Design and implementation of a trust calculation method for network components

Master of Science Thesis
University of Turku
Department of Computing
Security of Networked Systems
2021
Saverio Turetta

Supervisors:
Jouni Isoaho
Ali Farooq

UNIVERSITY OF TURKU
Department of Computing

SAVERIO TURETTA: Design and implementation of a trust calculation method for network components

Master of Science Thesis, 57 p.
Security of Networked Systems
May 2021

Today's organizations rely on internal or cloud-infrastructures to manage their data and their products. Due to the increasing importance and complexity of these infrastructures, there is the need to implement a reliable way to monitor the trustworthiness of the devices that are part of it. It is important to establish trust within the nodes of a single or multiple security domains to enhance the security of an enterprise's infrastructure.
This thesis aims to develop and evaluate a method to measure and calculate a trust score for each node and security domain of a network infrastructure. This method will be based on a centralized verifier that collects and verifies all the security and performance-based evidence from the nodes that compose the infrastructure. The evidence verification process is based on remote attestation through the use of a hardware root of trust. Moreover, this method allows the exchange of trust scores with other security domains: this enhances inter-domain communication trustworthiness.
The main advantages of this method compared to similar ones found in the literature are the possibility of an inter-domain trust exchange, the use of remote attestation, and its adaptability to work with different kinds of infrastructure. Furthermore, the tests confirmed that the method responds quickly in case of a vulnerable node.

Keywords: trust, network, security domain, security, performances, score, attestation, calculation model

Contents

1 Introduction **1**

1.1 The Importance of Trust 1

1.2 Purpose of the Thesis 3

1.3 Method and Outcome 3

1.4 Structure of the Thesis 5

2 Related Work **6**

2.1 Trust Measurement Methods review 9

3 A new method for trust measurement and calculation **13**

3.1 Attestation 16

3.1.1 TPM Attestation 18

3.1.2 Intel SGX Attestation 22

3.2 Trust Measurement Implementation Details 25

3.2.1 Network infrastructure 28

3.2.2 Evidence for trust calculation 31

3.3 Trust value calculation method 35

4 Proposed solution evaluation **38**

4.1 Evaluation Methodology 40

4.2 Use Cases for trust measurement 42

4.2.1 Policy management system 45

4.2.2 Dynamic packet routes manager 47

4.3 Experiments outcome 48

4.4 Considerations and parallelism with existing solutions 50

5 Conclusion **53**

6 Further works and considerations **56**

References **58**

List of Figures

2.1 Simple representation of 5G slicing [14] 8

2.2 Representation of the infrastructure proposed by Dimitrios et al. [17] 12

3.1 Example of remote attestation flow [24] 16

3.2 EPID-based Intel SGX remote attestation flow [22] 24

3.3 Basic Trust Management System communication flow 27

3.4 Basic framework infrastructure 32

4.1 Use cases' network schema 43

4.2 Detection times graph 50

List of Tables

4.1 Change of the trust score of node A over 3 events 49

4.2 Execution times in seconds on different network sizes 49

4.3 Evaluation Features List 51

4.4 Comparison with existing frameworks 52

1 Introduction

Establishing trust among network devices has become a key process in recent years. In the current literature we can find many examples of the application of trust calculation method.

1.1 The Importance of Trust

The main fields where we can find studies about trust are cloud computing, internet of things and pervasive computing [1][2]. IoT networks, for example is a field where trust calculation is very important [1]. In IoT networks, indeed, we connect many devices (in this case called things) that communicate among them to exchange information based on the network application. One example could be the measurement of environmental data such as the temperature or the humidity of a house. We have various applications of IoT networks, from consumer based ones to enterprise applications. An example of IoT application not related to consumer applications is a network of devices that monitor a structure's stability (bridges, buildings, etc.) [3]. Due to this spread integration of IoT in many fields (also ones where the retrieved information is essentials), we need to know that we can trust the information received.

Another field where trust is important is cloud computing [2]. In cloud computing, indeed, we have a network of computers that communicate with each other and each computer has a different role. In this case, we have many examples of computer networks that involve the transmission of security-sensitive information. That is why there is the need to know if the source of the transmitted information can be trusted and is not malicious. In cloud computing is important to know the status of each device of the network, since a compromised device is a risk for the other devices in the same network. Another problem in cloud computing is establishing trust among different security domain. Indeed potential security risks can also come from other security domains.

In the literature, it is possible to find many examples of cloud computing trust applications applied to 5G infrastructures. In the case of 5G, the application of trust management systems is mainly to ensure trust among network slices [4]. A slice in 5G infrastructure is a logical network portion that is optimized for a specific purpose. Due to this slice subdivision and this heterogeneous use of the 5G network, establishing trust is more challenging than previous mobile networks generations [5].

The last-mentioned big use case of trust in networks is in pervasive computing networks. Pervasive computing, also called ubiquitous computing, are types of embedded devices that are used as a replacement for everyday objects [6]. Examples of those devices can be found in medical devices used to monitor patients' health conditions. In this case, we need to know if those devices can be trusted so we can have reliable data. We can consider this as an IoT subset.

1.2 Purpose of the Thesis

The purpose of this thesis is to develop and evaluate a method to measure and calculate a trust score for each node and security domain of a network infrastructure. This method allows to collect and verifies security and performance-based evidence from the nodes that compose the infrastructure.

One of the objectives of this method is to ensure the integrity of the monitored nodes. Indeed, the evidence verification process is based on remote attestation through the use of a hardware root of trust. The developed method also allows the exchange of trust scores with other security domains: this enhances inter-domain communication trustworthiness.

1.3 Method and Outcome

To calculate trust, multiple methods have been proposed [7]. Those methods take into consideration multiple factors to calculate trust. The factors are decided based of what is the main purpose of the method. This thesis analyzes some of these methods and will find what the difference and the use cases are. With this analysis, will be possible to compare the developed method with others present in the scientific literature and show its mainstay.

This analysis will be used to build a custom trust framework that will be explained more in depth in the next chapters. This framework will be used to calculate trust mainly in cloud computing and proprietary infrastructures. Two types of evaluations will be performed to verify the performances and the effectiveness of this framework. The first type of evaluation consists in six performance tests on two use cases, the second in a comparison of the offered features with the ones offered by similar frameworks.

The framework will be tested using two use cases, one of which inspired to one of the products developed at Ericsson, where this research was conducted. The trust calculation will also be done also using an attestation framework based on TPM (Trusted Platform Module) [8] devices. TPM will provide a root of trust to make sure that the devices and the software running on it it is not tampered. To exchange TPM information to a central verifier (a central device whose primary functions are to collect evidence from network devices and manage network trust), technologies like the Keylime [9] framework will be used.

By using a root of trust [10], it will be possible to securely verify the kernel, the firmware and the software running on the network devices and, based on a white list owned by the verifier decide if the machine is in a trust state or not. One of the main challenges indeed is to be sure that the information retrieved by the nodes is real and that it was not tampered in some ways.

The trust information analyzed by the method developed in this thesis consists of environmental information and performance information. The method will not calculate a binary trust, but a trust index that goes from 0 to 10. The purpose is to provide the network administrator an overview of the network trust status. With this information the administrator (or the system that will retrieve that information) will decide how to act. This method tries to solve also the trust problem over different security domains [11], by calculating the overall domain trust and sharing it with other security domains, always using a root of trust.

1.4 Structure of the Thesis

The structure of the thesis will be as follow. Chapter 1, Introduction, definition of the topic context and of the aim of this thesis. Chapter 2, Related Work, there will be the definition of other trust methods proposed in the literature, and it will be explained how those influenced this research. Chapter 3, A new solution for trust measurement and calculation, detailed description of the method developed, with an overview of the technologies used. Chapter 4, Proposed solution evaluation, description of the evaluation methodology, evaluation with applying of the method to two use cases, and comparing it with similar methods. Chapter 5, Conclusion, final consideration, and wrap up. Chapter 6, Further works, considerations about possible improvements of the developed method.

2 Related Work

In the current literature, many solutions for calculating a trust index in a network of devices have been proposed. In this chapter, there will be a description of some of the most important ones to create this Trust Management system. Moreover, this introduction to existing ways to measure trust in a network of devices allows to better understand the current literature for these paradigms and what it lacks.

The management of device trust is important in many paradigms [1][2]. The most important that we can find are:

- IoT/Pervasive Computing
- Cloud Computing
- 5G Network Infrastructure

IoT devices are used in more and more applications, from wearing devices to health care systems [12]. Due to this wide application (even in critical situations) of these devices, it is important to find a way to establish a level of trust. IoT field, compared to others, is even more challenging due to the multiple constraints (mainly economic and computational) of the devices.

Also, cloud computing is an increasing trend, both in enterprises and also for privates. Indeed, with cloud computing, devices such as servers, memories, and computational power are moved from internal resources to external providers. That's why it is important to have systems that ensure the trust of remote devices [13].

Instead 5G network infrastructures need a way to establish trust due to how 5G infrastructure is made. For 5G, indeed, the concept of trust is more important than the previous generations of mobile networks due to the way it uses network slicing [4]. Network slicing is the technique to subdivide a physical part of the 5G spectrum into various slices, and we can observe a simplified representation in Figure 2.1. The owner of each slice is different, which means that the same physical network is shared among slices of different owners, making it essential to have a method to establish trust among network slices.

To understand better what features a Trust Management System has, the Zheng et al. paper "A survey on trust management for Internet of Things" [7] tries to differentiate some of the existing IoT Trust Management Systems implementations by a features evaluation. The features used in [7] are also valid in the context of this thesis and can also be used to evaluate the proposed method. In [7], more than thirty trust measurement frameworks were analyzed and compared. The comparison was made by choosing ten features, and analyzing which of them each framework has.

The outcome of [7] is a lack of the user's subjective considerations on the systems and a lack of context in the trust evaluation. Moreover, the study found out that there is no existing framework that covers all the evaluation characteristics that the paper considers. This outcome will be taken in consideration during the development of the project and will be useful to help us to fill the gap in the existing Trust Management Systems.

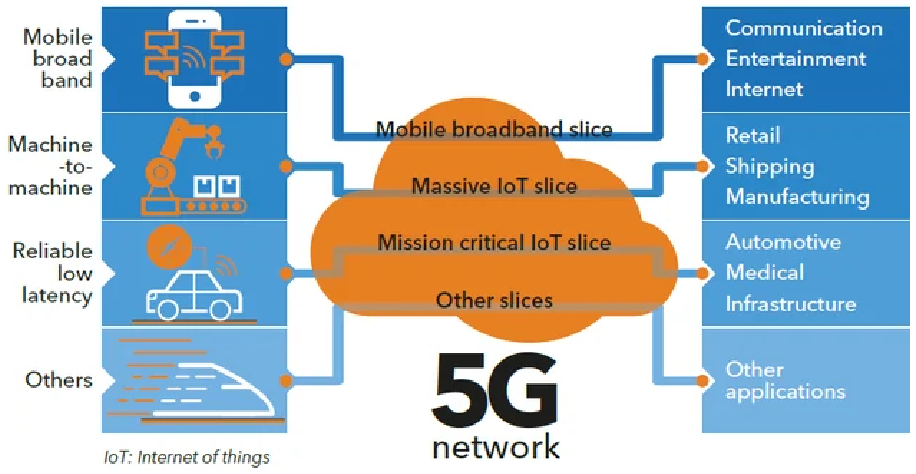

Figure 2.1: Simple representation of 5G slicing [14]

During the literature analysis (Section 4.4), one thing that was noticed is that trust management systems focus mainly either on evaluating the system performances or in evaluating the system security. Trust measurement indeed depends on how the concept of trustworthy system is intended. In some cases, indeed (for example, in IoT), a system is considered more trustworthy if it is faster to accomplish a certain task. In Cloud computing or the 5G infrastructure, a trustworthy system is intended as a secure and tamper-proof system. In my opinion, a good trust management system is a system that knows how to put together both security and performances, and based on the context, knows how to evaluate a trust score more biased on performance or more on security.

Some of the papers analyzed [7][15][16][17] gave some good starting points to decide on what to focus on for the development of this trust management system and how to evaluate it. In the following section, there will be an explanation of some of the papers that gave some ideas for this project. Other than explaining what was implemented in these papers, I will try to explain what parts were interesting and useful to create this Trust Management System.

2.1 Trust Measurement Methods review

In this section, three trust measurement methods will be analyzed. These three were chosen because they cover three different approaches in trust evaluation. Moreover, as stated in the previous section, those three offered some ideas for developing this project.

A trust management approach in pervasive computing

The first is “Performance Evaluation of Trust Management in Pervasive Computing” by Tao et al. [15]. [15] proposes a solution for trust measurement for pervasive computing. In [15] paper, the concept of trust is more based on the performance of the devices. The most interesting part of [15] research is how the trust value is calculated. The research is more focused on the trust value calculation, and there are no security frameworks that blocks the nodes to send wrong trust values, but since [15] solution is for pervasive computing, the author aimed to find a way to calculate the network trust without inferring too much on the computational resources.

The way trust is managed in [15] consists in the observation of the neighbors of each node. Indeed the trust, in this case, is subjective and the result of the observation made from each node. Of course, the author also thought a way to calculate trust of non-neighbors nodes, and that is done through trust recommendations from the other nodes of the network. Also the trust of the neighbors is influenced by other nodes recommendations. The trust calculation here starts when a new node is introduced in the network. At that point the new node does not have any knowledge about neighbors trust since no packets were sent and so no trust measurement from the new node is done. To get an initial trust value, the new node will use values it gets from recommendations of other nodes. By using these recommendations, the node can build an initial trust value, and start to refine it based its further analysis.

To calculate the trust index each node needs to gather some specific information about its neighbors. This information will be subdivided in two main categories: bad actions and good actions. In the use case adopted in [15] the good action was associated with the average throughput and the bad action with the average loss packet ratio. Based on those information and also based on the previous trust value, the new neighbor trust value can be calculated. Here the idea of a subjective trust value is important and is a concept that will also be adopted during the development of this thesis.

A trust management approach in IoT Networks

The second paper analyzed is “Trust Management Framework for Internet of Things” by Yefeng et al. [16]. [16] aims to propose a framework to solve the problem of trust management in a general IoT environment. As the previous paper [15], also [16] see the concept of trust as subjective. An interesting concept introduced in [16] work, is the definition of confidence in a trust measurement framework.

We can see an emphasis in differentiating the concept of trustworthiness and confidence. While the trustworthiness of a device is something that is calculated through the measurement of a system, the confidence corresponds to the amount of uncertainty of the measured data. Know the amount of uncertainty of the data gathered is important at the moment we need to calculate the trust score. Based on the confidence of our data indeed the final trust score for a device will change.

Another concept introduced in [16] is the concept of environment. Indeed we know that the devices for which we are measuring the trust probably are not all part of the same environment. So during the calculation of the trust index we need also to take into consideration the environment of a device. Data measured between two nodes in different environments will also have a lower confidence, while data collected between nodes of the same environment will have an higher confidence. In the framework proposed in [16], the concept of environment will be defined as the security domain [11]. As for the confidence, since the proposed solution is not for IoT devices and we have fewer performance constraints, attestation architectures like TPM will be used to ensure high confidence in every measurement.

A trust management approach in 5G Networks

The third paper is “Security Considerations in 5G Networks: A Slice-Aware Trust Zone Approach” by Dimitrios et al. [17]. The solution proposed in [17] is a security framework that manages the trust in a 5G network. As already explained, 5G infrastructure uses slicing to subdivide the physical network. In each slice, there will be a different type of device with different functions. This means that an attack on devices in a slice can also compromise other slices in the same 5G infrastructure.

To overcome this problem, the research proposes the usage of what they call Security Trust Zones. Each trust zone needs to have some devices (some mandatory and some optional) that monitor the status of the zone and refer the status to a central system in charge to update also other zones. This allow to have trust between 5G slices and avoid that a compromised slice interfere with a healthy one. The representation of the infrastructure proposed by [17] can be observed in Figure 2.2. In [17], no trust index is calculated but it offer a good perspective on how to mitigate security problems among different security domains based on a trust mechanism and for our research this is fundamental since it aims not only to calculate the trust of single devices but also of a group of devices (security domains).

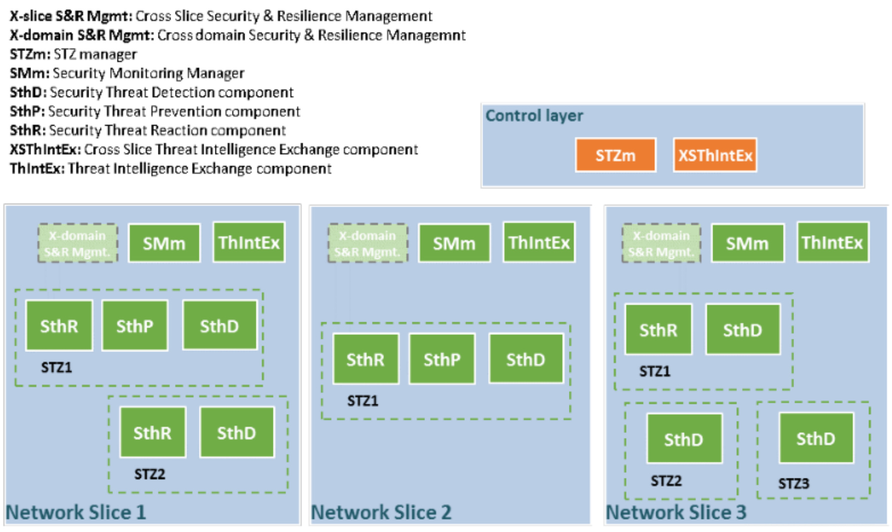
X-slice S&R Mgmt: Cross Slice Security & Resilience Management
X-domain S&R Mgmt: Cross domain Security & Resilience Managemnt
STZm: STZ manager
SMm: Security Monitoring Manager
SthD: Security Threat Detection component
SthP: Security Threat Prevention component
SthR: Security Threat Reaction component
XSThIntEx: Cross Slice Threat Intelligence Exchange component
ThIntEx: Threat Intelligence Exchange component
Control layer
STZm
XSThIntEx
X-domain S&R Mgmt.
SMm
ThIntEx
SthR
SthP
SthD
STZ1
STZ2
STZ3
Network Slice 1
Network Slice 2
Network Slice 3

Figure 2.2: Representation of the infrastructure proposed by Dimitrios et al. [17]

3 A new method for trust measurement and calculation

In this chapter, a new method for trust measurement and calculation will be described. This research was executed in two phases. The first phase aimed to find a method to measure trust by trying to understand what type of evidence is needed from a device to be considered trustworthy. In the second part, the research focused more on finding a formula that allows converting the evidence found in a trust index. The results will be used by the administrator (or an automated system) to manage the network based on the new trust scores. Examples of possible applications of the calculated indexes are to route packets and configure network nodes. Other than the index, the central node will also own the raw data obtained from the collection of the evidence. Those data will be useful for specific index calculation decisions.

The application of this trust measurement solution can be applied also to a multi-domain network architecture [18]. The only requirement is that there is at least one coordinator node that can retrieve information from the network's nodes. The coordinator for some use cases should also be able to configure the security policies of nodes. By changing nodes' configuration the network can be resilient to trust loss. At the same time this method allows having less strict security and routing rules in the network. This allows an increase in the packages transmission speed and in the node access.

A crucial factor in the implementation of this method is to have a trustworthy method to transmit data related to the status of the nodes and to their configuration. Indeed as explained in the previous paragraph, all the data relative to the node should be transmitted to a coordinator node for the verification and decision making.

Other than the transmission, we also need to be sure that the collected information are correct and has not been tampered. To verify it, we need a root of trust [10]. One of the most famous examples of a hardware root of trust is the TPM [8]. The TPM is a hardware chip that hard stores some cryptographic keys and that is capable of measuring the system status.

Using TPM it is possible to know both the current information about a network node and also the health status of the node. An altered status implies a decrease in the trust of the node. To allow us to know if a node is in an healthy state, the coordinator node will own a status white-list for each node and, based on that, will decide if the node was tampered with or not. Once a node is marked as tampered, it is temporarily cut out from the network, and further analysis will be made on it to decide whether to reintegrate it into the network and change its trust index. With this interruption phase, we can ensure the network as a whole remains in a secure state by avoiding to receive malicious packets from a potentially bad node. This also allows avoiding sending potentially sensitive information to the potentially bad node.

All this procedure of collecting information/monitoring nodes' status/send information to a central entity is called remote attestation [19]. There are already attestation solution freely available, and this solution will be based on one of them. The main attestation framework used for the nodes monitoring is Keylime [9]. Keylime is a TPM based attestation framework that allows to monitor a system with a root of trust and to send the information to a verifier, that in our case will be the coordinator and to check that information with a white-list. This framework also allows to send encrypted packets from the coordinator to the nodes and to send instructions to nodes on what to do in case of a compromised node.

The actions taken based on the change of trust of a node are out of the scope of this research since those decisions should be made based on the network type and scope. Indeed this solution will provide some interfaces that can be used from a device that manages the network to change the policies accordingly to the status of the nodes. In the use-cases that will be illustrated later in this thesis, the policy change part will be included to show some examples of usage. That will be also used to measure the effectiveness of this trust evaluation system.

The type of evidence collected from the nodes will be of three types: user defined, vulnerability details (such as installed software CVE score [20] and type), and performance. Those categories will cover three main factors of nodes' trustworthiness, how much the node is secure, how much the node is reliable, and if the node is in a healthy state [21]. Due to the large difference among the collected evidence, other than the trust index, other values extracted for nodes' evidence will be used to make decisions based on the trust score. Details of how all those types of evidence will be collected will be explained in Section 3.2.

In Section 3.1 there will be the definition of attestation, with a focus on the types of attestation frameworks useful for this solution. The two main types of attestation taken into consideration are TPM based attestation and SGX based attestation [22] [23]. Then there will be an explanation of the measurement implementation. In that part will be explained what data are collected and how and why those data are collected. In that part, there will be also explained the types of

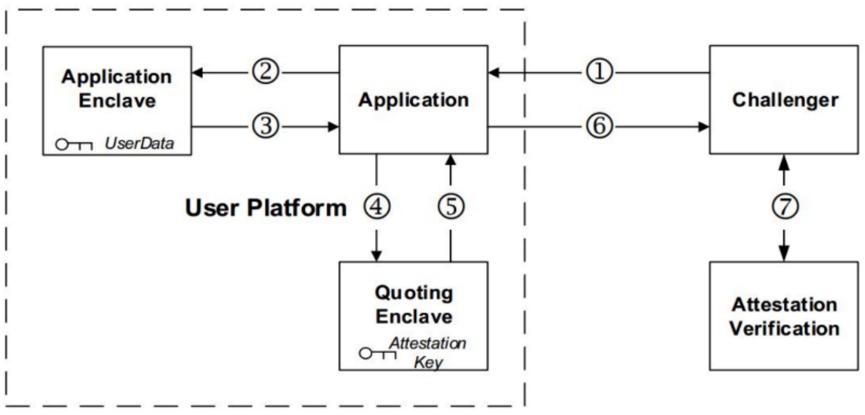

Figure 3.1: Example of remote attestation flow [24]

network infrastructures where it will be possible to use this solution. In Section 3.3 of this chapter, there will be an explanation of how the trust index is calculated and will include all the math involved in the calculation of the index.

3.1 Attestation

Remote attestation is a procedure that verifies the integrity and authenticity of a device. Both software and hardware can be attested [19].

Attestation is important since is fundamental to understand the trustworthiness of a device. Indeed by verifying the integrity and the authenticity of a device we can know its status and whether it was tampered. As tampered device, we refer to a device whose status is not the expected one and could be under attack. Figure 3.1 shows an example of remote attestation flow.

When a device is in a tampered state it should not be considered trustworthy anymore. That's why should be temporary removed from the network for a check and sanitation. Once checked and back in a trusted status after sanitation, the device can be reintroduced in the network. Removing a device from the network is important to ensure other connected devices not to be affected too. In infrastructures that support redundancy, the node can also be substituted with a verified new one.

By exploiting the capability of attestation to verify integrity we are also able to check, at least on attestation of software, what changes happened to the node. We can indeed have a change of node status to another status considered trustworthy as well. One possible change in the system can be a software update or a downgrade to a version allowed by the system.

In this case, we will have a system that will be as well trustworthy and that we can trust more or less than what it was before the change. In the example of the software update or downgrade, we can have a new version that will contain more or fewer vulnerabilities of the previous one. Base on that we will recalculate the new trust index for the new version. In this way, we can have a way to safely measure how much to trust a system.

Attestation can be based on different technologies. The most known ones are TPM and Intel SGX. The main component for attestation is indeed a root of trust that can be hardware or software. A hardware root of trust is always more secure since all the keys are stored in a chip.

The only downside of a hardware root of trusts like TPM is that they can be expensive and take up space. This can be a problem in the case of IoT devices, where the cost and/or the size should be as low as possible [25]. The solution exposed in this thesis is mainly focused on hardware root of trust, so every device that can not have a hardware root of trust will be considered out of scope.

Next in this chapter, there will be an explanation of how TPM and Intel SGX based attestations work [23] [26]. These two indeed are the two main technologies used in this research.

3.1.1 TPM Attestation

TPM (Trusted Platform Module, or ISO/IEC 11889) is a standard that defines a cryptoprocessor. With the integrated cryptographic keys, this cryptoprocessor allows to secure hardware [8].

TPM provides multiple features to secure the hardware [27]. It has an integrated RSA key, used to encrypt data, it has a random number generator [28] and, the most important for this research, it allows remote attestation by creating an hash key that is nearly unforgeable. This hash key is a summary of the software and hardware status. Moreover, each time the hardware or software change their status this hash key is updated [29].

TPM comes in two versions, TPM 1.2 and TPM 2.0. Nowadays, 2.0 is the most frequently used and all the tools developed in this project are based on TPM 2.0. TPM 2.0 is not backward compatible and has more functionalities. For example, TPM 1.2 specification requires SHA-1 and RSA hashing algorithms, while TPM 2.0 requires SHA-256 and SHA-1 as hashing algorithms, RSA, ECC, and NIST for public-key cryptography and AES and HMAC for symmetric-key cryptography [30].

TPM has various applications, for example, one of them is disk encryption. By using TPM indeed the system can recognize if the system is untampered (so trusted) and based on that it will decrypt the disk. Moreover, it is also able to securely store the disk encryption keys [27].

Another possible application (that is the one that will be mainly used in this research) is the platform integrity check. Indeed with TPM it is possible to check whether the system behaves as intended. TPM check, continue from the device boot to the operating system and applications fully start. Measurements performed by the TPM are stored in many Platform Configuration Registers (PCR). These registers contain the hashes that summarize the device status, and that we will use to calculate the device trust [31].

TPM is easy to use as a trusted device nowadays since many PCs motherboards adopt this technology by default or it is easily integrable. Until few years business laptop manufacturers offered the possibility to have the laptop with a TPM module integrated. Due to its availability, (especially in business devices) and its compatibility with all the operating systems, TPM is a good technology to help to calculate the trustworthiness of devices in a network [32] [33].

As explained above, the main functionality of the TPM that we will use in this project is its ability to record hardware and software state changes and to record those changes into some registers called Platform Configuration Registers (PCRs). In each of these PCR registers, the TPM records a hash that it gets from the devices and the software running on the system. The formula used by the TPM to calculate these hashes is:

$$PCR[i]_{t1} = SHA1(PCR[i]_{t0} + \alpha_{t1})$$

where i is the number of the PCR register and α is the new value that the TPM measured for a change in the software or the hardware of the system. The values *t0* and *t1* instead are time variables, so *t0* refers to the old value, and *t1* to the new one. So the new PCR value will be the hash of the sum of the old PCR and the new recorded value. The value of the PCR registers during the system boot is 0 [34]. In the formula SHA1 is used because it is compatible with TPM 1.2 and TPM 2.0. With TPM2.0 other SHA versions can be used (like SHA256).

Since the TPM is a passive device, the system in which it is installed should choose how to interact with it and how to use the provided values. Usually the order of the system and the operating system that starts to interface with the TPM is [35]:

1. UEFI [36]
2. Boot Loader
3. Kernel
4. Applications

UEFI is independent of the operating system and can use the Secure Boot technology to verify, by using the TPM, the authenticity of the system components and of the operating system. It can also block the system boot in case something does not match the white list values. This white list of trusted values usually is set by the OEM, so the end user will be unable to change it with ad-hoc values. Fortunately, there are some free and open source alternatives to UEFI Secure Boot. The most famous is HEADS [37], a firmware designed to be flashed on the BIOS chip and also provide TPM based attestation on PC and servers.

To also enable TPM functionalities soon after the system startup, also the Boot Loader should support the interaction with the TPM to register and monitor the system values. Within Linux systems, the most famous secure Boot Loader is TrustedGRUB2. TrustedGRUB2 is a fork of the GRUB2 Linux Boot Loader that measures the status of system critical components during boot, moreover, it also integrates with HEADS firmware. TrustedGRUB2 measures PCR values from 8 to 12, and records in it the following information [38]:

- **PCR 8** First sector of TrustedGRUB2 kernel (diskboot.img)

- **PCR 9** TrustedGRUB2 kernel (core.img)
- **PCR 10** Everything that is loaded from disk (grub2-modules, Linux-kernel, initrd, ntldr, etc.)
- **PCR 11** Contains all commandline arguments from scripts (e.g. grub.cfg) and those entered in the shell
- **PCR 12** LUKS-header

After the Boot Loader the kernel and the applications start, and in the case of Linux kernel, TPM is supported and just needs to be enabled with the proper kernel modules (if it is not enabled by your system by default). Then, now that all the boot sequence is monitored and registered to the TPM, we can also interact with system programs.

To access PCR values stored in the TPM we can use programs like the ones included in the tpm2_tools suite. This set of tools was widely used during tests of this trust measurement system implementation. To read PCR values indeed it is possible to use tpm2_pcrread, and with it, it is possible to read and verify the status of the system, while to extend a PCR hash it is possible to use the tool tpm2_pcrextend and give it as input the index of the PCR value to extend and the new value measured [39].

The main libraries used to develop TPM tools are tpm2-tss, ibmtss2 and wolfTPM. There also is a python library that allows to interface with the TCG TPM2 Software Stack (TSS2). This library called tpm2-pytss is the main one used to develop the proof of concept scripts since those are written in Python [40] [41].

TPM also allows to perform remote attestation, and some frameworks provide this functionality in an easy to use way. One of these frameworks is Keylime [9], and will be the main one used in this project.

3.1.2 Intel SGX Attestation

Intel SGX (Software Guard Extensions) is a set of instruction codes for Intel's CPUs that allow user-level code to allocate private regions of memory (or enclaves) that are protected from running processes to higher privilege levels. Intel designed SGX to be useful especially for secure remote computing, by ensuring the security of the remote system [22].

To see if the Intel CPU supports SGX it is possible to read the CPUID 'Structured Extended feature Leaf' EBX bit 02. Nevertheless, to see if SGX is also available for applications, the BIOS support is required together with the enabling of opt-in [42].

It was introduced in 2015 with the sixth generation Intel Core microprocessors based on the Skylake microarchitecture and nowadays also Intel Gemini Lake architecture supports it [22].

Due to further developments an open-source version of the SGX emulator (OpenSGX) was created, this allows developers to work with intrinsic properties of SGX [43].

Since the Intel SGX main purpose is to secure sensitive code and data against malicious processes, host OS and hypervisor (as well as other privileged software), it also allows performing remote attestation to allows to monitor by using a remote provider the trustworthiness of the hardware and the software running in the enclave.

The types of remote attestations provided by Intel SGX are primarily two:

- Intel EPID (Enhanced Privacy ID) attestation

- ECDSA-based attestation (that makes use of Intel SGX DCAP)

With Intel EPID is possible to perform the remote attestation, preserving the privacy of the remote system. Indeed, Intel EPID does not need to know the remote Intel processor that is running the enclave, and this is allowed by the Intel EPID signatures technology. Remote attestation in this case is provided by Intel services [22].

On the other side, DCAP-based attestation allows to build your own attestation service, and this is mainly useful for enterprise, who wants to keep all the network trust information in-house and not delegate third party providers. This also allows running Intel SGX attestation on networks that are not connected to the internet and cannot communicate with Intel servers. Moreover also allows to run attestation in a distributed way, without having a central entity that checks the status of the whole network.

For the purpose of this project we will focus on EPID-based remote attestation, since we want to allow multiple security domains to communicate. EPID-based remote attestation relies on Intel to certify the status of our nodes so even different security domains with different owners can ensure the security of each other by using Intel's service. Figure 3.2 shows the EPID-based remote attestation flow.

Intel SGX is used to verify the verifier node of this trust management system. The verifier node indeed also needs to communicate with other verifiers in other security domains to exchange the domain trust. With Intel SGX verifiers can check each other health status with Intel that attests the trusted status of the systems [22].

To help to use Intel SGX and implement it in an easy way in a remote attestation system, some framework exist. The one that will take in consideration for this project is Graphene-SGX. Graphene-SGX is a libraryOS (or as they use to

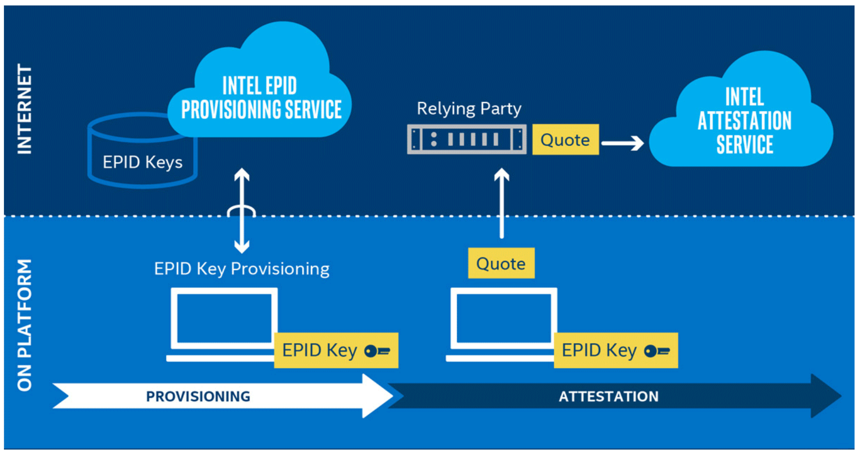

Figure 3.2: EPID-based Intel SGX remote attestation flow [22]

call it an OS library) that allows to integrate Intel SGX in a Linux multi process application [44].

Graphene-SGX is a minimal, fully functional operating system that allows to run applications inside it (at this development stage Graphene-SGX does not support all king of applications and syscall). This means that it works like a virtual machine, and allows full portability and reproducibility of the applications running in it. But those are just some side features of Graphene-SGX, indeed Graphene-SGX allows running unmodified applications within the secure enclaves. By running in a hardware encrypted memory region, it allows to run it in critical system applications, and monitor their status to make sure that those remain untampered and in a trustworthy environment.

Is important to notice that Graphene-SGX, by now is in an early development stage, so its functionalities will be used just to have a future proof, proof of concept. But until its development will not be completed should not be used in a production environment.

3.2 Trust Measurement Implementation Details

The main purpose of this Trust Evaluation System is to verify the trustworthiness of the devices in a network. As specified in the previous chapters this implementation in mainly for servers and routing devices and not for IoT devices. The reason why IoT devices are is covered is that this project requires the use of TPM 2.0.

TPM 2.0 offers good capabilities to measure the status of the system but the main cons is that due to its cost is rarely used in this type of devices. Servers instead, often offer TPM 2.0 integration by default, and in case it is not available it is possible to integrate it. TPM 2.0 can also be emulated, but this functionality suggested only for testing and development purposes.

During the development of this project, emulated TPM 2.0 was mainly used due to it's versatility and the possibility to install it in a virtual machine and create a virtual network of virtual machines running emulated TPM 2.0. This allowed to test and verify the results of different typologies of networks.

This Trust Evaluation System indeed also allows to measure the trustworthiness of a network with different security domains, where each of those is managed and monitored by a verifier (also called remote provider). Each remote provider has the capability to collect in a secure way the information coming from the devices of its security domain. After having collected all the information and calculated the trust values, the verifier can update the tenant that can decide how to manage the network based on the net trust.

It is important that each of the devices in a security domain is covered by this measurement system, since one undetected compromised device can compromise the whole network. More important, is that the verifier is trusted too. The verifier indeed in in charge of both collect trust information about its security domain but also to communicate the security domain status to neighbors security domains with which it needs to communicate.

The availability of the verifier is also important, since the whole trust system depends on it, but this dependent on the network infrastructure and it is out of the scope of this research.

The result of the verifier being compromised can, potentially, compromise the whole network. This is why all the critical processes running on the verifier should be in an enclave. A main point of this research indeed is the coexistence of TPM 2.0 and Intel SGX technologies. TPM 2.0 indeed, will be in charge of network devices monitoring, while Intel SGX scope will be to protect the verifier by making running the critical processes in a secure enclave and by monitoring its status.

Using these technologies together allows to have a fully secure security domain, where the actors are primarily based on the ones of the Keylime framework. Indeed each security domain will be composed by a registrar, a verifier, a tenant and multiple agents. The agent in this case are all the systems of the security domain that need to be verified. The structure of a security domain network and how multiple security domain networks interact will be explained more in depth in Section 3.2.1.

Once a secure and tamper proof network of devices is established we can start transmitting information about systems' status. Information are collected from various sources. We can split those sources in three categories: user defined, system status and system performances. These information refer all to characteristics of the devices that defines a trustworthy level.

All these characteristics will be defined in Section 3.2.2. In that section indeed there will be a complete explanation of what are the system features collected to

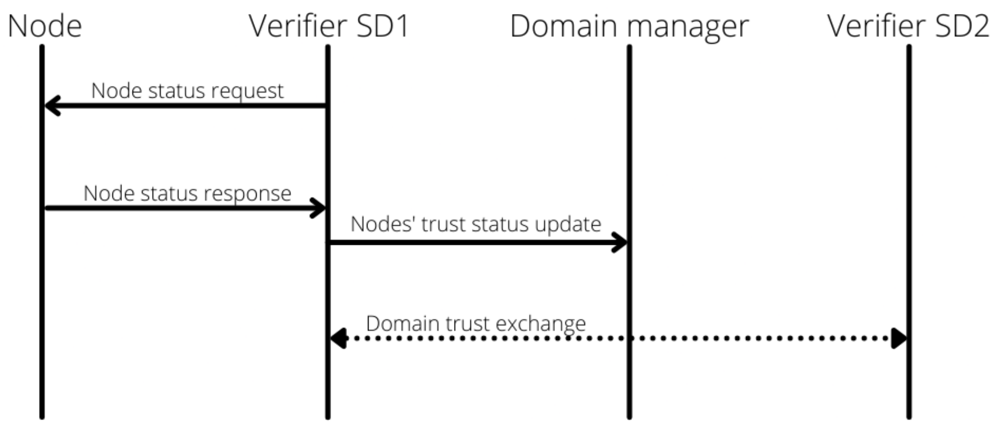

Figure 3.3: Basic Trust Management System communication flow

determine the overall trustworthiness of each system and why those characteristics where chosen.

After having collected all the trustworthiness characteristics, those will be sent to a central verifier, that, by using the data stored in the registrar, checks the level of trust and how to proceed with the network. The verifier once received the trustworthiness data from a node, based on a weighted average algorithm that will be explained in Section 3.3, will calculate a device trust value. Each device in the network should have a trust value assigned by the verifier.

In Figure 3.3, we can see the basic information flow of this Trust Management System. Everything starts from the verifier that probes the nodes for their status, than the nodes respond with the collected evidence. At this point the verifier calculate the new trust values based on the new evidences and update the domain manager, that decide how to act on the network. In the meantime the verifier keep exchanging the overall security domain trust with other verifiers.

This Trust Measurement System can be applied to network management systems like a dynamic route manager, to make the network more secure. Another possible application is to increase or decrease the level of hardening of some devices based on the trust of the neighbors [45]. In the case of low trust neighbors indeed a network node can harden its access policy and traffic control. This will end up in a slow down of the device due to a stricter access policy and packet check, but will considerably decrease the possibility to be compromised. With the possibility to decrease the hardening level, we can still have an increase on the device performances in case the trust level of the neighbors decrease.

All these decisions will be totally automated. By using this technology, we will have a continuously secured and monitored network without falling in an over hardened situation, that can increase the overall security that lead decrease in performance, or a over performing network that lead to an increased probability of compromised systems.

3.2.1 Network infrastructure

Nowadays we have a constantly growing Internet as a service (IaaS) popularity, the problem is that until now there are no solution to offer a trust monitoring system. The solution proposed in this paper aim to fill this gap by enhancing existing devices monitor solutions with a new trust evaluation technique.

The main technology used by the project to monitor the network devices and ensure that the transmitted information are valid and un-tampered is Keylime. Keylime is a scalable trusted cloud key management system based on TPM. The version of keylime used for this project was based on master branch of its official GitHub repository. To integrate it better with this project the source with new functionalities [9].

Keylime is compatible with TPM 1.2 and 2.0, but in this research we will focus mainly on TPM 2.0. Keylime supports scalable network infrastructures allowing to easily extend a network when needed without compromising this Trust Management System. One of the Keylime characteristics is that is also compatible with virtual environment running emulated TPM and this facilitated the test of this project.

The network infrastructure will be heavily in inspired to the one that Keylime proposes. The main actors of the network will be:

- The Tenant
- The Verifier
- The Registrar
- The Agents

The tenant corresponds on the owner of the services, so the one that seed to make sure of the correct operation of the network, and of the trust of all the devices. The tenants, by using Keylime, can manage all the operational network devices (agents) by adding those to the trusted network, and start the trust evaluation process on the active nodes. To ensure a correct status of the nodes the tenant can upload a white-list of the nodes allowed status to the verifier, ant this will allows to detect whether a not is not anymore in a trusted status. The verifier with Keylime has the ability to sent to the nodes sensible data, by encrypting it by using TPM generated symmetric key. This allows the information to reach the node in a totally secure way, even ensuring that the cloud computing service used (in case external services like Google Cloud or Amazon AWS are used) cannot intercept any of the transmitted data.

The role of the Verifier is mainly to continuously monitoring the nodes' status. Indeed each time there is a status change in one of the nodes, the verifier receives the update and decide on how to proceed. To know if the status change of the node is an allowed status, the verifier compare the status data received by the node with the white-list provided by the tenant. In this project the role of the verifier was enhanced. In the Keylime implementation, if the verifier detect that a node change its status to an untrusted one, it removed the nodes from the network. In the current implementation, the verifier, check the new status, elaborates the status information, and, calculates a new trust value based on the new status, and reintegrate the node after having applied the necessary changes (like devices' hardening) to the network. In case the verifier detect that after a status change, the node has a trust value too low, the node is not reintegrated in the network and the tenant is notified about the problem.

The registrar instead is the responsible for the nodes data storing. Indeed when a node is added to the network, the registrar receive the node's public key generated by the TPM. The main role of the registrar is to store the cryptographic information, to allows to securely send and receive information by the nodes. Is important to notice that the registrar does not store any secrets, and its function is mainly to store and give the node's IDs with the associated public key and the certificates. Validity of keys stored in the registrar is easy to verify using TPM attestation.

Last but not least element of this trust management system are at agents. The agents (prior also called simply nodes) are the element of the system the which trust needs to be validated. In those agents there are all the daemons running which function is to measure the system status. These daemons work primarily by interacting with the TPM's PCR registers. The way these daemons work and how they interact with the TPM will be explained in Chapter 3.2.2. Each time the system status changes, the PCR registers are updated and almost in real-time the agent send an update to the verifier. It is important that the window between when the system status change and when the verifier receives the change notification is as short as possible. Indeed if a node is compromised, the verifier or the tenant should act as fast as possible to ensure the safety of the node's data and of the network. A representation of the basic infrastructure needed for this framework can be seen on Figure 3.4.

While the registrar and the verifier should stay in the tenant network, the agent can also stay in the cloud, and with this Trust Measurement System, the cloud environment can be trusted by the tenant. The infrastructure explained till now, is the one that create a single security domain. Since security domains should be insulated, each security domain should have its agents, verifier, and registrars. With this Trust measurement system is possible to exchange information among different security domains. To do that, an exposed service of the verifier is responsible to communicate with other security domains, the status of the network. To verify the status of the verifier, so security domains can trust each other, it should run inside Intel SGX. By using Intel SGX, a verifier can ensure the integrity of the application that is running, since they are running in a secure enclave, and can notify its authenticity to other security domains using SGX remote attestation as explained in Chapter 3.1.2.

3.2.2 Evidence for trust calculation

To evaluate the trust of the nodes it is important to collect some evidence. In the previous chapter we explained the main elements of this Trust Measurement System network infrastructure, this chapter will explain how and what evidences are collected to establish the trust of a network node.

Existing solution for trust evaluation are divided in two main category, perfor-

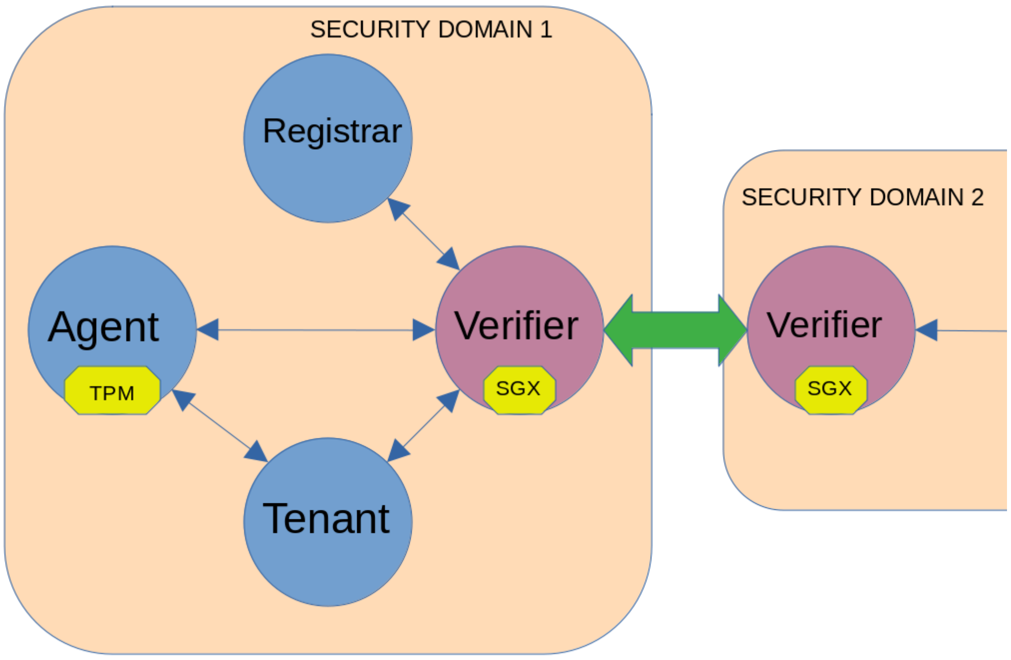

Figure 3.4: Basic framework infrastructure

mance based and system information bases. This research aim to find an hybrid solution that put together both performance and system information. This subdivision happen mainly because of how the concept of trustworthy system is intended. For some application indeed, a trustworthy system is a system that for example has a low latency response time of low computational times. In this case we can talk about a performances based trustworthy concept. In the second case, a trustworthy system is a system with an high level of security.

In our case, the evidence that we gather to calculate the trust of a system can be subdivided in three main categories:

- User Defined
- Performance Based
- Security Based

User defined are values defined by the tenant. Each node should have a user defined trust value defined when the tenant integrate the new node in the security domain. Those are static values and will be used to fine tune the trust calculation algorithm. User defined values are introduced to overcome the problem that each system is different and do different jobs. For example, if we compare two systems with the same trust technical characteristics, but with different functions and importance inside the network, those two systems should have a different trust score. This is the main reason why the user defined trust value is introduced. This value is also useful to calculate the initial trust value of a node whether there are still sot measured evidence.

The second type of trustworthy evidence is the performance based one. Performance based evidence will change the final trust value and will be measured based on the average throughput and the packet loss ratio. The average throughput defines how many packets that node delivered in a certain period. The average packet loss ratio instead refers to how many packets that node lost compared to the generated ones in a certain period of time. Each node has a performance table of the neighbors with those two values and, based on those measurements, a node can tweak the trust value received by the verifier to choose the best path where to instradate the packets also based on the neighbors performances. Each new node as soon as it is introduced in the network should send some test messages to its neighbors to get the initial throughput and packet loss ratio values.

Security based evidence, is the most important of the three, whether performance and user based are calculated for fine tune adjustments, security based evidence is the main one to calculate the final trust score. This type of evidence is also the only one that needs to be transmitted. Indeed after the evidence is collected in the node, that needs to be transmitted to the verifier. Instead, user based evidence resides directly in the verifier and performance based resides in the nodes. The main technology used securely to collect security information about the system is the TPM 2.0 with Keylime. As explained in the previous chapters, the Keylime, by using the TPM can measure in real-time the status of the system and send it to the verifier in a secure way. Since system status is a broad concept. It is important to define what characteristics of the system status we want to measure. In this project we focused mainly on:

- The Boot Sequence Status
- The Installed Applications CVE
- The Unauthorized Files Executed

Boot sequence status is measured by TPM during system boot and registered in the PCR registers. The installed software CVE verification will be executed in the verifier. The node, always by using the TPM will measure the installed applications version, and will store those information inside other PCR registers. Finally, unauthorized files execution will be measured by using the Linux Integrity Measurement Architecture (IMA) [46]. With IMA it is possible constantly monitor the new executable files that are launched in the system. After measuring that a new executable is launched it stores the name of the file and the file HASH in a kernel resident list. This list integrity and security is protected by TPM, that stores it status in one of its PCR registers.

This summarize, how the system is monitored and how to ensure the safety of the values. But, as described in the previous paragraph, as soon as something change in the system, those values need to be send to the verifier to monitor the system status and calculate the new trust value. To transmit those value in a secure way and ensuring their validity and integrity Keylime is used. By using Keylime the verifier can have a real-time overview of each device. The only thing left is to read the PCR values that the verifier received from the nodes. For the Boot Sequence Status, the verifier has a white-list of the allowed boot status. This white-list is composed by allowed boot status hash valued, that later will be compared to the ones received by the node. To measure the installed application CVE, the verifier has a list of application hashes to compare with the received PCR register values. So, from the application hash it tries to find its version. Once the application name and version is found, those information goes through a CVE finding library, that will return to the verifier all the CVE that that particular application version has. More the CVE scores are high and more the final trust value will be low. The last measurement is about unauthorized files execution. Each time this event happen the system will go in a low trust state. The way trust score are calculated will be explained in Section 3.3.

3.3 Trust value calculation method

In the previous chapter we talked about how to analyze the systems and how to collect the evidence for the trust score. Another important part of this research is the trust calculation method. In the case of this trust measurement system there are two types of trust score: node based trust score and security domain based trust score. Both scores are in a range from 0 to 10.

In this first part we will focus on node based trust score. Analyzing what information is collected and how to retrieve a trust score from it.

Collected data form the nodes consists of:

- Installed software with the relative version
- System performance data
- Boot status provided by attestation
- Files status provided by attestation

With all this data is possible to retrieve a system based trust score. To calculate the final trust score, all these evidences will be first taken into consideration individually.

Starting with installed software and versions, as discussed in the previous chapters, to calculate the trust we will consider the CVE score of the vulnerability present in that specific version of software. The trust score relative to installed software will be called T_s. The n^{th} CVE score of the software S is called C_{ns}. So to calculate T_s the following formula is used:

$$T_s = \min_{x \in C_{ns}} \left(\frac{52 - 3,9x - 1,3\overline{C_{ns}}}{4} \right)$$

Then, we also need to calculate the performance trust score, that we can define as T_p. As already discussed this value is calculated based on good and bad performances behaviors of a specific node. In this implementation the evidence is collected based on the packet loss and the traffic throughput. These values are measured in specific time windows. The throughput calculated in a specific time window n is defined as P_n, the packet loss instead is defined as N_n. To calculate T_p, the following formula is used:

$$T_p = \left(\frac{10P_n}{P_n + N_n} \right)$$

The last two pieces of evidence relative to attestation are considered hard trustworthiness evidence. This means that whether one of these two evidences report anomalies in a system, that system is immediately considered not trustworthy and its overall trust T will be equal to 1 and the system will immediately alert the administrator.

The overall node trust score T is measured as a weighted average between T_s and T_p:

$$T = \frac{\omega T_s + (10 - \omega)\, T_p}{10}$$

Where $0 < \omega < 10$ is a value decided by the tenant and indicates how much we want security based trust overcome performance based trust.

The last type of trust score measured by this trust management system is the security domain overall trust score DT. This trust score is purely security based and is based on the average of security based trust scores of the security domain's nodes:

$$DT = \frac{\sum_{n=1}^{k} T_{sn}}{k}$$

Where as defined before T_s is the security based trust score of a node and n indicate a specific node of the security domain where k is the number of nodes in it.

4 Proposed solution evaluation

This chapter will explain how the solution was evaluated. The evaluation of the project is subdivided in two main parts:

- Evaluation of the solution performances
- Evaluation of this solution features by comparing them with the ones of other existing trust evaluation frameworks

The first evaluation is done by using some use cases. The use cases are used to verify the feasibility of the project in real case scenario and to check if it is better then a random trust evaluation. To conduct this type of evaluation there will be first an explanation of the methodology used. Here the setup used to conduct the experiments will be illustrated. The setup description will include the architecture setup, the used software and the values used for the trust calculation.

After this first introduction part there will be the actual use cases description. The use case which have been taken into consideration are two to demonstrate that this framework is not valid just in a particular condition. This part will include the use case description, the custom environment setup based on the use case and the outcome of the experiment.

The use cases taken in consideration are two:

- A security policy management system
- A dynamic packet route manager

The first use case consists in a centralized system that manages security policies on nodes. Those security policies could be things like limits for invalid login attempts, limit the access of a services from a specific IP address or from a network, etc.

The second use case consists in a system that manages the route of the packet based on how sensible and important the information contained in a packet are. For example in the case of an high confidentiality packet, the most secure route will be chosen, otherwise the fastest one.

After the use case description part, there will some final thought about the experiments results. The experiments will be done using different initial setup values, and in this part there will be a comparison of the results based on the used values. There will be also a consideration on what type of initial values are better for what scenario.

Finally, in the last part of this chapter, there will be an evaluation based on the features of this solution. These features will be compared to the one of other existing trust evaluation frameworks. To extract those features, an accurate analysis of the other framework taken in consideration is done. From these frameworks we will extract the features based on what those can do and on what are the things why a framework is different from another.

Those last considerations are important in this research since many competitors exists. This last section indeed allow to see in what this solution differs from the other present in the current literature. This will also give us an overview of what is the current state of the art and also what are the functionalities that lacks the most in the existing solutions.

4.1 Evaluation Methodology

Evaluation is an important part to understand whether this solution is valid and worth to apply and continue improving and if there are some advantages compared to existing solutions.

As explained in the previous section the evaluation will be divided in two main parts. The first is the application of the solution to two different use cases. The choice of experimenting with more than one use case is to demonstrate that it is possible to implement this solution in more environments and it is not created to solve trust problems in just a specific situation. In these use case there will be a measurement of the solution performance, in situation where the network is on dangerous states, such as vulnerable applications running on a system. The values measured are the response time in case of an altered state detection. The response time indeed should be as shorter as possible, indeed during that time the whole network is more vulnerable and could be compromised.

The performance are measured in a network composed of minimum nine nodes, subdivided in two security domains. Since this solution aims not just to manage trust on a single network but on many security domains, the tests will measure both the response time of a security domain when a node in it changes its trust and also when another security domain changes its overall trust.

The measured response time of the security domain in which one on the nodes change its trust will be composed of:

- Detection time of the agent running on the affected node of the altered state
- Detection by the verifier of an altered state on the monitored nodes and new trust measurement
- Detection by the verifier of the new security domain trust to be communicated to other security domains

The time measured for the trust change of another security domain instead will be composed of the aforementioned points and of the time for the security domain to receive the update of the overall trust value from the affected security domain.

The measured will be multiple and will consider a variation of network size. The purpose is to analyze if and how much the network size will affect on the performances of this trust measurement system.

Another important factor to keep in consideration is the change of nodes' trust value to see if it makes sense compared to the nodes' status.

Other than on the performances there will be an analysis of the differences between this solution and other solutions. This last part is important because will allow us to understand in what our solution differs from existing ones. In that section we can reason what are the best use cases for this particular type of solution. There will be also a reasoning on what use case are not optimal to adopt this solution and why.

The last comparison will be interesting also because it will be possible to see what is the current station of existing systems to measure the trust of a network of devices. By reading that section the reader will have an overview of the pro an cons of existing solutions and will help to decide what is the best one based on its use case.

4.2 Use Cases for trust measurement

To test the proposed solution two use cases were chosen. In this chapter there will be a description of how the tests were conducted and a description of the whole infrastructure and tools used. The results of these tests will be analyzed on the next chapter.

The first use case, that later will be explained more in detail, is regarding a policy management system. The purpose of a policy management system is to check and apply security predefined policies in a network of devices. With this trust management system the policies will be dynamic, based on the trust status of the devices that are part of the network. With this trust management system will also be possible to change dynamically the policies regarding neighbor security domains.

In the second use case, this trust management system will be applied to a packet routes manager. In this particular system the packet transmitted in a network will be routed based on the level of confidentiality of the content, to different routes. With this trust management system, will be possible to route the packet of the network dynamically, based on the trust of the nodes or of the security domains.

The network infrastructure used in the two use cases is the same and can be observed in Figure 4.1. It is composed by two security domains, in this case each security domain has its own subnet. In the case of the first security domain (SD1), the subnet used is 192.168.0.0/16. Instead, the subnet of the second security domain (SD2) is 172.0.0.0/16.

All the main network operation will happen in SD1. SD2 will be used to analyze the behavior of this trust management system when overall trust of a neighbor security domain change.

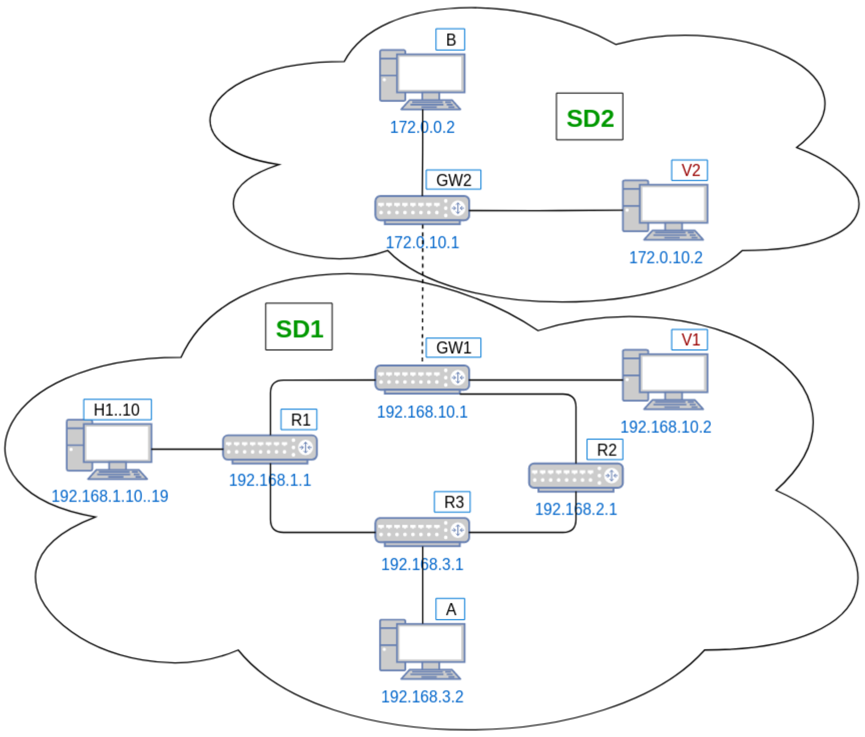

Figure 4.1: Use cases' network schema

In this simulations, the two security domains will be composed by one security domain gateway (GW1 for SD1 and GW2 for SD2), a device that is used to instradate the traffic to a different security domain. Then SD1 will have other three routers R1, R2 and R3. These routers will allow us to implement different routing policies and see how a packet is instradate based on the trust of the nodes in the network. Each router have its own subnet, where other hosts such as servers will reside. The devices connected to the router are A that is connected to R3 and H1 to H10, ten hosts that are dynamically connected to R1.

In the second security domain instead there will be a single host B, that is the one with which we will test the security communication among different security domains.

The last host present in each security domain is the verifier (V1 for SD1 and

V2 for SD2). This is the most important node for these two use cases, since its the main device in charge to mange the trust of a single security domain and to monitor the overall trust of neighbor security domains.

Each node of this simulation has a running agent on it, and the purpose is to either measuring trustworthiness evidence or to manage and calculate the network trust. The first type of agent is installed on all non-verifier nodes. Another essential part of this network is that each node should have a TPM [8]. Indeed, as explained in the previous chapters, to collect the trustworthiness evidence we need a root of trust, that ensure the authenticity of this evidence and that the device is in a trustworthy state. The data collected by the agent for this simulation is the security status of the installed software and kernel, by searching for the vulnerabilities present in the installed version of the software. The vulnerabilities risk is measured with their CVE score [20]. The CVE score of he vulnerabilities present in the installed software is used to measure the host trust score. Another important factor that we ill use to measure the host trust score is the machine status. The machine status is retrieved from the TPM measurements. If the system change its status in an untrusted one, the host trust score will be significantly lowered. Since this trust measurement system take in consideration the system performances as node trustworthiness, the agent will also measure the performances of neighbor nodes, based on the throughput and the packet loss rate. All this evidence will then be sent to the verifier that is responsible of measuring the node trust.

The type of agent installed in the verifier is different compared to the one installed on the other nodes. This agent is indeed responsible to collect the evidence from the nodes of the security domain and to assign a trust value to each of those. The agent in the verifier is responsible also to calculate the overall security domain trust score and to communicate it to neighbor security domains.

It is important for the security domain verifier to have a TPM as well, since with it, it will be able to communicate with other nodes in the network by using asymmetric key encryption. The TPM allows a pretty secure asymmetric key encryption, since the private key is hard stored in the TPM chip and cannot be stolen.

The last task of the verifier is the one to tell to the system responsible to manage the network what the trust values are. This system indeed by using the provided values will be able to dynamically manage the network. For this study we used two different systems to apply this solution. In the following sub-chapters those systems and their interaction with this trust management system will be explained more in detail.

4.2.1 Policy management system

The first use case used to test this trust management system is a policy management system. The purpose of a policy management system is to manage the security policies of all the systems in a network.

To implement a policy management system usually static policies are used. Those policies are decided after a study on the systems that populate a network and on the way they interact. Metrics to decide how much to enforce a system are based on factors like:

- The type of the system (the OS, if it is a server, a firewall, a container or a networking node like a router or a switch)
- Where the system is located in the network (if for example in a DMZ)
- What are the services running in the system

- The importance of the data managed by the system

The problem is that sometimes some of the applied policies cause performance issues or can cause problems on the system accessibility. On the other side they could be too permissive. We need to keep in mind that a network of devices is mutable, for example some systems of the network can be more susceptible to security issues in the case a new vulnerability is disclosed about a service running on that system. Or, due to technical problem one host can have availability problems.

All those changes in the classical implementation of a policy management system are not taken in consideration. With the integration of the trust management system developed in this thesis it is possible to solve this lack. Indeed with this integration, the policies applied to the system can change, based on the variation of a node in the same security domain, on the variation on the node itself, or on the variation of another security domain.

To test our policy management system we simulated a basic policy manager, that manages the ssh access policies of a node. For easiness of implementation, we deployed the central node that is responsible of managing the network policies in the same node as the trust verifier. To be more specific the ssh related security policies that we will change will be: the number of failed login to lock out an user and an IP blacklist for ssh access.

The number of failed login to block the ssh access to an account will change every time a node in the same security domain changes its trust. In the case of an higher trust score, this value will be higher and vice versa.

Regarding the IP black list, the blacklisted IP could be a specific one or a range of IP. This policy is applied every time the trust score of a node decrease under a certain threshold decided decided by the admin. During the test we used a trust threshold of 3 out of 10. When a node trust decrease under the threshold the policy management system set the IP of that particular machine in the blacklists of all the nodes of the same security domain. Instead, if a neighbor security domain overall trust decreases under a certain threshold, the security domain where we are operating will automatically blacklist all the IPs of the affected security domain to prevent possible attacks.

4.2.2 Dynamic packet routes manager

In the second use case the trust management system was implemented on a packet route manager. A packet route manager aim to change the preferred route for the packets sent from a system to comply with the confidentiality level of the information contained in the packet.

This system indeed privileges the fastest route for packets with low confidentiality and the most secure route for high confidentiality communications. Also in this type of systems, static rules are chosen during the deploy of the infrastructure. This can cause security or performances issues in case some nodes of the network change.

With this trust management system, these routes can change dynamically based on the status of the nodes in the network. Since this trust management system collect also information about the performances of the nodes in the network, it will be easy to decide what are the fastest or the most secure route. In this case the way performance base trust and security based trust score interact is fundamental. Indeed, in case low confidentiality packets, performance based trust score will have an higher weight than the security based one, and this is valid also the other way around.

Also in this case, for convenience, the route manager will be placed in the same nodes as the verifier. All the routes will be decided in a centralized manner in the route manager and all the nodes of the security domain will be changed accordingly.

4.3 Experiments outcome

To measure the performances of this solution, the execution times have been measured. To be more specific, the measured performances are:

- the detection time of the agent running on the affected node of the altered state
- the detection by the verifier of an altered state on the monitored node and the calculation of the new trust score
- the calculation of the overall security domain trust to be shared with other security domains

Each one of these three values has been measured with different network size. Indeed, the number of node in the network changed to understand if and how much the reaction time changes based on the number of nodes in a single security domain.

All the extra nodes were connected to R1, and their IP addresses range went from 192.168.1.10 to 192.168.1.10 (H1 to H10). The experiment started with a basic infrastructure without those extra nodes, then measurement were carried out with 2, 4, 6, 8 and 10 extra nodes.

All system modifications to change the trust score have been carried out on node A in SD1. All the trust changed on the system were measured to see if they are consistent with the system statuses. This part is important to understand if the trust value calculation is correct and efficient.

Table 4.1: Change of the trust score of node A over 3 events

Time	**Trust score**	**Change Description**
t0	6.0	Initial node trust
t1	6.9	Security trust after a system software update (new max CVE score:4.9 new average CVE score:4.2)
t2	4.1	Security trust after the installation of Joomla 3.8.12 (new max CVE score: 7.5, new average CVE score:4.9)
t3	6.7	Security trust after the update of Joomla to version 3.9.10 (new max CVE score:5.0, new average CVE score:4.3)

Table 4.2: Execution times in seconds on different network sizes

Number of nodes $\rightarrow$	**9**	**11**	**13**	**15**	**17**	**19**
Node-side detection time	1.4	0.6	1.1	1.4	1.4	0.7
Verifier-side detection and trust calculation time	1.7	1.8	1.7	2.2	3.7	2.8
Domain trust calculation and advertisement time	2.3	2.4	2.4	2.7	4.4	3.5

In Table 4.1 we can see how the node A changes over the time and what security trust scores are applied.

From the results showed in table one, we can see how the trust score changes consistently with the changes applied in the node. Indeed we can see how a vulnerable version of the installed software changes the trust score to a low value, that the system will then consider as less trusted.

In Table 4.2 instead we can observe how the average detection and elaboration time of the system change based on the number of nodes in the network. All the times in the table are relative to $t0$, the time when the change happened in the node.

From that table we can also see how the response time in the node is not depen-

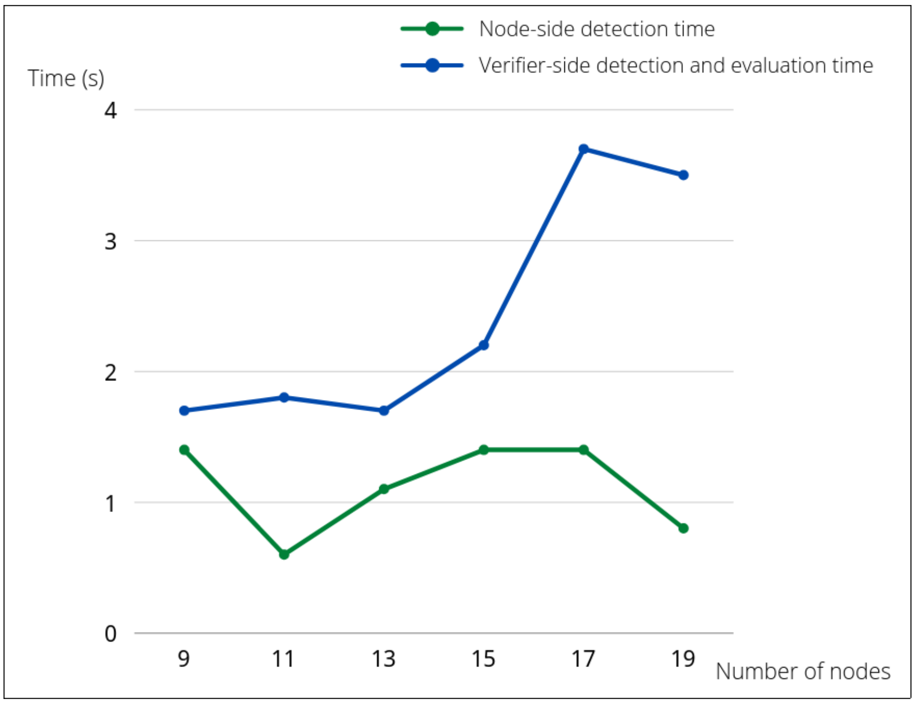

Figure 4.2: Detection times graph

dent on the network size. Whether, the detection and elaboration time verifier-side is dependent on the number of nodes. In Figure 4.2 we can see more in detail how the verifier detection and elaboration times change based on the network size.

Nevertheless all the times measured are in the order of a few seconds, always less than 1.4s on node-side, and can be considered good response times to allow to the system administrator to prevent an attack.

4.4 Considerations and parallelism with existing solutions

In this last part of the evaluation, this research will focus on compare the trust management system developed in this thesis with other existing solutions. The

Table 4.3: Evaluation Features List

ID	Feature Name	Feature Description
1	Device Trust	The framework evaluates the trust of single nodes in a domain
2	Domain Trust	The framework evaluates the trust of the whole domain to exchange it with other domains
3	Security Based	The calculation of the trust score is based on security elements
4	Performance Based	The calculation of the trust score is based on performance elements
5	Use of Attestation	The framework make use of some form of attestation to verify the status of the nodes/domains and to verify the retrieved information
6	Centralized	The framework relies to a central node to manage the trust of the nodes/domain
7	Limitations	What are the main limitations of this framework

main purpose of this comparison it to understand what are the key differences of this solution compared to the others and will be useful to decide whether for a specific application, this solution is the best choice or not.

To compare the analyzed solution for trust management we selected some key features found in those systems, than using a table we will compare those solutions with the extracted features.

To retrieve existing solutions, we analyzed some examples that can be found in the scientific literature regarding similar trust management systems.

In Table 4.3 is possible to see the list of the chosen feature with the relative description.

Table 4.4: Comparison with existing frameworks

Feature ID →	**1**	**2**	**3**	**4**	**5**	**6**	**7**
This trust management system	**X**	**X**	**X**	**X**	**X**	**X**	Requires hardware root of trust
Tao Sun and Mieso K. Denko [15]	**X**		**X**	**X**			
Dimitrios Schinianakis et al. [17]	**X**	**X**	**X**			**X**	5G specific framework
Tim Boland et al. [47]	**X**		**X**				Check only software's trust
Yefeng Ruan et al. [16]	**X**			**X**			IoT specific framework
Ben Niu et al. [48]		**X**	**X**	**X**		**X**	5G specific framework
Esubalew Alemneh et al. [49]	**X**	**X**	**X**	**X**			
Xiao Chen [50]	**X**		**X**				Requires high performance devices

In Table 4.4 instead, we can see in the vertical column the list of the analyzed system and in the horizontal one, the list of the features, represented by a Feature ID that can be found in Table 4.3. For each system we marked with an '**X**' the space relative to the feature, if the feature applies.

From this table we can see how the developed system is the only generic trust management system that works with attestation and takes in consideration both security and performances of the systems. Moreover this system does not work just in a single security domain but take in consideration also other security domain that implement the same trust management system independently.

From the analyzed frameworks we can also observe a lack of the use of attestation technologies. Moreover, most of them focus either on node trust or domain trust and just two on both.

5 Conclusion

Today's organizations rely on internal or cloud-infrastructures to manage their data and their products. Due to the increasing importance and complexity of these infrastructures, there is the need to implement a reliable way to monitor the trustworthiness of the devices that are part of it. In this thesis, we analyzed how a new approach to trust management for network devices works. In the first part, there was a focus on the analysis of similar projects to understand what features existing trust measurement systems have and how they implemented and evaluated them. Then, based on what we have learned we implemented this new trust management system including features that others lack. The developed method allows measuring the trustworthiness of the nodes that compose a network using an agent that collects security and performance-based evidence. These agents send all the collected evidence to a central verifier that then calculates the trust scores of the nodes.

This method also implements a method to evaluate the overall domain trust. This allows exchanging this trust with other security domains without impacting the confidentiality of the network. In a cloud infrastructure, indeed, it is important to have an overview of the status of the nodes and other security domains. Cloud infrastructures indeed can be business-critical for a company, and ensure that each node or security domain is trustworthy is important to avoid potential security risks that can affect crucial parts of the network.

To ensure that this method works consistently and can provide good response times, it was evaluated using two use cases. The first use case is a policy management system and the second a dynamic packet routes manager. The trust measurement method developed in this thesis was implemented on both of these use cases, to analyze its response times and its adaptability to different types of scenarios. During the evaluation phase, we confirmed our assumptions about the efficiency of this method to collect and evaluate the data collected from the nodes. The only problem found was a decrease in the system performances based on the network size. Nevertheless, the maximum response time we got was under 5 seconds with the 19 network nodes, and this can still be considered acceptable. A possible limitation of the proposed solution, based on the tests done, could be a response latency too high in case of a higher node number than the tested one. A possible solution could be to use more verifier nodes under a load-balancer.

Furthermore, during the evaluation phase, this trust evaluation method was compared with similar ones (Section 4.4). This type of evaluation was done by choosing the main features from each of the analyzed methods and comparing them. There, it is possible to observe how in other existing trust evaluation methods, remote attestation is almost never used. Although, we think that in a trust measurement system attestation is very important to be as sure as possible that the data collected is trustworthy. Another feature of the proposed solution is the possibility to exchange the domain trust among different and independent security domains that implement the same system.

As aforementioned, all the thesis assumptions were successfully implemented, analyzed, and confirmed, with the only possible limitation of performance issues based on the network size. It is possible to conclude that this is a solid methodology to evaluate the trust of a security domain and that if improved from this proof of concept status, this method could be implemented in a production environment.

6 Further works and considerations

During the development of this project, all the basic assumptions were confirmed, but all the tests have been done using virtual machines in a simulated environment. The further step would be to try to implement the proposed solution using bare-metal systems to see whether the performance and results are consistent with the ones achieved in the simulated environment.

Still, it was possible to find some limitations, like an increase in the response time as the number of devices increase. This problem needs further analysis to understand what the bottleneck is. An assumption is that the problem is caused by an overload of the verifier. In this case, a possible solution could be to use multiple verifiers under a load balancer and see if this way it is possible to achieve better results than the ones we got during testing. Further improvements could be to find more types of evidence to calculate the security trust. Indeed in some scenarios CVE based trust may not be enough. The same applies to performance based trust. Nevertheless, for proof of concept developed and for the tested use cases, these types of evidence have proved sufficient.

During this thesis, IoT and 5G specific frameworks were analyzed. Nevertheless, in the case of IoT, the computational requirements of the developed method make it difficult to implement on an IoT network. However, this is only an assumption, as no tests were carried out in IoT networks during the evaluation of the developed method. A suggestion for an IoT approach is to run only the agents in the IoT nodes and run all main evidence analysis and trust computation in the central verifier. In the case of 5G infrastructures, there are no performance constraints as in IoT, but there could be compatibility problems between the network infrastructure proposed in this thesis and the 5G one, therefore further tests are needed.

References

[1] D. Nabil, D. Tandjaoui, R. Imed, and F. Medjek, "Trust management in internet of things", 2018.

[2] M. Firdhous, O. Ghazali, and S. Hassan, "Trust management in cloud computing: A critical review", 2012.

[3] C. Scuro, P. F. Sciammarella, F. Lamonaca, R. S. Olivito, and D. L. Carni, "Iot for structural health monitoring", 2018.

[4] S. Zhang, "An overview of network slicing for 5g", 2019.

[5] X. Li, M. Sameka, H. A. Chan, and B. Deval, "Network slicing for 5g: Challenges and opportunities", 2017.

[6] D. Saha and A. Mukherjee, "Pervasive computing: A paradigm for the 21st century", 2003.

[7] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things", 2013.

[8] "Trusted platform module", *Trusted Computing Group*, [Online]. Available: `http://www.trustedcomputinggroup.org/developers/` (visited on 03/28/2020).

[9] N. Schear, P. T. Cable, T. M. Moyer, B. Richard, and R. Rudd, "Bootstrapping and maintaining trust in the cloud", 2016.

[10] A. Regenscheid, “Roots of trust”, 2016. [Online]. Available: https://csrc.nist.gov/Projects/Hardware-Roots-of-Trust (visited on 03/28/2020).

[11] NIST, “Security domain”, [Online]. Available: https://csrc.nist.gov/glossary/term/security_domain (visited on 03/28/2020).

[12] C. Petrov, “Internet of things statistics 2020 [the rise of iot]”, 2021. [Online]. Available: https://techjury.net/blog/internet-of-things-statistics/ (visited on 03/28/2020).

[13] “Cloud computing statistics: 2020 overview”, 2021. [Online]. Available: https://cloud-standards.org/cloud-computing-statistics/ (visited on 03/28/2020).

[14] C. Craven, “What is 5g network slicing?”, 2020. [Online]. Available: https://www.sdxcentral.com/5g/definitions/5g-network-slicing/ (visited on 03/28/2020).

[15] T. Sun and M. K. Denko, “Performance evaluation of trust management in pervasive computing”, 2008.

[16] Y. Ruan, A. Durresi, and L. Alfantoukh, “Trust management framework for internet of things”, 2016.

[17] D. Schinianakis, R. Trapero, D. S. Michalopoulos, and B. G.-N. Crespo, “Security considerations in 5g networks: A slice-aware trust zone approach”, 2019.

[18] M. Sambi, “The power of multi-domain integration for your network”, [Online]. Available: https://blogs.cisco.com/networking/the-power-of-multi-domain-integration-for-your-network (visited on 03/28/2020).

[19] H. Birkholz, D. Thaler, and M. Richardson, “Remote attestation procedures architecture”, 2021.

[20] J. Wallen, “How does the cve scoring system work?”, 2019. [Online]. Available: https://www.techrepublic.com/article/how-does-the-cve-scoring-system-work/ (visited on 03/28/2020).

[21] S. Huin, “What is a trustworthy device? how to ensure they are trustworthy and why it’s important?”, 2020. [Online]. Available: https://blog.irdeto.com/healthcare/what-is-a-trustworthy-device-how-to-ensure-they-are-trustworthy-and-why-its-important/ (visited on 03/28/2020).

[22] “Intel software guard extensions (intel sgx)”, *Intel*, [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html (visited on 03/28/2020).

[23] “Strengthen enclave trust with sgx attestation”, *Intel*, [Online]. Available: https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions/attestation-services.html (visited on 03/28/2020).

[24] “Remote attestation”, *gts3*, [Online]. Available: https://gts3.org/pages/remote-attestation.html (visited on 03/28/2020).

[25] T. Abera, N. Asokan, L. Davi, F. Koushanfar, and A. Sadeghi, “Things, trouble, trust: On building trust in iot systems”, 2016.

[26] “Tpm 2.0 keys for device identity and attestation”, *Trusted Computing Group*, 2020.

[27] “Trusted platform module 2.0: A brief introduction”, *Trusted Computing Group*, 2016.

[28] A. Suciu and T. Carean, “Benchmarking the true random number generator of tpm chips”, 2010.

[29] “Complete security for pcs and embedded systems”, *Microchip*, [Online]. Available: https://www.microchip.com/en-us/products/security-ics/tpm (visited on 03/28/2020).

[30] "Tpm 1.2 vs 2.0 features", *Dell*, [Online]. Available: https://www.dell.com/support/article/en-us/sln312590/tpm-1-2-vs-2-0-features?lang=en (visited on 03/28/2020).

[31] W. Arthur, D. Challener, and K. Goldman, "Platform configuration registers", 2015.

[32] "Trusted platform module technology overview", *Microsoft*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-overview (visited on 03/28/2020).

[33] N. Cui, "A technical introduction to the use of trusted platform module 2.0 with linux", 2017.

[34] "Extends a pcr.", *tpm2-tools*, [Online]. Available: https://www.mankier.com/1/tpm2_pcrextend (visited on 03/28/2020).

[35] T. Olzak, "Uefi and the tpm: Building a foundation for platform trust", 2012. [Online]. Available: https://resources.infosecinstitute.com/topic/uefi-and-tpm-2/ (visited on 03/28/2020).

[36] "Unified extensible firmware interface (uefi) specification", 2019. [Online]. Available: www.uefi.org.

[37] "Heads", [Online]. Available: https://osresearch.net/ (visited on 03/28/2020).

[38] "Tpm enabled grub2 bootloader", [Online]. Available: https://github.com/Rohde-Schwarz/TrustedGRUB2 (visited on 03/28/2020).

[39] "Trusted platform module (tpm2.0) tools", [Online]. Available: https://github.com/tpm2-software/tpm2-tools (visited on 03/28/2020).

[40] "Oss implementation of the tcg tpm2 software stack (tss2)", [Online]. Available: https://github.com/tpm2-software/tpm2-tss (visited on 03/28/2020).

[41] “Tpm2 tss python bindings for enhanced system api (esys)”, [Online]. Available: https://pypi.org/project/tpm2-pytss/ (visited on 03/28/2020).

[42] “Getting started with intel software guard extensions sdk”, 2017. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/getting-started-with-sgx-sdk-for-windows.html (visited on 03/28/2020).

[43] P. Jain, S. Desai, S. Kim, M. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B. B. Kang, and D. Han, “Opensgx: An open platform for sgx research”, 2016.

[44] C. Tsai, D. E. Porter, and M. Vij, “Graphene-sgx: A practical library os for unmodified applications on sgx”, 2017.

[45] “What is network security policy management?”, *Cisco*, [Online]. Available: https://www.cisco.com/c/en/us/products/security/what-is-network-security-policy-management.html (visited on 03/28/2020).

[46] R. Sailer, X. Zhang, and T. Jaeger, “Design and implementation of a tcg-based integrity measurement architecture”, 2004.

[47] T. Boland, C. Cleraux, and E. Fong, “Toward a preliminary framework for assessing thetrustworthiness of software”, 2010.

[48] B. Niu, W. You, H. Tang, and X. Wang, “5g network slice security trust degree calculation model”, 2017.

[49] E. Alemneh, S. Senouci, P. Brunet, and T. Tegegne, “A two-way trust management system for fog computing”, 2020.

[50] X. Chen, “A decentralized trust management system for intelligent transportation environments”, 2020.