrence of a non-compliant action; however, malicious (compromised) devices also replenish their trust more quickly, hence they gain the opportunity to act as trusted (or partially trusted) devices for a period of time t_{ma} which may constitute a window of opportunity for conducting attacks.



(a) (b)

Figure 6.1. Trade-off between blocking time of benign devices and periods at which malicious devices are granted access: (a) low compliance-based trust level restoration rate (b) high compliance-based trust level restoration rate

Taking the above into account, the following cases for setting the $TSRR_{compliance}$ trust restoration rate parameter:

[6] The "acceptable level of trust" used in Figure 6.1 is indicative only and does not imply that this value should be used in some setting.

- Within smart homes of naïve users, where users do not engage in experimentation and configurations are fairly standard, and therefore the probability that benign devices perform non-compliant actions is small, $TSRR_{compliance}$ can be set to a small value (e.g. 0.05 / 1h), or even to a close-to-zero value.
- Within the smart homes of technology-aware users, where configurations may not be standard or users may engage in experimentation, leading thus to a higher probability that benign devices perform non-compliant actions, $TSRR_{compliance}$ can be set to a higher value (e.g. 0.15 / 1h).
- In industrial settings, where no malicious activities can be tolerated, configurations and compliance tests are expected to have been crafted by experts and system administrators are bound to intervene and explicitly set trust levels through APIs/UIs (c.f. subsection 2.5.1), $TSRR_{compliance}$ can be set to a zero or near-zero value.

Notably, when infrastructure owners/maintainers discover that some instances of non-compliance are owing to MUD file imperfections or particularities of the installations, the compliance rules may be updated accordingly. Such compliance rules corrections can be coupled with a decrease in the value of $TSRR_{compliance}$.

6.2.1.2 Nominal behaviour-based trust score restoration rate

The nominal behaviour-based trust score restoration rate parameter, denoted as $TSRR_{nominality}$, regulates the rate at which the nominal behaviour-based aspect of the trust score of some device is restored after the detection of device behaviour that deviates from nominal metrics; nominal metrics are typically compiled by observing historical data of the device, and may pertain to CPU load, memory usage, network traffic or any other measurable aspect of the device behaviour. Recall from section 4.2.2 that when deviations from nominal behaviour are detected, the nominal behaviour-based trust score of the device is reduced by a factor corresponding to the degree of deviation, i.e. high deviations result to reductions of a higher magnitude (c.f. equation (4)). The reduction in the nominal behaviour-based trust score is reflected in the overall behaviour-based trust score to zero (c.f. equation (5)), and this is further propagated to the overall trust score (c.f. section 4.2.4).

In an ideal setting, the nominal behaviour metrics would always be observed by healthy devices and all deviations from these nominal metrics would signify some kind of compromise or otherwise suspicious/malicious activity of behalf of the device. However, deviations from nominal behaviour metrics may be circumstantial since:

- Certain legitimate administrative operations performed on a device generate metrics that are deviant from typical behaviour; these operations include backups on remote devices (elevated disk and network activities), full device scans for viruses and malware (increased disk I/O and CPU load), firmware/operating system/software updates (high CPU, disk and network load) etc.
- Legitimate programs or services running on the device may hang, causing high disk, CPU, network or memory resource consumption.
- Some tasks that users run occasionally, e.g. video editing, may case elevated use of resources which can be deemed abnormal, as compared to usual metrics. This aspect depends highly on the type of applications employed by users.

Similarly to the case of non-compliant behaviour detection, a level of deviation from nominal behaviour should be tolerated in an installation, to accommodate the exceptions listed above. This is realized through the reinstatement of the device's nominality-based trust, which allows for occasional deviations from nominal metrics to be "forgotten", however penalizing cases where deviant behaviours exceed this acceptable level of deviations. As stated above, parameter moderating the restoration level of the nominality-based trust score is denoted as $TSRR_{nominality}$.

Analogously to the observations made in subsection 6.2.1.1 for the setting of parameter $TSRR_{compliance}$, the setting of the $TSRR_{nominality}$ parameter entails a trade-off between the time t_{bb} that a benign device which coincidentally exhibited deviant behaviour will remain below the acceptable trust level for benign devices, and the time t_{ma} that malicious (compromised) devices exhibiting deviant behaviour will remain above that level, offering a window of opportunity for more efficient attacks. However, recall that the reduction made

to a deviant device's nominality-based trust level is analogous to the magnitude deviation, hence in the following analysis we elaborate on two cases:

1. In Figure 6.2 the evolution of devices' nominality-based trust scores in cases where deviations of high magnitudes occur is depicted. In Figure 6.2(a) we observe that the nominality-based trust scores of both benign and malicious devices drops immediately under the acceptable trust level for benign devices when a deviation of a large magnitude occurs. Subsequently, as the benign device does not exhibit any more deviations its trust level is restored at a low rate (i.e. $TSRR_{nominality}$ is set to a low value), and after a prolonged time interval t_{bb} it surpasses the acceptable trust level for benign devices. Conversely, the malicious device will exhibit again deviations of low magnitudes and, with the exception of a small peak, will remain constantly under the .acceptable trust level for benign devices.

   Figure 6.2(b) we can observe that when $TSRR_{nominality}$ is set to a high value, the benign device's nominality-based trust score is restored above the acceptable trust level for benign devices more quickly, however the malicious devices again gain windows of opportunity for launching (more successful) attacks.

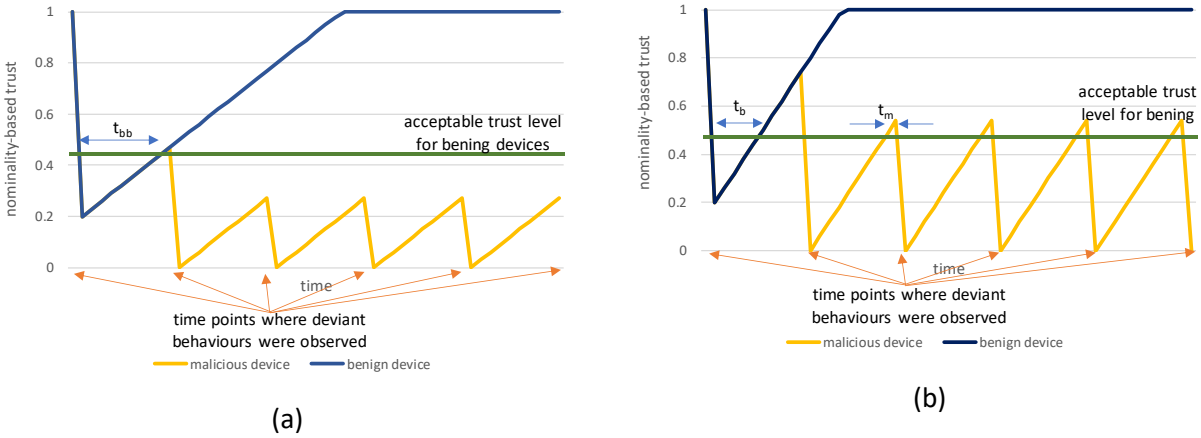


Figure 6.2. Trade-off between blocking time of benign devices and periods at which malicious devices are granted access when deviations of high magnitudes occur: (a) low nominality-based trust level restoration rate (b) high nominality-based trust level restoration rate

2. In Figure 6.3 we can observer the evolution of devices' nominality-based trust scores in cases where deviations of low magnitudes take place. In Figure 6.3(a) a low value is used for $TSRR_{nominality}$ and, according to this setting, the benign device's nominality-based trust score is always over the acceptable trust level for benign devices, however the time needed to reach its maximum value is prolonged; on the other hand, the malicious device's nominality-based trust score is gradually decreased, since deviations from nominal metrics are repeatedly flagged and penalized, until it sinks below the acceptable trust level for benign devices. In Figure 6.3(a) we observe the analogous behaviour for cases where $TSRR_{nominality}$ is set to a high value: here, the benign device's trust score is restored more quickly to its maximum level, however the time that will be needed for the malicious device to attain a trust score less than the acceptable trust level for benign devices will be considerably increased. Note that the malicious device's trust level will drop under the acceptable trust level for benign devices, only if the trust reduction rate is higher than the value of $TSRR_{nominality}$.

Figure 6.3. Trade-off between blocking time of benign devices and periods at which malicious devices are granted access when deviations of low magnitudes occur: (a) low nominality-based trust level restoration rate (b) high nominality-based trust level restoration rate

Taking the above into account, the following cases for setting the $TSRR_{nominality}$ trust restoration rate parameter:

- Within smart homes of naïve users, where the probability of remote operations other than streaming is small (backups are mostly performed in USB-attached disks in these cases), no significant deviations from nominal network metrics are expected hence the $TSRR_{nominality}$ parameter may be set to a small value. Deviations from other metrics (CPU, disk I/O) can be penalized less severely to account for operations like virus scans or updates; this can be accommodated either at the deviation detection level (by adjusting the magnitude of the reported deviation) or at the TMS level (by adjusting the deduction according to the type of deviation reported).
- Within smart homes of advanced users, where some remote operations could be performed, $TSRR_{nominality}$ values could be set to a higher level. Alternatively, a level of awareness for users could be sought for, so as to cater for explicitly setting trust levels of devices involved in operations that are bound to cause deviant behaviour, for the time frame of these operations. This can be accommodated by specialized user-friendly programs, similarly to the case of port knocking in networks [82].
- In industrial settings, where no deviant activities can be tolerated, system administrators are bound to intervene and explicitly set trust levels through APIs/UIs (c.f. subsection 2.5.1) or user-friendly programs, $TSRR_{nominality}$ can be set to a zero or near-zero value.

6.2.1.3 Status-, behaviour- and associated risk-based trust score weights

In the Cyber-Trust TMS model, trust is composed of three separate dimensions, namely status, behaviour and associated risk. The scores for the three distinct dimensions are calculated separately (c.f. subsections 4.2.1-4.2.3) and then synthesized into a comprehensive score as described in subsection 4.2.4, using a *simple additive weighting method* [66]. The simple additive weighting method employs one weight per dimension (w_s, w_b and w_a, respectively), under the restrictions that:

$$\begin{aligned} 0 \le w_s \le 1 \\ 0 \le w_b \le 1 \\ 0 \le w_a \le 1 \\ w_s + w_b + w_a = 1 \end{aligned} \quad (23)$$

As noted in subsection 4.2.4, the behaviour trust score is based on evidence on the activity of the device; on the other hand, the presence of vulnerabilities on a device, while undesirable, may or may not lead to its compromise (depending on a number of factors such as the reachability of the device or the level of the attackers' skills and the perceived value of the device for attackers); even in the case that status-based health

is demoted due a factual compromise (e.g. a firmware hack), some compromises are not used for further exploitation but serve other purposes (challenge, reputation, skill development etc. [83]). Consequently, we expect that $w_b > w_s$. Similarly, the associated risk dimension pertains to events that *may* occur and their impact, and correspondingly $w_b > w_a$. Finally, as noted above, status-based trust may be partially based on factual compromises, while the associated risk dimension pertains only to potential events; under the same rationale, we set $w_s > w_a$ and, overall, $w_b > w_s > w_a$.

Furthermore, we consider the probability that compromises are actually exploited for further attacks or cause of damage, as contrasted to the case of serving other purposes such as challenge and reputation. In contexts of industrial, financial, governmental or similar environments, where the infrastructure is well protected and resources can be allocated for the discovery and punishment of intruders, increasing thus the risk level of prospective intruders, compromises and related attacks are conducted by agents that actually intend to exploit breaches and cause harm; in such a setting, a demoted health status should be taken into account more strongly, and therefore we use a weight setting {w_s=*0.25,* w_b=*0.65,* w_a=0.1}. In smart home environments or other infrastructures where the probability of attacks not aiming to further damage or exploitations is higher, smaller values for the status-based weight can be considered, e.g. a prominent setting may be {w_s=*0.15,* w_b=*0.75,* w_a=0.1}.

6.2.1.4 Minimum local weight (LW_{min}) and peer trust threshold ($PT_{threshold}$)

Bao and Chen [16] in their model conclude that the direct experience of the node (i.e. the local weight) should prevail against the indirect experiences collected as recommendations from other peers (i.e. the peer's trust assessments). The ratio of the indirect experiences' weight to the weight of the direct experiences ranges from 0 to 0.4, depending on the trustworthiness of the recommenders; equivalently, the local weight ranges from to 0.714. In the implementation of the Cyber-Trust TMS we adopt this setting, setting the minimum local weight LW_{min} to 70% (0.7). This is also affirmed by Chen et al. [20], where in the case that peers from whom recommendations are received not totally trusted, the local trust level converges to values $\mu > 0.7$.

Regarding the setting of the peer trust threshold $PT_{threshold}$, we opt for the value of 1.0, to match the trust level of the local TMS's trust to its own assessment.

6.2.1.5 Default user trust value for unregistered users

Unregistered users may be assigned different trust levels, depending on the environment in which the Cyber-Trust solution is deployed and the degree of tolerance of unregistered users/devices in this environment. In particular:

- In smart home environments, it is possible that the smart home owner may want to share some resources with users that are not registered with the Cyber-Trust platform, e.g. allow them to print to the smart home printers or stream/upload photos and videos to the smart TV. In such environments, in order to perform such sharing actions, the default trust value for unknown users should be set above the threshold that allows these actions, but below the threshold that warrants access permissions to devices that the owner wants to keep private, or shared only under more strict safeguards. The actual value of $UT_{default.unregistered}$ will thus be set in accordance to the access and countermeasure policies defined in the iIRS. In [16], unknown devices are assigned a trust level of 0.5, designating ignorance about the device.
- In industrial settings, or other contexts where the presence of unknown devices is not tolerated, $UT_{default.unregistered}$ can be set to zero or a near-zero value, precluding thus any activity on behalf of such devices. Some settings may include mixed environments, e.g. one operational, high-security environment where the presence of unknown devices is not tolerated, and some from "guest" environments that tolerate the presence of unknown devices. In such settings, the use of segregated networks is envisioned [84], with each network portion running its own TMS instance(s), where each TMS would be parametrized in accordance to the needs of its environment.

6.2.1.6 *Default user trust value for registered users*

The parameter $UT_{default.registered}$ regulates the level of trust assigned to devices belonging to users that are registered in the Cyber-Trust platform but for whom the owner of the TMS has not set a trust level. These users (and their devices) can be assigned a higher default level of trust than unknown users, taking into account that they can be traced and made accountable for the activities of their devices; under this view, accountability should act as a deterrent for deliberate malicious actions (although malicious activities carried out without the user's knowledge cannot be precluded).

Still, the default trust level associated with registered users again would depend on the level of tolerance for unknown devices in the environment:

- In smart home environments, where resources may be shared with visitors, $UT_{default.registered}$ may be set indicatively to some value in the range [0.6-0.7], being higher than the indicative value for the $UT_{default.unregistered}$ parameter. Similarly to the case of setting the value of $UT_{default.registered}$, the actual value should be set in accordance to the access and countermeasure policies defined in the iIRS.
- In industrial settings, or other contexts where the presence of unknown devices is not tolerated, $UT_{default.registered}$ can be set to a zero or near-zero value. In mixed environments, again network segregation [84] may be employed, with each network portion running its own TMS instance(s), where each TMS would be parametrized in accordance to the needs of its environment.

6.2.2 Performance issues

Besides the functional behaviour of the TMS and its ability to deliver highly accurate assessments of devices, the performance of the TMS is also of essence, since trust assessments and notifications for trust changes should be delivered in a timely fashion, in order to enable other modules to take prompt actions, putting suitable countermeasures in effect. In this section, we survey aspects of the TMS performance. All measurements presented in this section were obtained on a TMS instance running on a machine equipped with an Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz with 6 hyperthreading-enabled cores, capable of running a maximum of 12 simultaneous threads. Messages were sent to the message bus via a simulator program, and subsequently retrieved and processed by the TMS triggering event receptor (c.f. subsection 4.3).

Figure 6.4 illustrates the message throughput, i.e. the number of messages that can be processed by the TMS per second (average over a mixture of incoming messages), in relation to the amount of parallelism allowed in message processing[7]. In this configuration, only the built-in level 1 cache of Hibernate was enabled [85]. We can observe that message processing throughput is maximized value when 10 cores are employed; this is due to the fact that at this level of parallelism, messages can be efficiently processed concurrently by exploiting the parallelism capability of the underlying hardware, while also allowing the allocation of two cores to the storage subsystem (Hibernate operating on top of a MariaDB instance).

[7] Every incoming message was assigned to a different thread, and the amount of allowed parallelism was programmatically constrained.

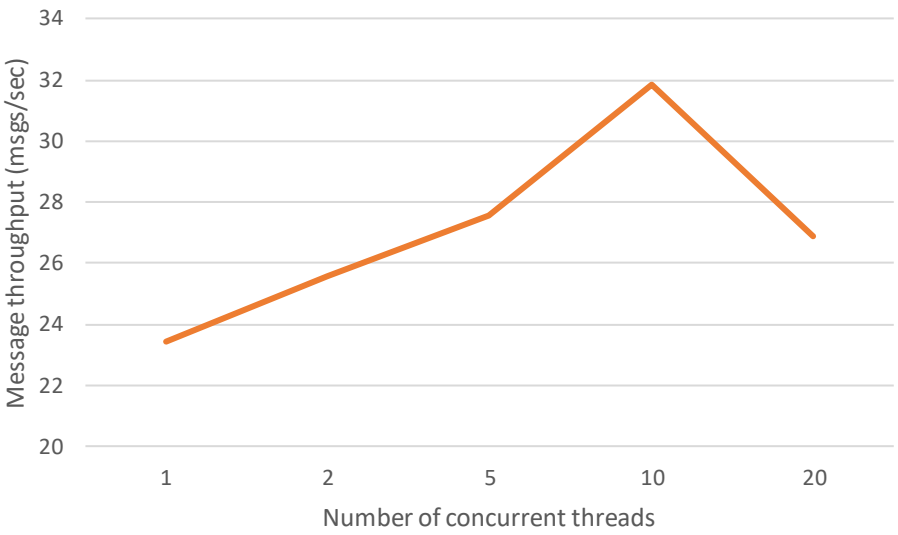


Figure 6.4. Message processing time using only Hibernate level 1 cache

Figure 6.5 illustrates the message throughput in relation to the amount of parallelism allowed in message processing when Hibernate level 2 [85] cache was enabled. We can observe that again message processing time is maximized when 10 cores are employed, however the overall message throughput is less than the one obtained when only level 1 caching was employed. This is attributed to the fact that level 1 cache actually suffices for the purposes of the TMS, and the introduction of level 2 caching adds an overhead for managing this cache, without providing any actual benefits for application performance.

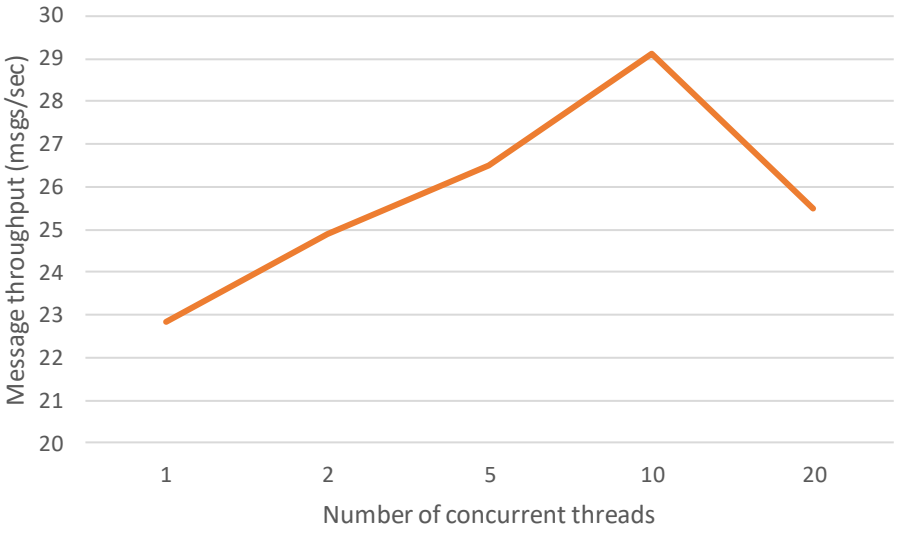


Figure 6.5. Message processing time when the Hibernate level 2 cache is enabled

In both Figure 6.4 and Figure 6.5, throughput is measured at an end-to-end level, i.e. from the time that messages are sent from their sources (the simulator program, in our case) until the time they are processed by the TMS. This interval include (a) the time needed for the messages to be posted to the message bus (b) the time needed for the message bus to internally organize messages and make them available for consumption, (c) the time needed for the TMS to dequeue messages and (d) the time needed by the TMS to process messages and update the devices' trust level. The last factor of message processing time was further analysed to determine the net time needed by the TMS to process messages internally. Table 6.2 depicts the further breakdown of the message processing time within the TMS. In Table 6.2 we can observe that the time needed to decode the message from its JSON or XML representation and validate its digital signature ranges from 1-6 msec, depending on the message encoding and size. Large, XML-encoded messages are the ones

necessitating more processing time at this stage; this particularly applies to messages sourced from the iIRS, where the whole topology of the smart home is described/updated (including aspects of connectivity and vulnerabilities for each device). On the other hand, small JSON-encoded messages that only affect a single device (e.g. messages reporting that non-compliant behaviour is detected for a device) necessitate considerably less time. The same remarks hold for the stage of processing the messages and updating the trust assessment: for instance, topology update messages and this is bound to affect many devices within the smart home/SOHO configuration.

The statistics in Table 6.2 also affirm that while the use of digital signatures in the Cyber-Trust bus messages incurs an extra overhead, this overhead is very small, and is fully justifiable, taking into account the level of resilience against false data injection that it offers.

Table 6.2. Breakdown of message processing time within the TMS

Process	Time needed
Decoding/de-serialization and validation of digital signatures	1-6 msec
Processing of message and update of trust assessments	8-62msec

The TMS also implements a REST API, through which information can be retrieved or set. Figure 6.6 depicts the performance of the REST API in terms of requests per second that can be served (a mixture of requests were used). The Apache Benchmark tool[8] was used to perform these measurements. The throughput of the TMS REST API is much higher than the corresponding metric for messages received through the Cyber-Trust bus, because the REST API implementations retrieve or set individual data items, and do not perform extensive processing. Again, the concurrency level of 10 delivers the higher throughput, since it best matches and exploits the underlying hardware parallelism.

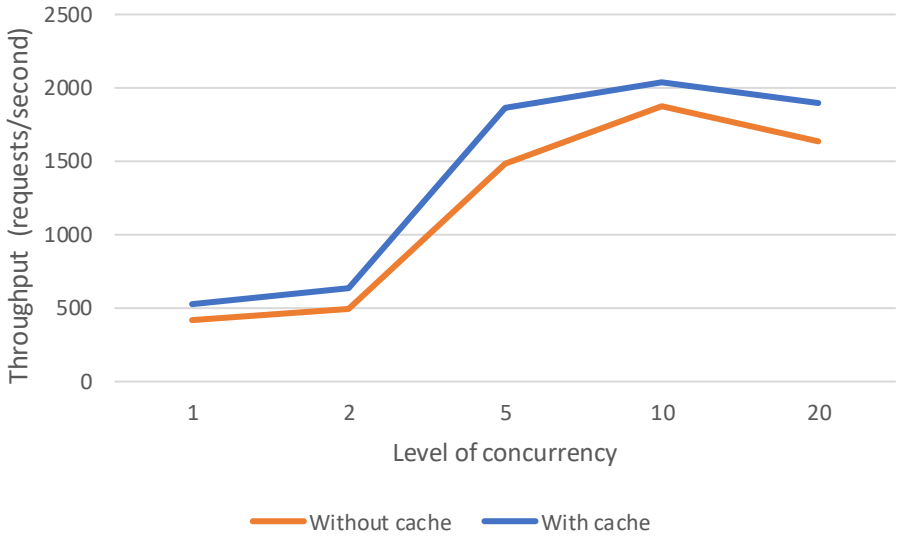


Figure 6.6. Throughput of the TMS REST API

In Figure 6.6 we can observe that when cache is enabled on the REST interface, the performance is higher by a factor ranging from 8% (at concurrency level 10) to 22% (at concurrency level equal to 1). Since the REST API is implemented as a separate process than the bus message processor, and under this implementation the configuration with no cache enabled guarantees better result consistency, the no cache configuration is deployed, taking also into account the very small magnitude of the absolute request processing time (98% of requests are served in less than 5 msec when the concurrency level is set to 10).

[8] https://httpd.apache.org/docs/2.4/programs/ab.html

6.3 TMS validation

Table 6.3 lists the KPIs for the TMS, as identified in Cyber-Trust project deliverable D8.1 [72]. There are two KPIs for the CYBER-TRUST KPIs for the TMS and Administrative module. One is introduced in the DoA, while the second one is proposed by the consortium.

Table 6.3: CYBER-TRUST KPIs for the TMS and Administrative module

KPI-ID	DOA	Name	Measurement	Goal
KPI-5.3	√	TMS effectiveness	The ratio of cyber threats (of varying sophistication and severity) mitigated due to denying access to isolating IoT devices tagged as low-trust/high risk by the TMS.	≥ 2%
KPI-5.6		TMS awareness degree	Percentage of attacks for which a timely warning was issued by the TMS.	≥ 5%

The full assessment of these KPIs necessitates a fully working platform and should take into account (a) the capability of Cyber-Trust traffic scanning components to identify attacks and deviations, (b) the capability of Cyber-Trust device scanning components to identify compromises and vulnerabilities, (c) the timeliness that such incidents are reported to the TMS (d) the capability of the TMS to appropriately adjust the offending devices' trust according to the observed incidents and report these modifications through the Cyber-Trust message bus and (e) the appropriateness of measures taken by Cyber-Trust response components to the announcements of demoted trust issued by the TMS. At the current implementation stage, Cyber-Trust platform integration is an ongoing task and hence the KPIs listed in Table 6.3 cannot be holistically assessed.

In the following subsections we will examine item (d) above, focusing on the response of the TMS to irregular activities that are detected and reported within the protected infrastructure, taking into account the parametrization aspects of the TMS, as analysed in subsection 6.2.

6.3.1 TMS response to detected attacks

Figure 6.7 illustrates the TMS response to detected attacks, in some indicative scenarios:

- The case labelled as *worst case* corresponds to the case that a device owned by the owner of the infrastructure, or a different user completely trusted by the owner, is compromised and launches attacks; furthermore, the same device is known by peer TMSs which testify for its "benignness" (i.e. give a trust score equal to 1), and the weight assigned to the behaviour-based dimension of trust is at its lowest setting, 0.65 (c.f. subsection 6.2.1.3). Finally, no device compromises or vulnerabilities are detected for this device, and therefore its status-based and risk-based scores are set to 1.

  Under these premises, the overall score of the device is set equal to 0.545; the time needed by the TMS to demote its score from 1 to 0.545 is equal to the time required to process the message reporting the misbehaviour. This amount of demotion should be adequate to inhibit access by the misbehaving device to all important resources within the installation (denoted as "critical" by the user).

  As discussed in the TMS performance evaluation section (subsection 6.2.2), the time needed by the TMS to process an incoming message is always less than 70 msec; in particular, since messages reporting attacks are small and focused on the single node launching the attack, the time will be considerably lower than this upper bound, approximately equal to 20 msec. The new assessment of the node trust will be instantaneously reported on the message bus, for the information and perusal of other Cyber-Trust modules.

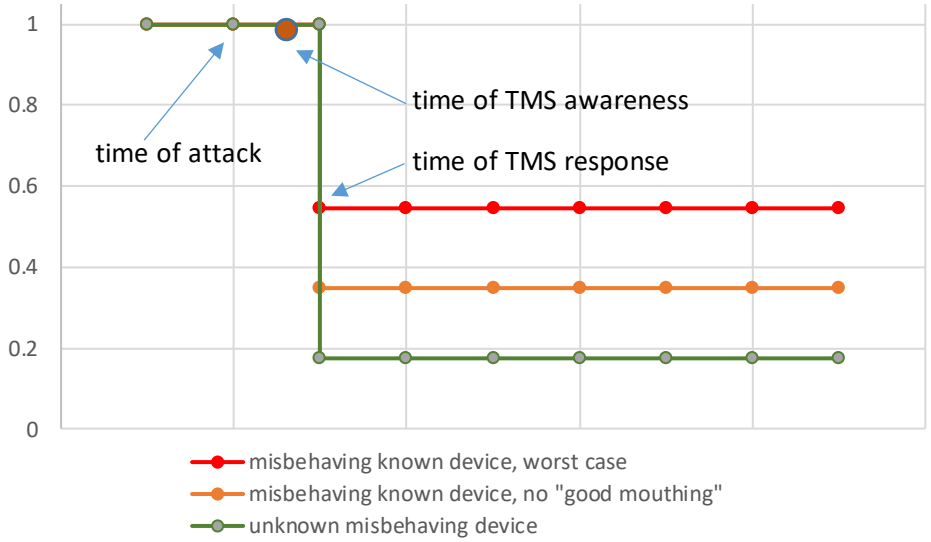


Figure 6.7. TMS response to detected attacks

- A second indicative case is that a known device is detected to launch attacks, however no compromise is detected for this device. Again, the device belongs to the infrastructure owner or a different, completely trusted user, but contrary to the “worst case” discussed above, no peer TMSs testify for the “benignness” of the device; this may happen either because no peer TMSs exist, or peer TMSs have no opinion on the device, or because they are aware of its malicious behaviour. In such circumstances, the TMS will demote the trust score of the device to 0.35; this is deemed adequate to block the device’s access to most resources within the infrastructure. The time needed for the TMS to proceed with trust demotion and notification of the infrastructure is again in the range of 20 msec, which is deemed satisfactory.
- Finally, we consider the case that the device detected to launch the attack is unknown to the infrastructure, i.e. it does not belong to any user registered with the Cyber-Trust platform. In this case, the trust level of the device will be further demoted as discussed in subsections 4.2.6 and 6.2.1.5, and will be set to 0.175. This is a very low value, which would normally preclude all access by the device to resources within the protected infrastructure. In this case too, the time needed for the TMS to demote the trust level of the and notify other Cyber-Trust components is in the range of 20 msec, which is deemed satisfactory.

Figure 6.7 focuses on the reaction of the TMS to behaviours that are positively characterised as “attacks” by relevant Cyber-Trust components; in the context of networks however certain behaviours can be traced which are not positively characterized as attacks but are however harmful for the infrastructure. These behaviours include, among others, denial of service (DoS) attacks as well as distributed DoS attacks (DDoS). Note that while it is possible that the Cyber-Trust components characterize these behaviours as attacks, in which case the discussion made for Figure 6.7 applies, the following analysis pertains to the case that such behaviours are not reported as attacks, but using other tags (and more specifically, deviant behaviours). The TMS can thus foster the defence against such activities, by handling reports on deviations from behaviours that are considered “normal”.

In the context of DoS attacks, a single device generates very high workloads for one or more machines, aiming to deplete their resources and thus render them incapable of offering their services to their legitimate users. When a device launches a DoS attack, the network traffic generated by the device will be considerably higher than the usual values observed for the same device. Correspondingly, a device under a DoS attack will exhibit high network traffic as well as resource usage that deviates from usual metrics for the particular device. Both these behaviours will be monitored by Cyber-Trust components and subsequently be reported via messages through the Cyber-Trust message bus, and therefore the relevant information will be intercepted by the TMS.

Figure 6.8 displays the evolution of a device's trust that is detected to generate abnormally high network traffic and/or have excessively high resource usage metrics, for the same three indicative cases described above. In the context of typical DoS attacks, these deviations are continuous, hence the behaviour-based trust is constantly reset to zero and not replenished. In more detail:

- In the case labelled as "continuous deviations, *worst case*" (a device owned by the owner of the infrastructure, or a different user completely trusted by the owner, is compromised and launches DoS attacks; the same device is known by peer TMSs which testify for its "benignness"; the weight assigned to the behaviour-based dimension of trust is equal to 0.65; no device compromises or vulnerabilities are detected for this device, and therefore its status-based and associated risk-based scores are set to 1), , the overall score of the device is set equal to 0.545; the time needed by the TMS to demote its score from 1 to 0.545 is equal to the time required to process the message reporting the deviation. This amount of demotion should be adequate to preclude access by the deviant device to all important resources within the installation (denoted as "critical" by the user). Notably, this demoted trust level will also apply to the device targeted by the attack: this may result in inability of this device to access resources that are needed for delivering the services it realizes to legitimate user. This can be tackled by setting explicitly the trust level of this device to a higher level, yet again the TMS will be able to report a low behaviour-based trust, hence alerting tools will be able to notify the infrastructure owner and/or the security officer of the demotion.

  As discussed in the TMS performance evaluation section (subsection 6.2.2), the time needed by the TMS to process an incoming message is always less than 70 msec; in particular, since messages reporting attacks are small and focused on the single node launching the attack, the time will be considerably lower than this upper bound, approximately equal to 20 msec. The new assessment of the node trust will be instantaneously reported on the message bus, for the information and perusal of other Cyber-Trust modules.

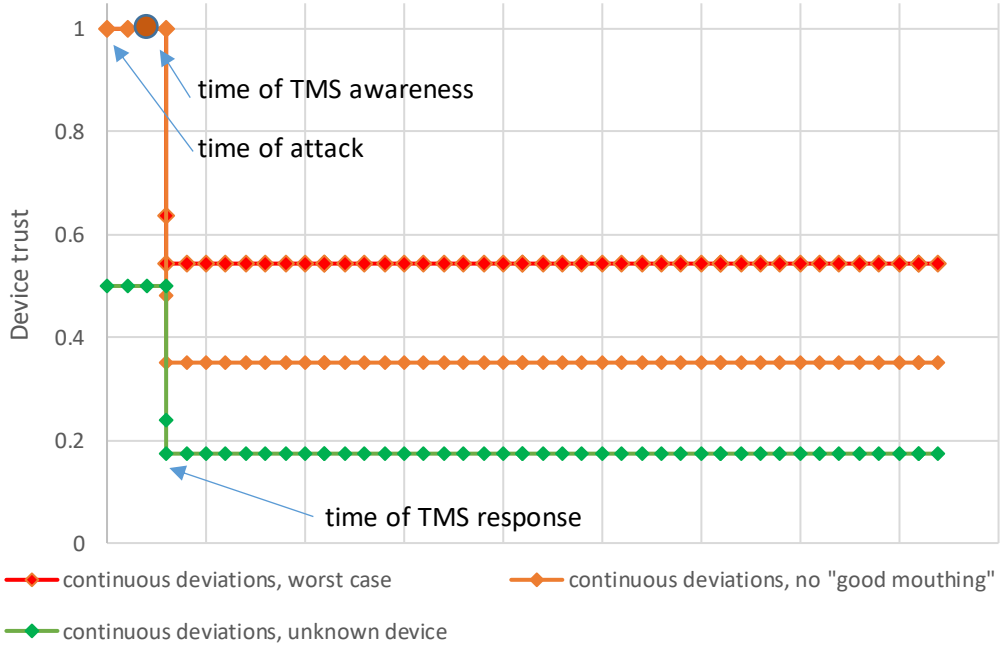


Figure 6.8. Evolution of a deviant device's trust, continuous deviations

- In the case labelled as "continuous deviations, no good mouthing" (a known device is compromised and detected to launch DoS attacks; no compromise is detected for this device; the device belongs to the infrastructure owner or a different, completely trusted user; no peer TMSs testify for the "benignness" of the device), the TMS will demote the trust score of the device to 0.35; this is deemed adequate to block the device's access to most resources within the infrastructure. The time needed for the TMS to proceed with trust demotion and notification of the infrastructure is again in the range

of 20 msec, which is deemed satisfactory. Again, this demoted trust level will also apply to the device targeted by the attack, and this can be tackled as described in the previous case.

- Finally, in the case labelled as “continuous deviations, unknown device” (the device detected to launch the attack is unknown to the infrastructure; no compromise is detected; no peer TMS testifies for the device’s “beningness”), the trust level of the device will be further demoted as discussed in subsections 4.2.6 and 6.2.1.5, and will be set to 0.175. This is a very low value, which would normally preclude all access by the device to resources within the protected infrastructure. In this case too, the time needed for the TMS to demote the trust level of the and notify other Cyber-Trust components is in the range of 20 msec, which is deemed satisfactory.

Figure 6.9 depicts the same three cases in the event that the malicious node combines the DoS attack with the opportunistic attack pattern, i.e. it adopts a good behaviour for some time periods, in order to have its trust level replenished. Notably, the lengthier the time period within which the benign behaviour is adopted, the less efficient the DoS attack, since for these periods the services of the target devices will be offered normally, yet again even the occasional disruptions are undesirable. We can observe here that unknown devices will be assigned a low trust score for most of the time, and therefore their access to most infrastructure resources will be inhibited. In the absence of good mouthing, again accesses to important resources will be normally blocked, but access to resources of lesser importance may be granted. Finally, in the worst-case scenario, there exist periods where the malicious device trust is above 0.66, a threshold that may be associated with benign/trusted devices, however these intervals are small. In all cases, the TMS will be report low behaviour-based scores, hence alerting tools will be able to notify the infrastructure owner and/or the security officer of the demotion.

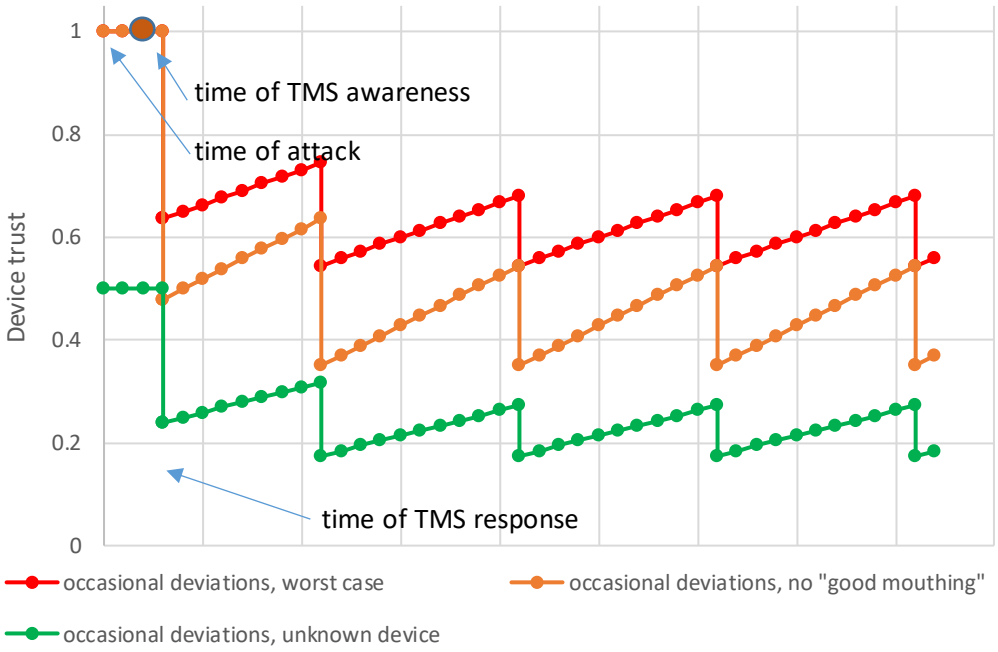


Figure 6.9. Evolution of a deviant device’s trust, occasional deviations

Finally, it is worth noting that in DDoS attacks, it is highly probable that no single attacking device will exhibit a deviant behaviour, because the task of request submission towards the target machine(s) is distributed among numerous cooperating malicious nodes. However, the deviation in the behaviour of the target device(s) will be flagged, and therefore alerts will be issues to the infrastructure owner and/or the security officer of the demotion.

6.3.2 TMS contribution to proactive defence

Proactive defence refers to the measures that can be taken to prevent the launching of attacks against the protected infrastructure, limit the probability of their success or confine the damage that will be sustained in case that an attack is successful.

Proactive defence is based on the identification of vulnerabilities and their interdependencies [86], while the value of assets can also be taken into account [87]. To support proactive defence, the TMS computes and makes available the *status-based trust assessment* and the *associated risk-based trust assessment*. These dimensions can be reported separately by the TMS, and alerting tools can issue notifications to the infrastructure owner and/or security officers regarding the presence of weaknesses or risks within the infrastructure.

In particular, *status-based trust assessments* will report a zero score for devices that are known to be compromised, while for non-compromised devices the status-based trust assessments will depend on the number and impact of vulnerabilities that are present on the device. Hence, the separate reporting of status-based trust assessments will allow infrastructure owners to identify devices needing remediation or patching, in order to restore their health or increase their resilience against attacks, respectively.

7. Conclusions

In this deliverable we have presented the current development status of the Cyber-Trust trust management system (TMS), to be integrated into the operational environment. In particular, the following aspects of the TMS were described:

1. the functionality, technological innovations, and API of the TMS;
2. a reiteration of the review of TMS models, architectures and systems, under the viewpoint of the overall Cyber-Trust architecture;
3. the design of the TMS, with a comprehensive presentation of the trust computation algorithm and a detailed view of the architecture;
4. a compilation of the threats that the TMS should mitigate or support their mitigation, surveying both attacks against the trust computation algorithm and attacks against the infrastructure;
5. an evaluation of the TMS resilience against attacks, complete with parameter tuning, assessment of performance aspects and validation against the goals of the Cyber-Trust platform.

The TMS implementation will be integrated with the other platform components in the context of Task 8.3. The integrated prototype will then be evaluated in a holistic fashion with respect to the Cyber-Trust platform KPIs.

8. References

[1] N. Kolokotronis *et al.*, "Cyber-Trust Project D5.1 State-of-the-art on proactive technologies," 2019.

[2] D.-Z. Sun and L. Sun, "On Secure Simple Pairing in Bluetooth Standard v5.0-Part I: Authenticated Link Key Security and Its Home Automation and Entertainment Applications," *Sensors*, vol. 19, no. 5, p. 1158, Mar. 2019, doi: 10.3390/s19051158.

[3] S. Cuomo *et al.*, "Cyber-Trust Project D5.3: CYBER-TRUST proactive technology tools," 2020.

[4] R. Binnendijk *et al.*, "Cyber-Trust Project D4.4: Architecture and design specifications: final," 2019.

[5] M. Blaze, J. Ioannidis, and A. D. Keromytis, "Experience with the KeyNote Trust Management System: Applications and Future Directions," 2003, pp. 284–300.

[6] M. Petković and W. Jonker, Eds., *Security, Privacy, and Trust in Modern Data Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[7] P. A. Bonatti and P. Samarati, "A uniform framework for regulating service access and information release on the Web," *J. Comput. Secur.*, vol. 10, no. 3, pp. 241–271, Jul. 2002, doi: 10.3233/JCS-2002-10303.

[8] K. Irwin and T. Yu, "Preventing attribute information leakage in automated trust negotiation," in *Proceedings of the 12th ACM conference on Computer and communications security - CCS '05*, 2005, p. 36, doi: 10.1145/1102120.1102128.

[9] N. Li, J. C. Mitchell, and W. H. Winsborough, "Beyond proof-of-compliance: security analysis in trust management," *J. ACM*, vol. 52, no. 3, pp. 474–514, May 2005, doi: 10.1145/1066100.1066103.

[10] C. Vassilakis *et al.*, "Cyber-Trust Project D2.1: Threat landscape: trends and methods," 2018.

[11] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security - CCS '06*, 2006, pp. 336–345, doi: 10.1145/1180405.1180446.

[12] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012, doi: 10.1109/TDSC.2011.34.

[13] J. Guo, I.-R. Chen, and J. J. P. Tsai, "A survey of trust computation models for service management in internet of things systems," *Comput. Commun.*, vol. 97, pp. 1–14, Jan. 2017, doi: 10.1016/j.comcom.2016.10.012.

[14] Y. Ruan and A. Durresi, "A survey of trust management systems for online social communities – Trust modeling, trust inference and attacks," *Knowledge-Based Syst.*, vol. 106, pp. 150–163, Aug. 2016, doi: 10.1016/j.knosys.2016.05.042.

[15] I.-R. Chen, F. Bao, and J. Guo, "Trust-Based Service Management for Social Internet of Things Systems," *IEEE Trans. Dependable Secur. Comput.*, vol. 13, no. 6, pp. 684–696, Nov. 2016, doi: 10.1109/TDSC.2015.2420552.

[16] F. Bao and I.-R. Chen, "Dynamic trust management for internet of things applications," in *Proceedings of the 2012 international workshop on Self-aware internet of things - Self-IoT '12*, 2012, p. 1, doi: 10.1145/2378023.2378025.

[17] N. Djedjig, D. Tandjaoui, F. Medjek, and I. Romdhani, "New trust metric for the RPL routing protocol," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, Apr. 2017, pp. 328–335, doi: 10.1109/IACS.2017.7921993.

[18] F. Medjek, D. Tandjaoui, I. Romdhani, and N. Djedjig, "A Trust-Based Intrusion Detection System for Mobile RPL Based Networks," in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social*

Computing (CPSCom) and IEEE Smart Data (SmartData), Jun. 2017, pp. 735–742, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.113.

[19] F. Bao, I.-R. Chen, and J. Guo, "Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems," in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, Mar. 2013, pp. 1–7, doi: 10.1109/ISADS.2013.6513398.

[20] I.-R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-Based IoT and Its Application to Service Composition," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 482–495, May 2016, doi: 10.1109/TSC.2014.2365797.

[21] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A trust management model based on fuzzy reputation for internet of things," *Comput. Sci. Inf. Syst.*, vol. 8, no. 4, pp. 1207–1228, 2011, doi: 10.2298/CSIS110303056C.

[22] P. N. Mahalle, P. A. Thakre, N. R. Prasad, and R. Prasad, "A fuzzy approach to trust based access control in internet of things," in *Wireless VITAE 2013*, Jun. 2013, pp. 1–5, doi: 10.1109/VITAE.2013.6617083.

[23] C. V. L. Mendoza and J. H. Kleinschmidt, "Mitigating On-Off Attacks in the Internet of Things Using a Distributed Trust Management Scheme," *Int. J. Distrib. Sens. Networks*, vol. 11, no. 11, p. 859731, Nov. 2015, doi: 10.1155/2015/859731.

[24] S. Namal, H. Gamaarachchi, G. MyoungLee, and T.-W. Um, "Autonomic trust management in cloud-based and highly dynamic IoT applications," in *2015 ITU Kaleidoscope: Trust in the Information Society (K-2015)*, Dec. 2015, pp. 1–8, doi: 10.1109/Kaleidoscope.2015.7383635.

[25] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness Management in the Social Internet of Things," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1253–1266, May 2014, doi: 10.1109/TKDE.2013.105.

[26] Z. A. Khan, J. Ullrich, A. G. Voyiatzis, and P. Herrmann, "A Trust-based Resilient Routing Mechanism for the Internet of Things," in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17*, 2017, pp. 1–6, doi: 10.1145/3098954.3098963.

[27] Y. Ben Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the Internet of Things: A context-aware and multi-service approach," *Comput. Secur.*, vol. 39, pp. 351–365, Nov. 2013, doi: 10.1016/j.cose.2013.09.001.

[28] S. K. Prajapati, S. Changder, and A. Sarkar, "Trust Management Model for Cloud Computing Environment," *arXiv.org*, Apr. 2013, [Online]. Available: https://arxiv.org/abs/1304.5313.

[29] X. Wu and F. Li, "A multi-domain trust management model for supporting RFID applications of IoT," *PLoS One*, vol. 12, no. 7, p. e0181124, Jul. 2017, doi: 10.1371/journal.pone.0181124.

[30] J. Yuan and X. Li, "A Reliable and Lightweight Trust Computing Mechanism for IoT Edge Devices Based on Multi-Source Feedback Information Fusion," *IEEE Access*, vol. 6, pp. 23626–23638, 2018, doi: 10.1109/ACCESS.2018.2831898.

[31] M. Mahmud *et al.*, "A Brain-Inspired Trust Management Model to Assure Security in a Cloud Based IoT Framework for Neuroscience Applications," *Cognit. Comput.*, vol. 10, no. 5, pp. 864–873, Oct. 2018, doi: 10.1007/s12559-018-9543-3.

[32] K. Govindan and P. Mohapatra, "Trust Computations and Trust Dynamics in Mobile Adhoc Networks: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 2, pp. 279–298, 2012, doi: 10.1109/SURV.2011.042711.00083.

[33] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the twelfth international conference on World Wide Web - WWW '03*, 2003, p. 640, doi: 10.1145/775152.775242.

[34] Li Xiong and Ling Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 07, pp. 843–857, Jul. 2004, doi: 10.1109/TKDE.2004.1318566.

[35] F. Gómez Mármol and G. Martínez Pérez, “Providing trust in wireless sensor networks using a bio-inspired technique,” *Telecommun. Syst.*, vol. 46, no. 2, pp. 163–180, Feb. 2011, doi: 10.1007/s11235-010-9281-7.

[36] A. J. H. Witwit and A. K. Idrees, “A Comprehensive Review for RPL Routing Protocol in Low Power and Lossy Networks,” 2018, pp. 50–66.

[37] R. Ismail and A. Josang, “The Beta Reputation System,” in *Proceedings of the BLED 2002 Conference*, 2002, [Online]. Available: https://aisel.aisnet.org/bled2002/41.

[38] A. Arabsorkhi, M. Sayad Haghighi, and R. Ghorbanloo, “A conceptual trust model for the Internet of Things interactions,” in *2016 8th International Symposium on Telecommunications (IST)*, Sep. 2016, pp. 89–93, doi: 10.1109/ISTEL.2016.7881789.

[39] J. Golbeck, “Personalizing Applications through Integration of Inferred Trust Values in Semantic Web-Based Social Networks,” in *Proceedings of the Semantic Networks Analysis Workshop*, 2005.

[40] I. D. Chakeres and E. M. Belding-Royer, “AODV routing protocol implementation design,” in *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings.*, 2004, pp. 698–703, doi: 10.1109/ICDCSW.2004.1284108.

[41] X. Huang, R. Yu, J. Kang, and Y. Zhang, “Distributed Reputation Management for Secure and Efficient Vehicular Edge Computing and Networks,” *IEEE Access*, vol. 5, pp. 25408–25420, 2017, doi: 10.1109/ACCESS.2017.2769878.

[42] M. K. Muchahari and S. K. Sinha, “A New Trust Management Architecture for Cloud Computing Environment,” in *2012 International Symposium on Cloud and Services Computing*, Dec. 2012, pp. 136–140, doi: 10.1109/ISCOS.2012.30.

[43] V. Merekoulias *et al.*, “A trust management architecture for autonomic Future Internet,” in *2010 IEEE Globecom Workshops*, Dec. 2010, pp. 616–620, doi: 10.1109/GLOCOMW.2010.5700394.

[44] J. Zhang, R. Shankaran, A. O. Mehmet, V. Varadharajan, and A. Sattar, “A trust management architecture for hierarchical wireless sensor networks,” in *IEEE Local Computer Network Conference*, Oct. 2010, pp. 264–267, doi: 10.1109/LCN.2010.5735718.

[45] G. Karame, I. T. Christou, and T. Dimitriou, “A Secure Hybrid Reputation Management System for Super-Peer Networks,” in *2008 5th IEEE Consumer Communications and Networking Conference*, 2008, pp. 495–499, doi: 10.1109/ccnc08.2007.116.

[46] H. Salah and M. Eltoweissy, “PETRA: Personalized Trust Management Architecture (Application Paper),” in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, Jul. 2016, pp. 287–296, doi: 10.1109/IRI.2016.46.

[47] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, “A reputation-based approach for choosing reliable resources in peer-to-peer networks,” in *Proceedings of the 9th ACM conference on Computer and communications security - CCS ’02*, 2002, p. 207, doi: 10.1145/586110.586138.

[48] Girish Suryanarayana, J. R. Erenkrantz, and R. N. Taylor, “An architectural approach for decentralized trust management,” *IEEE Internet Comput.*, vol. 9, no. 6, pp. 16–23, Nov. 2005, doi: 10.1109/MIC.2005.119.

[49] R. Friedman and A. Portnoy, “A generic decentralized trust management framework,” *Softw. Pract. Exp.*, vol. 45, no. 4, pp. 435–454, Apr. 2015, doi: 10.1002/spe.2226.

[50] I. Dionysiou, H. Gjermundrød, and D. E. Bakken, “GUTS: A Framework for Adaptive and Configureable Grid User Trust Service,” 2011, pp. 84–99.

[51] H. Team, “Haskell and Android,” 2019. https://wiki.haskell.org/Android (accessed Apr. 10, 2020).

[52] M. Srivatsa, L. Xiong, and L. Liu, “TrustGuard,” in *Proceedings of the 14th international conference on*

World Wide Web - WWW '05, 2005, p. 422, doi: 10.1145/1060745.1060808.

[53] V. Thummala and J. Chase, "SAFE: A Declarative Trust Management System with Linked Credentials," Oct. 2015, doi: arXiv:1510.04629v2.

[54] Q. Cao, V. Thummala, J. S. Chase, Y. Yao, and B. Xie, "Certificate Linking and Caching for Logical Trust," Jan. 2017, [Online]. Available: http://arxiv.org/abs/1701.06562.

[55] Cloud Security Alliance, "CTP Data Model and API, rev. 2.13," 2016. [Online]. Available: https://downloads.cloudsecurityalliance.org/assets/research/cloudtrust-protocol/CTP-Data-Model-And-API.pdf.

[56] Cloud Security Alliance, "Cloud Trust Protocol Daemon Prototype," 2016. [Online]. Available: https://github.com/CloudSecurityAlliancePublic/ctpd.

[57] Cloud Security Alliance, "The CTP prototype back office API, rev. 0.2," 2015. [Online]. Available: https://github.com/CloudSecurityAlliancePublic/ctpd/blob/master/client/ CTP-Admin-API.pdf.

[58] B. Škorić, S. J. A. de Hoogh, and N. Zannone, "Flow-based reputation with uncertainty: evidence-based subjective logic," *Int. J. Inf. Secur.*, vol. 15, no. 4, pp. 381–402, Aug. 2016, doi: 10.1007/s10207-015-0298-5.

[59] ThingML, "JArduino project," 2018. https://github.com/SINTEF-9012/JArduino (accessed Apr. 10, 2020).

[60] J. Renita and N. E. Elizabeth, "Network's server monitoring and analysis using Nagios," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Mar. 2017, pp. 1904–1909, doi: 10.1109/WiSPNET.2017.8300092.

[61] D. R. E. Lear, R. Droms, "Manufacturer Usage Description Specification, draft-ietf-opsawg-mud-25," 2018. https://tools.ietf.org/html/draft-ietf-opsawg-mud-25 (accessed Apr. 13, 2020).

[62] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal Defense Policies for Partially Observable Spreading Processes on Bayesian Attack Graphs," in *Proceedings of the Second ACM Workshop on Moving Target Defense - MTD '15*, 2015, pp. 67–76, doi: 10.1145/2808475.2808482.

[63] FIRST, "Common Vulnerability Scoring System version 3.1: Specification Document," 2019. [Online]. Available: https://www.first.org/cvss/specification-document.

[64] E. Hulitt and R. B. Vaughn, "Information system security compliance to FISMA standard: a quantitative measure," *Telecommun. Syst.*, vol. 45, no. 2–3, pp. 139–152, Oct. 2010, doi: 10.1007/s11235-009-9248-8.

[65] United States General Accounting Office, "Information security risk assessment practices of leading organizations," 1998. [Online]. Available: http://www.gao.gov/special.pubs/ai00033.pdf.

[66] I. Kaliszewski and D. Podkopaev, "Simple additive weighting—A metamodel for multiple criteria decision analysis methods," *Expert Syst. Appl.*, vol. 54, pp. 155–161, Jul. 2016, doi: 10.1016/j.eswa.2016.01.042.

[67] M. Modarres and S. Sadi-Nezhad, "Fuzzy Simple Additive Weighting Method by Preference Ratio," *Intell. Autom. Soft Comput.*, vol. 11, no. 4, pp. 235–244, Jan. 2005, doi: 10.1080/10642907.2005.10642907.

[68] N. Griffiths, "Task delegation using experience-based multi-dimensional trust," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, 2005, p. 489, doi: 10.1145/1082473.1082548.

[69] C. Gross, "REST-Based Model View Controller Pattern," in *Ajax Patterns and Best Practices*, Apress, pp. 333–368.

[70] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Comput. Surv.*, vol. 42, no. 1, pp. 1–31, Dec. 2009, doi: 10.1145/1592451.1592452.

[71] ENISA, "Threat Taxonomy," 2016. [Online]. Available: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/enisa-threat-landscape/threat-taxonomy/view.

[72] N. Kolokotronis *et al.*, "Cyber-Trust Project D8.1: Platform evaluation plans," 2020.

[73] L. K. Bysani and A. K. Turuk, "A Survey on Selective Forwarding Attack in Wireless Sensor Networks," in *2011 International Conference on Devices and Communications (ICDeCom)*, Feb. 2011, pp. 1–5, doi: 10.1109/ICDECOM.2011.5738547.

[74] X. Wei, "Analysis and Protection of SYN Flood Attack," 2011, pp. 183–187.

[75] C. M. Mathas *et al.*, "Evaluation of Apache Spot's machine learning capabilities in an SDN/NFV enabled environment," in *Proceedings of the 13th International Conference on Availability, Reliability and Security - ARES 2018*, 2018, pp. 1–10, doi: 10.1145/3230833.3233278.

[76] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications," Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.02041.

[77] M. Ford *et al.*, "A process to transfer Fail2ban data to an adaptive enterprise intrusion detection and prevention system," in *SoutheastCon 2016*, Mar. 2016, pp. 1–4, doi: 10.1109/SECON.2016.7506771.

[78] D. Hadžiosmanović, D. Bolzoni, and P. H. Hartel, "A log mining approach for process monitoring in SCADA," *Int. J. Inf. Secur.*, vol. 11, no. 4, pp. 231–251, Aug. 2012, doi: 10.1007/s10207-012-0163-8.

[79] P. G. Kelley *et al.*, "Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 523–537, doi: 10.1109/SP.2012.38.

[80] H. Holm, "Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?," in *2014 47th Hawaii International Conference on System Sciences*, Jan. 2014, pp. 4895–4904, doi: 10.1109/HICSS.2014.600.

[81] J.-L. Lassez, R. Rossi, S. Sheel, and S. Mukkamala, "Signature based intrusion detection using latent semantic analysis," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1068–1074, doi: 10.1109/IJCNN.2008.4633931.

[82] R. deGraaf, J. Aycock, and M. J. Jacobson, "Improved Port Knocking with Strong Authentication," in *21st Annual Computer Security Applications Conference (ACSAC'05)*, pp. 451–462, doi: 10.1109/CSAC.2005.32.

[83] K. Floyd, S. J. Harrington, and P. Hivale, "The autotelic propensity of types of hackers," in *Proceedings of the 4th annual conference on Information security curriculum development - InfoSecCD '07*, 2007, p. 1, doi: 10.1145/1409908.1409926.

[84] Ning Cai, Jidong Wang, and Xinghuo Yu, "SCADA system security: Complexity, history and new developments," in *2008 6th IEEE International Conference on Industrial Informatics*, Jul. 2008, pp. 569–574, doi: 10.1109/INDIN.2008.4618165.

[85] TutorialsPoint, "Hibernate - Caching," 2019. https://www.tutorialspoint.com/hibernate/hibernate_caching.htm (accessed Apr. 20, 2020).

[86] S. Jajodia and S. Noel, "Topological Vulnerability Analysis," 2010, pp. 139–154.

[87] A. N. Craig, S. J. Shackelford, and J. S. Hiller, "Proactive Cybersecurity: A Comparative Industry and Regulatory Analysis," *Am. Bus. Law J.*, vol. 52, no. 4, pp. 721–787, Dec. 2015, doi: 10.1111/ablj.12055.

