Ronny Bäckman

Simulating Rail Traffic Management with Trusted Computing

Bachelor’s thesis

Bachelor of Engineering

2020

XAMK

South-Eastern Finland
University of Applied Sciences

XAMK
South-Eastern Finland
University of Applied Sciences

Author (authors)	**Degree title**	**Time**
Ronny Bäckman	Bachelor of Engineering	May 2020
Thesis title Simulating rail traffic management with Trusted Computing		69 pages
Commissioned by Nokia Bell Labs		
Supervisor Vesa Kankare		

Abstract

The railway traffic management and security system are undergoing a renewal. Computer based systems are becoming the backbone that control moving trains. At the same time, cyber-attacks are becoming more common in the sphere of industrial control systems. The purpose of this thesis is to show how the trusted computing module can help mitigating attacks on industrial control systems.

The thesis introduces the basic framework of social trust and the options available to expand that trust into the computer domain with the trusted platform module and attestation. Industrial control system attacks such as Stuxnet, Triton and Industroyer are introduced to present vectors on how the railway security system can be targeted in a cyber-physical attack. The thesis also briefly examines the Finnish Railway security system and traffic management.

Study was conducted by means of intervention methodology. A background study was made concerning the implementation of a simulation environment for testing integrity failures in rail traffic. Testing was conducted to find out if integrity measurements are needed in this environment.

The findings show that attacks can generate incidents that can be noticed by monitoring firmware integrity. The study also shows that in a rail security environment where measured boot and attestation have been implemented, integrity deviations are not only easily noticed but also possible to pinpoint.

The simulation framework developed in this study uses containers to simulate devices. Admittedly this is a limited approach when measuring firmware integrity. Containers illustrating the firmware startup and runtime, can adequately showcase complex structures of attestation with multiple devices. The scope of the study we hope to expand into testing on real rail security systems.

Keywords

simulation, trusted platform module, rail traffic management

CONTENTS

1 INTRODUCTION

Railway infrastructure has been secured by a reliable infrastructure based on relays for decades. However, these systems lifecycle is ending. Old systems are replaced by new computer-based systems. (Buurmans et al. 2018.)

The shift to computer-based systems will bring targeted cost savings, but also the inheritance of all security weaknesses. These are new challenges for the rail industry. In order to make the change smoothly while new safety systems are introduced, new personnel, methods and standardization are needed to maintain security. (Buurmans et al. 2018.)

Industrial Control Systems (ICS), have seen a ramp-up in cyber-attacks during the last 10-years. Before this era, cyber security was not a crucial part of the ICS. (Assante & Lee 2015; Caracano et al. 2018; Langner 2013.)

The integrity of the software in ICS is important, but so is the option of remote programming (Stumpp 2019). Restrictions for programming or updating can render a device inoperable.

The challenge is to notice integrity failures in ICS within a time frame when incident prevention is possible (Langner 2013). The failure notification should provide additional information so that measures can be taken faster than with a general notice of failure (Chien et al. 2011).

1.1 Purpose of the thesis

Previous ICS security systems where implemented without a connection to the Information Technology (IT) segment. In the railway industry, the security system consisted of relays. These would provide security without intervention from a computer-based system. Today every new track section is controlled by a computer system responsible for the security. (Kantamaa & Sorsimo 2018.)

In the past, the rail industry would implement the relay-based security system, verify the security implementation and put it into production (Kantamaa & Sorsimo 2018). Integrity failures could be monitored through analog signals. In a computer-based system this is not enough, a new verification method is needed. There is no unified solution for monitoring the computer-based security systems integrity in the rail industry (Stumpp 2019).

The purpose of this study is to incorporate an integrity monitoring solution for the computer-based security ICS in a railway environment. The aim is to simulate the implementation to provide an example for further studies. In a larger scale, the aim is to help the rail industry and others in implementing trusted computing.

This thesis was commissioned by the Cyber Security Research Team at Nokia Bell Labs in Espoo, Finland, as an extension to their prior research into Trusted Computing.

1.2 Research questions and scope

The thesis research questions are:

- How can a firmware integrity failure in the security system lead to a dangerous situation in the rail environment?
- How can a railway traffic operator validate the firmware integrity of the security system?

The first question evaluates the need to monitor the integrity of railway security systems. The question can be answered by exploring attacks made on similar systems and then simulating them on the railway system.

The second question seeks to find a solution to make integrity monitoring accessible for the traffic operator. Active integrity monitoring has not been a requirement for device manufacturers (Stumpp 2019). The question also seeks to accomplish faster mitigation times on integrity failures to prevent incidents. By utilizing the simulation, tests can be conducted to evaluate integrity monitoring by an operator.

The focus of this thesis is on the railway security system implemented in Finland. A simulation scenario is implemented and presented in this study. The Trusted Platform Module specification 2.0 by Trusted Computing Group (TCG) will be used as the integrity monitoring solution. We will follow the Personal Computer (PC) implementation.

1.3 Research method and material

The research method is intervention. This study will provide a method of testing an integrity monitoring solution for devices attached to a railway security system. The purpose is to increase the security of a system. (Kananen 2017.)

Technology for firmware measurements have been accessible for 20 years. This have been used in monitoring the integrity of the PC platform, other computer platforms have not yet adopted the method. We strongly believe that by developing a framework that can demonstrate benefits in different cases, we can accelerate the implementation of this technology. A reason why the simulation framework is developed to be used on a single computer is the flexibility of demonstration.

Implementing a new system into a real railway security system is not applicable without testing. There are very strict standards, on testing new components to a railway security system (Kantamaa & Sorsimo 2018).

However, the rail industry has testing facilities and equipment that could be used for testing. The problem is lack of Trusted Platform Module (TPM) implementations in the existing equipment. This is a requirement for measuring the integrity on the computer-based security system.

Therefore, we developed a simulation framework to test our proposed solution. By using background material on the Finnish railway security implementation, containerization and software TPMs, we were able to build our own test framework for evaluation.

In the theory section, scientific literature is reviewed, the concepts of trust and asymmetric cryptography is explained and the characteristics of cyber attacks are studied. In addition, the concepts of TPM, attestation and virtualization are examined with reference to the railway security system. Based on the gathered information, a simulation environment where built on which tests can be run.

Establishing integrity measurements in computer-based systems follows the Trusted Computing Groups (TCG) specifications. The specification for the PC platform has wide adoption, Microsoft Windows have required Original Equipment Manufacturers (OEM) to include a TPM 2.0 implementation on new computers since 2016 (TPM recommendations 2018).

1.4 Phases of the thesis

Chapters 2–4 introduce the conceptual basis of this study. Chapters 5 and 7 are describing the structure of the simulation framework made for the case study of the rail traffic management system from chapter 6. Chapter 8 analyzes and presents the results of the simulation. Chapter 9 presents a summary of conclusions made in the study.

2 TRUST

The concept of trust in computer systems originates from social trust. The focus of this thesis is on the aspect of containing trust in computer systems. This chapter is an introduction to the general idea of social trust and how it can be implemented into computer systems.

2.1 A philosophical view on trust

Fukuyama (1995) states that trust does not reside in integrated circuits and it is not reducible to information. He defines trust as the expectations one has on others within a community of regular, honest and cooperative people that share common norms. (Fukuyama 1995, 25–26.)

It is difficult to describe or define why we trust someone since trust is a complex concept. When something is considered complex there is always an amount of uncertainty, actions whose results cannot be fully predicted (Nason 2017, 8). If trust only where complicated, it could be separated into steps and processed into results. This is not feasible with regards to social trust. It might be possible to break a trust relationship with a friend into components, but not to fully explain our gut feeling of trust in a person.

Computer systems are originally pieces of components that interact in a logical way with a set of static rules and algorithms. One could ask why we need trust in a system, that perform with a set of predefined rules. It should be possible to calculate the output of a computer, but it would not make much sense using a computer to compute a result that is already known. A various amount of the growth in the world has happened owing to immediately available results generated by computers. A process we do not necessary understand but trust.

Trust would not be needed if actions could be undertaken with complete certainty and no risk (Lewis & Weigert 1985). Implementing computing processes with complete certainty would require a great amount of effort, and every component and piece of code would have to be verified by the end user. Utilizing a system like that would not very easily contribute to growth.

The application of trust has given society significant gains (Harari 2014). The financial revolution which started at the end of the 17th century established a system where states could loan money due to general trust that they would pay back (Roseveare 1991). Since then, the credit systems have been fine-tuned to establish trust in the loaners. However, the general trust occasionally fails which leads to an economic crisis. Today, most often such crisis is reflected all over the world.

Trust in computer systems are in an infant stage compared to the financial system. The first commercial computers were introduced in 1945, after the transistor was invented. Only much later have people started to rely on trust when using computers. In the early years of computers, people verified by other means the results the computer calculated (Ceruzzi 2003).

Human trust complexity is a major part of all computing. All systems that are in use today rely on code and hardware that have been developed by thousands of people. It is not feasible to establish an individual trust relationship with everyone who is developing these systems. Therefore, certificates play a crucial role in establishing trust.

According to Oxfords dictionary in English, a certificate is an official document attesting a fact (Soanes & Stevenson 2005). Digital certificates can be used to sign computer hardware and code. This certificate ties the signed piece to a verified identity. Other attributes can be added to the digital certificate if necessary (Azad & Pathan 2014).

In this thesis, we look at the most common methods to verify the integrity of a computing platform are examined. More trust in the platform can be established if it is possible to verify that every piece of code is the one intended to be used.

2.2 Private messaging and signed verification

This sub-chapter presents an introduction to Public Key Infrastructure (PKI) and hash functions. PKI is one of the main methods of implementing trust on computers. It is used to securely and privately exchange data on the Internet. It is also widely used to sign hash values of code and data.

Public key infrastructure (PKI) use public-key cryptography or asymmetric key cryptography. Cryptography can be referred to as the science of preventing access to sensitive data by parties who are not authorized to access the data. Using asymmetric keys, the encryption key is public (public-key) and the decryption key is private (private-key). Figure 1 shows the basic scheme of encrypting and decrypting information.

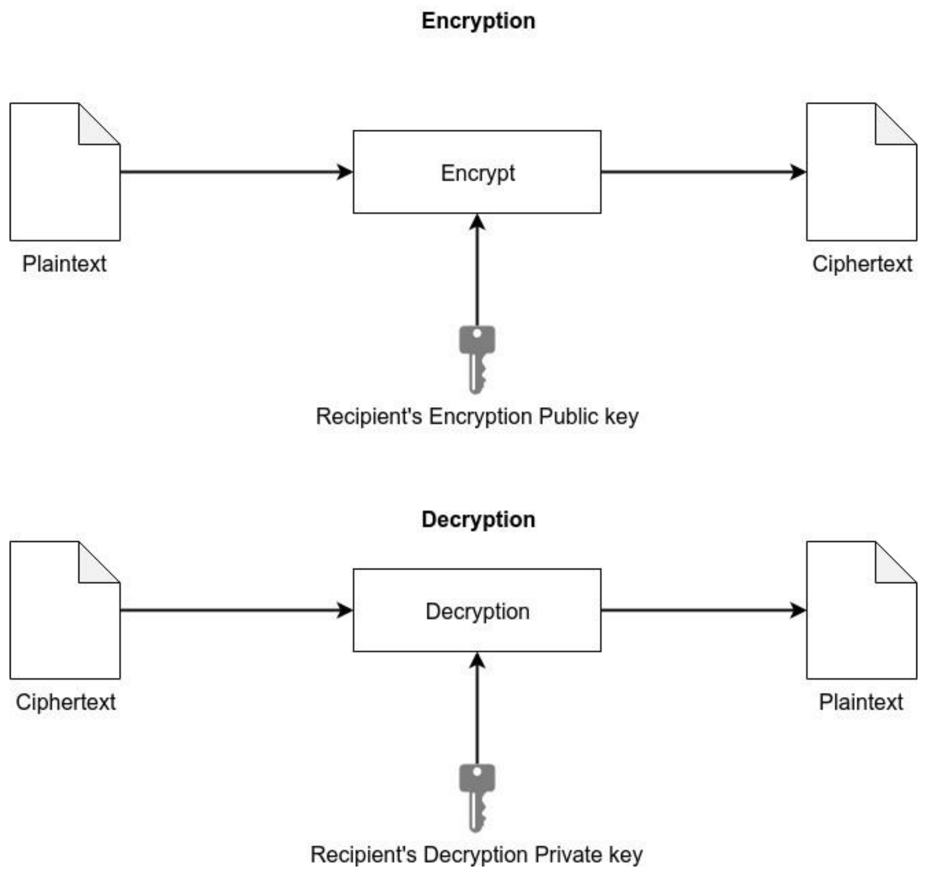


Figure 1. Asymmetric-key crypto scheme (Adams & Lloyd 2002, 13)

PKI extends trust with a Certification Authority (CA) that will verify the requester's identity before issuing a certificate and linking the key to that identity. Parties that trust the CA can rely on the verification of other parties. Certification is the act of binding identity details with a public key (Adams & Lloyd 2002, 85.).

For example, if Bob and Alice want to communicate. Alice registers her asymmetric key with her identity to a CA. Bob can then ask for Alice's certificate for the public key through a CA he trusts. Bob then encrypts his message with Alice's public key and sends the message to Alice. Alice is then able to decrypt the message with her private key. (Das & Madhavan 2009, 2–5.)

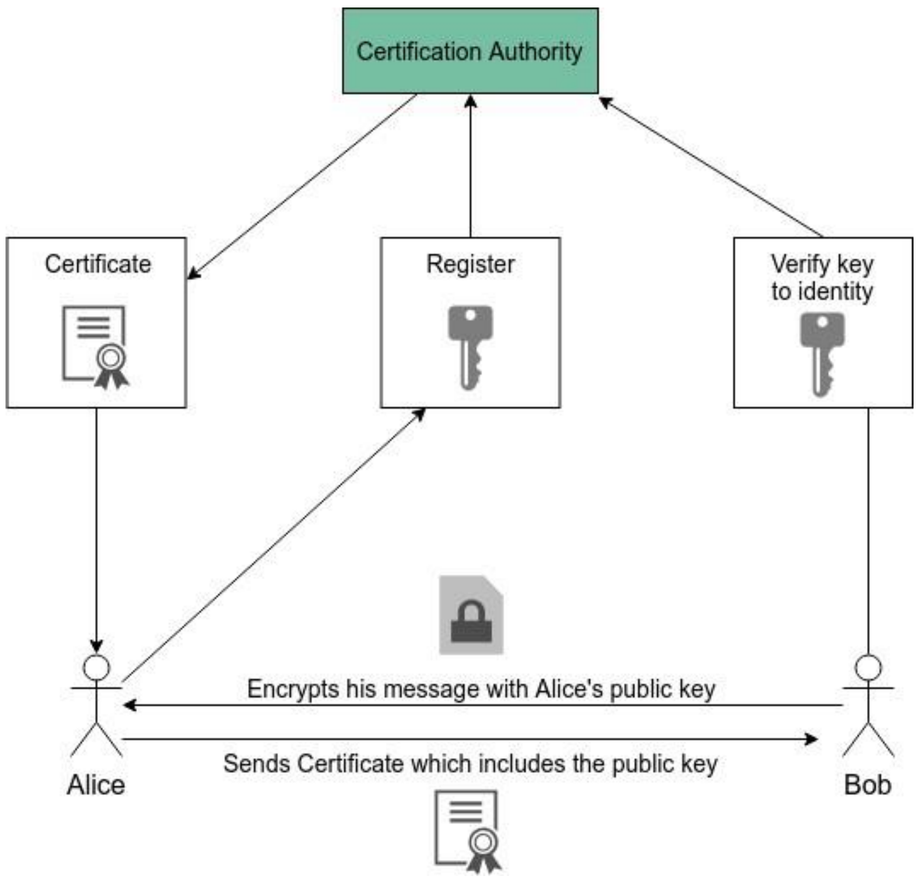

Certification Authority
Certificate
Register
Verify key to identity
Encrypts his message with Alice's public key
Sends Certificate which includes the public key
Alice
Bob

Figure 2. Public Key Infrastructure verification (Vacca 2004).

CAs use trust chaining to extend trust boundaries, hence forming PKI hierarchies. A hierarchy has at least one root certificate. Intermediate certificate's are signed with the root certificates private-key. There can be many levels of intermediate certificates. Verification can be done by going back the chain to the root certificate or through cross-certificate chains that are established between intermediate certificates. (Vacca 2004, 23–24.)

Trust provided by PKI is only as strong as the CA chain. If a certificate's private key is compromised by any means, the PKI hierarchy under that key is compromised. The process of key revocation is cumbersome, and the initial PKI implementation often suffers from problems. Revocation relies on updating information on compromised keys and the information is usually published in Certification Revocation Lists (CRL). (Vacca 2004, 25.)

Integrity values can be signed by a private key that is registered to a CA. The values are most often hashes of the data. A hash is a value returned from a hash function. This function maps data of arbitrary length to a fixed size and the process is infeasible to reverse (Azad & Pathan 2014). This function can be used to verify data integrity. Due to the fixed length, a small storage is sufficient to store hash values.

Together, PKI and hash functions provide a method of establishing trust in computer systems. These methods are widely used but can be difficult to implement securely. The easiest way is to let a third party perform the implementation, thus outsourcing the liability.

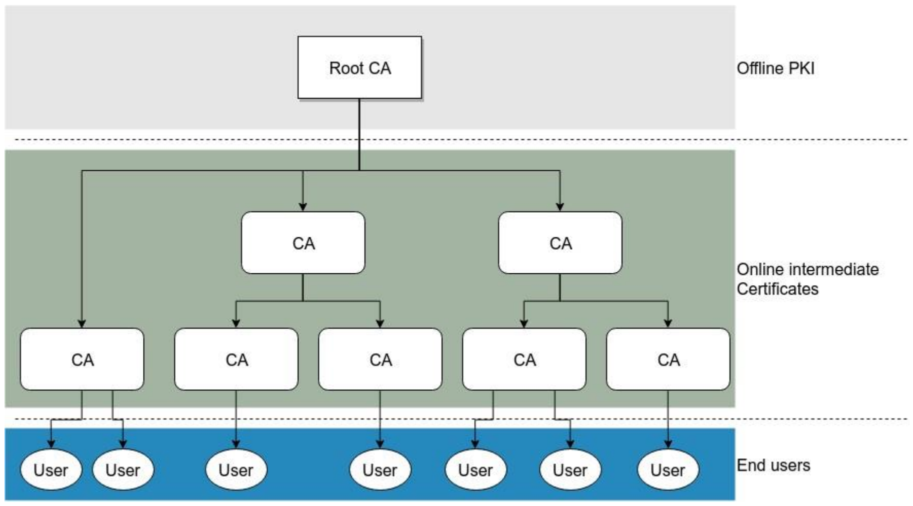


Figure 3. PKI hierarchy (Vacca 2004).

This model makes it possible to sign malware as well, which can happen if access control to the private key fails (Zetter 2019). Because most verification systems work automatically, they will only fail when the signature is invalid. The signed malware is not noticed until something else fails in the computer system.

2.3 State of Railway Security Systems

The European Railway Agency (ERA) launched a study to acquire an overview of the existing Command, Control and Signalling (CCS) systems. This was done to assist ERA with the European Rail Traffic Management Systems (ERTMS) deployment. ERTMS aims at replacing the different national train control and command systems in Europe. (Buurmans et al. 2018, 6.)

The ERA study report was finished in 2018, 10 countries were part of the research. All Railway Infrastructure Managers in this study were at least considering implementing digital-based CCS systems. It is believed that the lack of people with expertise in this new field is affecting the adaptation of digital systems. Insufficient competence affects the regulatory side which struggles to predict the new risks. (Buurmans et al. 2018, 6–8.)

The EN 50126 standard specifies the CCS systems safety requirements. EN 50126 part 5 specifies functional safety in Railway applications (EN 50126-5: 2014). Key specifications are made for the development process (Kantamaa & Sorsimo 2018). However, few specifications are made on software maintenance and patching. Implementations are left to the vendor or contractor that oversees maintenance (Stumpp 2019).

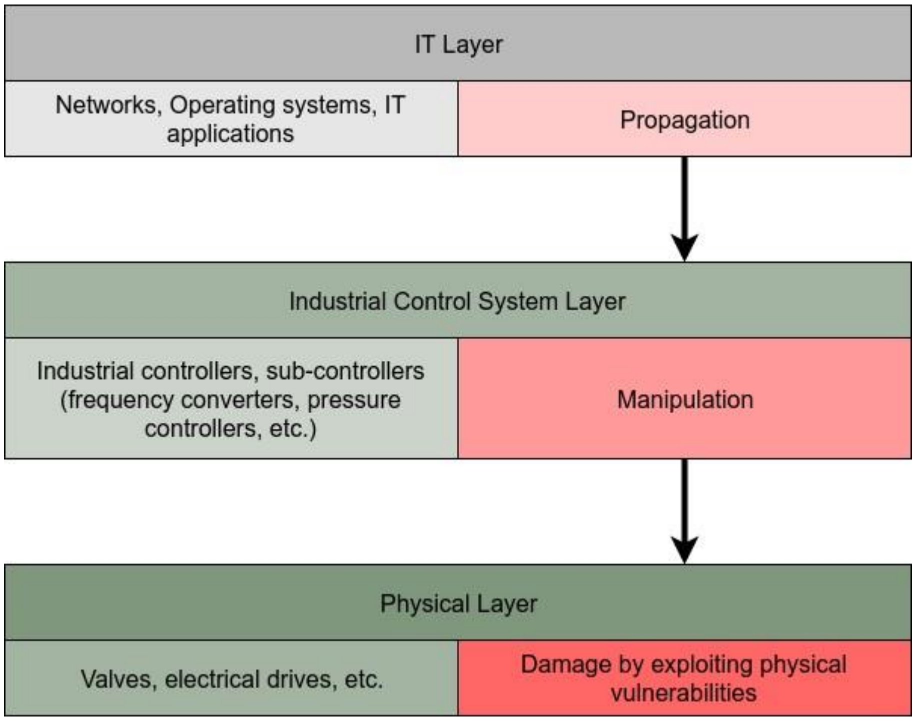
The Railway CCS system was in the past a vendor specific implementation with specific applications, components and interfaces to comply with national specifications (Buurmans et al. 2018). This has made the systems unfavourable among attackers due to the limited affect.

Today, new interoperability specifications in Europe and the demand to lower the cost of the old relay-based CCS systems, new standardized digital CCS systems will start emerging. If no appropriate tools and procedures are implemented, standardized systems can become a target for criminals and hackers in the future. The next chapter introduces previous attacks on systems similar to those that are being introduced into the railway sector.

3 ATTACKING INDUSTRIAL CONTROL SYSTEMS

Attacks against Industrial Control Systems (ICS) can be described as cyber-physical attacks. They involve more layers than the everyday criminal attack on the Internet. In Figure 4, the layers are reproduced from Langner (2013). The IT layer is used to inject and spread the malware. The control system layer is used to manipulate process control to accomplish damage on the physical layer. (Langner 2013, 4.)

In this chapter, three well known attacks against ICS-systems are introduced. The oldest attack, Stuxnet, changed the security environment for these systems. Before Stuxnet, air gaping these systems to secure them seemed sufficient. The amount of resources needed to develop malware for ICS-systems is dramatically lower now (Caracano et al. 2018). Common communication protocols, standards, equipment and computer architectures will even lower the bar for attackers if the security aspect is not taken into greater consideration.


IT Layer
Networks, Operating systems, IT applications
Propagation
Industrial Control System Layer
Industrial controllers, sub-controllers (frequency converters, pressure controllers, etc.)
Manipulation
Physical Layer
Valves, electrical drives, etc.
Damage by exploiting physical vulnerabilities
Figure 4. Layers of a sophisticated cyber-physical attack (Langner 2013).

3.1 Stuxnet

The purpose of the Stuxnet attack was to slow down the enrichment of weapon grade uranium in Iran’s enrichment facility at Nathanz. The attack utilized the Information Technology (IT) layer to manipulate the ICS layer which then caused harm to the physical layer. (Langner 2013, 4.)

Centrifugal rotors were broken as an effect of the attack. Because the rotors were radioactive, they were troublesome to replace. This meant that enrichment time were lost, and the throughput of weapon grade Uranium was decreased.

Two different attacks were carried out between 2007-2010 to accomplish the objective of prolonging development of a nuclear bomb in Iran. It is suggested that the second attack were introduced later because the attackers wanted faster results. This made the second attack bolder and more visible, which eventually lead to the detection of the whole attack in 2010. This was approximately two years later than the beginning of the first attack. (Langner 2013, 4–5.)

The first attack propagated into the IT layer only through physical media that were opened with a certain vendor’s engineering software. Through a laptop, the malware was able to reach the ICS-layer where it infected the main Siemens S7-417 industrial controller which controlled the valves and pressure sensors of 984 centrifuges.

The malware positioned itself between the legitimate operational logic and the analog inputs and outputs (I/O) were it was able to control I/O operations. It passed signals or generated faked signals repeatedly to the legitimate logic. (Langner 2013, 8–9.).

The first phase malware also de-calibrated the pressure sensors that were connected to the Siemens S7-417 controller. This was a measure to hide the operation and not directly control the sensors. The direct control of the sensors could have been discovered by the facility personnel more easily than the recalibration. (Langner 2013.)

Stuxnet’s second IT propagation path was designed with bold tactics and hiding was not prioritized. It utilized previously unknown vulnerabilities (zero-days) against Microsoft Windows and was able to enter the system as a legitimate driver using stolen credentials. It was even able to maintain Command and Control functionality to the infected air-gaped ICS system by peer-to-peer communication through the IT layer. (Chien et al. 2011, 21–23.)

Siemens S7-315, a smaller Programmable Logic Controller (PLC) than the S7-417, was targeted in the second attack. This controller was in charge of the Centrifuge Drive System. By suspending the control logic and iterating this procedure, the centrifugal rotors were slowed down from 63,000 rpm to 120 rpm and up again to the original speed. (Langner 2013.)

During speed up, there is certain critical phases that causes the rotor to vibrate. Every time the rotor vibrates, there is a chance that it will break (Langner 2013). This eventually had a greater impact than the previous attack.

Due to the aggressive IT propagation path of the second attack, malware samples started to appear early 2010. Some were discovered as early as 2009 but the slow propagation on the IT layer kept the general interest low. (Langner 2013, 4.)

At least two driver certificates were stolen and used to sign malware that posed itself as legitimate Windows drivers. The stolen certificates were from Realtek Semiconductor and JMicron Technology Corporation (Langner 2013, 20.).

3.2 Industroyer

On 23 December 2015, a power outage occurred in Ukraine that left 225,000 citizens without electricity before Christmas celebration. A year later, on 17 December 2016, Ukraine suffered another power outage. Both power outages were the result of a cyber-attack against the electric power system. (Cherepanov 2017.)

The first attack took offline at least 27 sub-stations across three energy companies. Attackers were successful in compromising the IT system and used the Human Machine Interface (HMI) to turn off the sub-stations. Mitigation against the attack was slowed down by destroying software and interfaces to the ICS system. (Analysis of the Cyber Attack on the Ukrainian Power Grid 2016, 8–9.)

The first attack utilized legitimate control tools in the IT layer to shut down the power. In the second attack, the attackers used tools that were able to control switches and circuit breakers directly with their own tools. Four different standardized communication protocols were used:

- IEC 60870-5-101, protocol for monitoring and controlling electronic power systems on serial connections.
- IEC 60870-5-104, extension of 101 to TCP/IP networks.
- IEC 61850, protocol for multi-vendor communication of electrical substation automation systems.
- OPC DA, software standard for real-time data exchange between distributed components. (Cherepanov 2017, 2.)

The first two protocols are part of a set of standards which define systems used for telecontrol in electrical engineering and power system automation applications (IEC 60870-5: 2020). The former utilizes serial communication interfaces and the latter the internet protocol suite (TCP/IP).

IEC 61850 is an international standard for communication networks and systems for power utility automation (IEC 61850: 2020). Its aim is to accomplish multivendor communication between devices on electrical substation automation systems. A device task can be protection, automation, metering, monitoring or control. (Cherepanov 2017, 10–12.)

The last protocol the malware used was Microsoft's OLE for Process Control (OPC) which applied several Microsoft technologies for Remote Procedure Calls (RPC) to enable real-time data exchanges between distributed components. (Cherepanov 2017.)

The malware Industroyer were structured in different payloads for these protocols. They were controlled by a launcher that on a trigger date executed the payloads. It is not confirmed but strongly believed that this malware were the one causing the second power outage in Ukraine. A hard-coded trigger date was detected inside the malware that matched the second power outage (Cherepanov 2017,
15.).

3.3 Triton

During the last months of 2017, Mandiant, a subsidiary of FireEye, responded to a cyber incident in the Middle East. The attackers were targeting a Safety Instrumented System (SIS) which provided emergency shutdown capabilities to an industrial process. By examination of the attack, FireEye assessed that the attackers were developing a capability to cause physical damage at the facility (Johnson et al. 2017).

A SIS is the last safety system in an industrial facility. Its task is to safely shutdown the facility in a threatening situation to prevent damage to equipment and personnel. It runs independently of other systems, monitoring thresholds and activating on its own if values are out of range. (Johnson et al. 2017.)

The attacker compromised the IT layer and gained control to an engineering workstation. From the workstation, the attacker reprogrammed the SIS controllers. This triggered some controllers to enter a failsafe mode which was noticed by the personnel at the plant (Greenberg 2017).

The code that injected the payload to the targeted Triconex Safety Instrumented System from Schneider Electric used a zero-day vulnerability. The injected malware would have provided remote control capabilities of the SIS device to the attackers. (Johnson et al. 2017.)

It is not known which code triggered the failsafe, the injector or the payload. However, one of the processors inside the multi-core device triggered a redundancy alarm which forced all three main processors to start the safety shutdown process. (Caracano et al. 2018, 17.)

Security researchers at Nozomi Networks restructured the process of developing and testing malware at the targeted system. Their results indicate that attacking ICS is no longer outside the reach of criminals. When Stuxnet was developed, it required specialised skills to attack ICS. This is not the case anymore because tools, techniques and equipment are accessible to anyone wishing to start developing their own attacks against these systems. (Caracano et al. 2018, 19.)

4 TRUSTED PLATFORM MODULE

The former Trusted Computing Platform Alliance (TCPA) established in 1999 served as the foundation upon which the Trusted Computing Group (TCG) was formed in 2003. Since the beginning, it has worked on providing an efficient and low-cost way to implement trusted computing. A concrete part of the outcome is the Trusted Platform Module (TPM) which has been manufactured and implemented into many platforms. (Arthur & Challener 2015.)

The TPM can establish a Trusted Platform (TP) that is able to store early firmware measurements from the platform. These measurements can then be evaluated against known good values which can reflect on human trust.

TCG has specifications of the TPM implementation for different platforms, the x86 PC platform specification was utilized in this study with the latest specification version 2.0, which is an architectural redesign from the 1.2 version (Arthur & Challener 2015, 5).

This chapter starts by introducing the generic TPM provided features. Then the components of a trusted platform are examined. Finally, TPM identity and attestation is briefly described with reference to the platform boot.

4.1 TPM features

A TPM can be implemented by at least four different means (TCG 2019):

- Discrete TPM
- Integrated TPM
- Firmware TPM
- Software TPM

A discrete TPM is a stand-alone chip on a platform. It provides the highest level of security by also withstanding hardware tampering. An integrated TPM is also implemented in hardware but incorporated in another chip. This exposes the TPM for hardware tampering through exposed interfaces. (TCG 2019.)

Figure 5 shows some of the different TPM implementations and features.

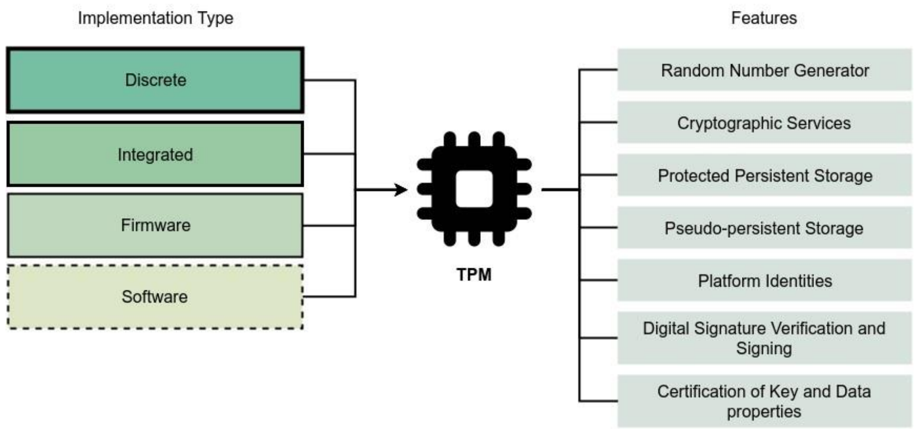

Implementation Type
Discrete
Integrated
Firmware
Software
TPM
Features
Random Number Generator
Cryptographic Services
Protected Persistent Storage
Pseudo-persistent Storage
Platform Identities
Digital Signature Verification and Signing
Certification of Key and Data properties

Figure 5. TPM implementations and features (TCG 2019).

The rest of the TPM implementations are different software implementations. The most common x86 processor manufacturers has implemented the TPM as firmware inside their Central Processing Units (CPU). This is the standard consumer line implementation.

Intel’s firmware TPM implementation is part of the Converged Security and Management Engine (CSME) formerly known as the Management Engine (ME) (Ruan 2014). AMD’s implementation is part of the Platform Secure Processor (PSP) (Cimpanu 2018).

The firmware TPM is protected inside a Trusted Execution Environment (TEE). This is a form of separation from the normal workload running on the platform. However, the TPM needs to rely on other software for its security. (TCG 2019.)

This thesis uses a software TPM developed by Ken Goldman from IBM. His implementation extends on source code donated by Microsoft. The simulator works according to the TCG TPM 2.0 specification (Goldman 2016).

A software TPM is not intended for production use. It is, however, an efficient way to test and develop applications that use TPM features. Here are some of the features the TPM provides (TCG 2019):

- High quality random numbers
- Cryptographic services
- Small protected persistent storage
- Pseudo-persistent store of keys and data
- Platform identities
- Signing and verifying digital signatures
- Certifying the properties of keys and data

The TPM can generate random numbers through a cryptographic function done on random seeds implemented inside a protected storage of the TPM. The seed information is not accessible directly by the user or the platform, only by the TPM. (Trusted Platform Module Library, Part 1: Architecture 2016.)

A seed is implemented as multiple one-time programmable eFuses. Once an eFuse-bit is set to the value of 1, it cannot be set back to 0. A high-quality random value is injected during manufacturing to the seed which gives the TPM a trusted entropy. (TCG PC Client Platform Firmware Profile 2019.)

Three different seeds are implemented on the TPM. These are maped to different life cycle roles with their own authorization. In this thesis only the endorsement seed is used, which maps to the endorsement hierarchy. Authorization values or methods for accessing the hierarchy are omitted from this study. (Trusted Platform Module Library: Part 1: Architecture 2019.)

Through the seeds, the module is also able to generate keys used for cryptographic functions such as encryption, decryption and signing. The TPM provides both symmetric and asymmetric cryptographic services.

The protected persistent storage is small and can only hold a few keys at once. By using the seeds and a Key Derivation Function (KDF), pseudo-persistent keys can be generated. This is achieved by providing an input for the KDF. If the input and the seed are the same, key generation will be consistent. (Trusted Platform Module Library: Part 1: Architecture 2019.)

By restricting the key usage inside the TPM, an asymmetric private key can be part of the platform identity. The public key is available for verification, but the private key never leaves the TPM.

4.2 Trusted Platform

A Trusted Platform is constituted of a Trusted Building Block (TBB). Three components are needed (Trusted Platform Module Library: Part 1: Architecture 2019):

- TPM
- CPU
- Storage for the initial application on the platform.

The initial application is called the Core Root of Trust for Measurements (CRTM) in the TCG specifications. The initial start up script in this study is referred to as the CRTM.

Depending on the TPM type, the aforementioned components are implemented differently. In a discrete TPM solution, they should all be integrated inside the TPM chip. An example of an integrated solution is presented in Figure 6. The connections in Figure 6 are also considered part of the TBB.

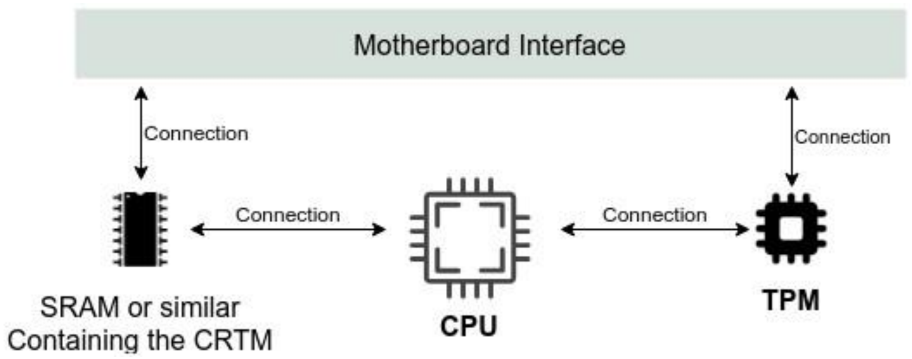


Figure 6. Trusted Building Block (Trusted Platform Module Library: Part 1: Architecture 2019).

The TBB needs to be vouched for by the manufacturer of the platform. The trust in a platform starts with the reputation of the manufacturer and the implementation of the TBB. The end user should be able to expect that the TBB does not compromise the goals of a trusted platforms. (Trusted Platform Module Library: Part 1: Architecture 2019, 21-22.)

From the elements in a TBB, three Roots of Trusts can be derived. These are TCG specified system elements whose misbehaviour cannot be detected by the user. Therefore, these elements need to be trusted in order to establish a Trusted Platform. The three Roots of Trust are (Trusted Platform Module Library: Part 1: Architecture 2019):

- Root of Trust for Storage (RTS)
- Root of Trust for Measurement (RTM)
- Root of Trust for Reporting (RTR)

The RTS is contained inside the TPM. Interactions are only made through the TCG specified interfaces. The storage is constituted mostly by shielded space but also the accessible Platform Configuration Register (PCR). The PCR plays a crucial role in establishing a trusted platform and is described later in this chapter.

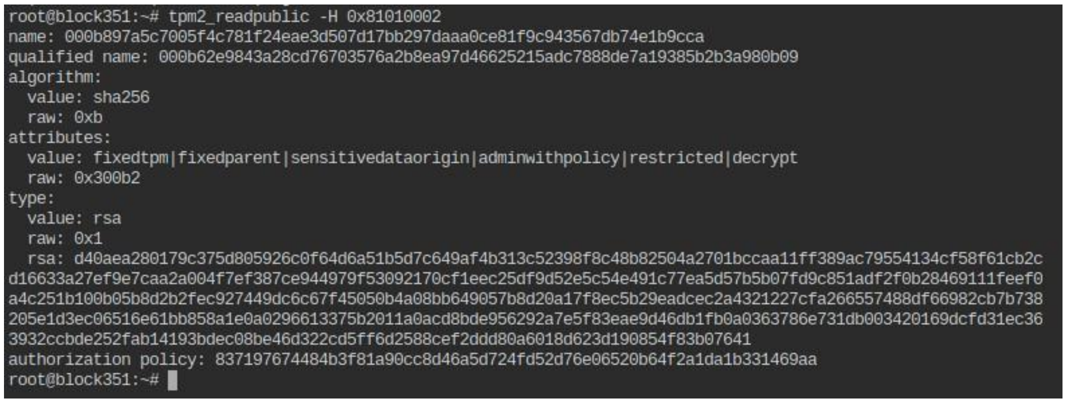
The RTM consists of a CPU and the first set of instructions (CRTM) that are executed when a system is reset. The CRTM will measure itself and the next instruction set that will be run after. The measurements will be stored in the RTS. Reporting is done on the contents of the RTS, which is the function of RTR (Trusted Platform Module Library: Part 1: Architecture 2019). The report is usually a signed digest of the values inside the RTR. The whole TBB is needed to accomplish reporting. In addition, a platform identity is needed.

4.3 Platform Identity

The TPM contains cryptographically verifiable identities for reporting. The feature to produce asymmetric keys inside the TPM provides asymmetric aliases for the platform. An alias private key should only be usable inside the TPM. (Trusted Platform Module Library: Part 1: Architecture 2019).

An Endorsement Key (EK) is produced when the TPM is manufactured. A certificate is signed by the manufacturer for the asymmetric decryption key. The certificate can verify that the TPM is produced according to TCG specifications. Usually, the platform EK certificate is stored inside the TPM. This can then be verified by a PKI chain to the manufacturer. An example of the EK certificate generation and verification is shown in Figure 8. (TCG TPM v2.0 Provisioning Guidance 2017.)

Figure 7 shows an example of the public information of an EK generated by the TPM simulator used in this study. The TPM restricts the use of a key through attributes.



```
root@block351:~# tpm2_readpublic -H 0x81010002
name: 000b897a5c7005f4c781f24eae3d507d17bb297daaa0ce81f9c943567db74e1b9cca
qualified name: 000b62e9843a28cd76703576a2b8ea97d46625215adc7888de7a19385b2b3a980b09
algorithm:
  value: sha256
  raw: 0xb
attributes:
  value: fixedtpm|fixedparent|sensitivedataorigin|adminwithpolicy|restricted|decrypt
  raw: 0x300b2
type:
  value: rsa
  raw: 0x1
  rsa: d40aea280179c375d805926c0f64d6a51b5d7c649af4b313c52398f8c48b82504a2701bccaa11ff389ac79554134cf58f61cb2c
d16633a27ef9e7caa2a004f7ef387ce944979f53092170cf1eec25df9d52e5c54e491c77ea5d57b5b07fd9c851adf2f0b28469111feef0
a4c251b100b05b8d2b2fec927449dc6c67f45050b4a08bb649057b8d20a17f8ec5b29eadcec2a4321227cfa266557488df66982cb7b738
205e1d3ec06516e61bb858a1e0a0296613375b2011a0acd8bde956292a7e5f83eae9d46db1fb0a0363786e731db003420169dcfd31ec36
3932ccbde252fab14193bdec08be46d322cd5ff6d2588cef2ddd80a6018d623d190854f83b07641
authorization policy: 837197674484b3f81a90cc8d46a5d724fd52d76e06520b64f2a1da1b331469aa
root@block351:~# 
```

Figure 7. Endorsement Key public information.

Table 1 describes in more detail the function of these attributes for the EK. Keys that are generated from seeds are primary objects that acquire automatically the attributes fixedtpm and fixedparent (Trusted Platform Module Library: Part 1: Architecture 2019, 160). The administrator authorization role is beyond the scope of this thesis.

Table 1. List of Endorsement Key attributes (Trusted Platform Module Library: Part 1: Architecture 2019).

Attribute	Description
fixedtpm	The private key can not be extracted from the TPM.
fixedparent	The seed is not extractable from the TPM.
sensitivedataorigin	Key derived from a seed within the TPM. Restricts usage to protect the seed value.

adminwithpolicy	Admin role authorization method.
restricted	Restricts asymmetric key functionality.
decrypt	Type of functionality the key is restricted to.

For reporting purposes, a signing key is needed. This is usually generated later by the platform owner. The best practice of reporting is performed through remote attestation. This is why the reporting key is called the Attestation Key (AK).

Attestation can be achieved without a TPM. In this study the TPM provided functions and methods to accomplish remote attestation is used. Attestation with a TPM is performed by signing data inside the TPM.

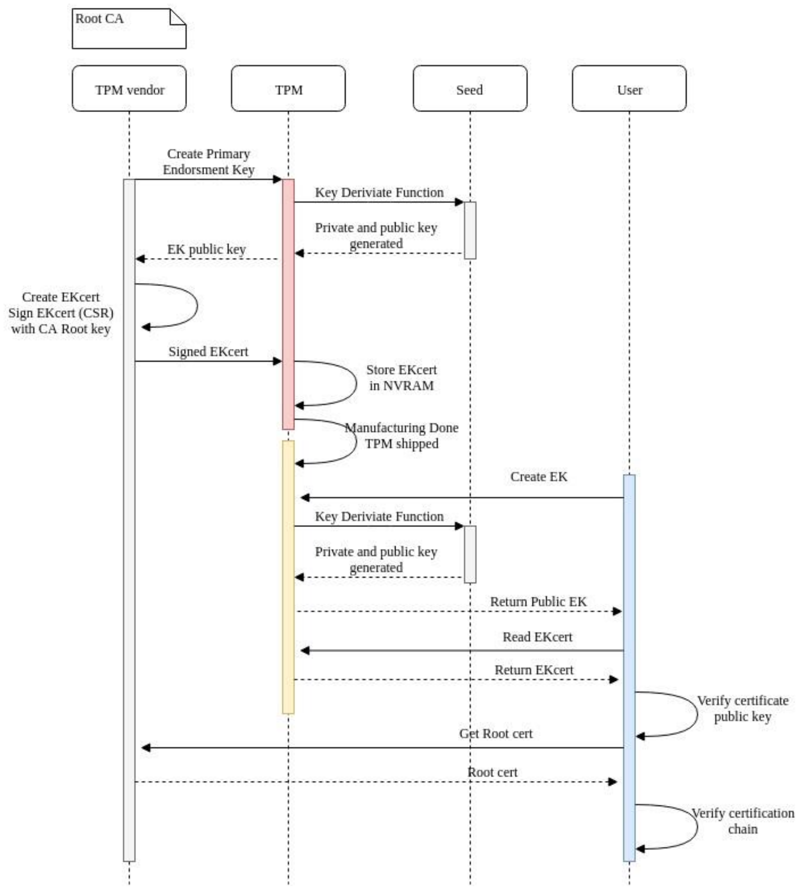

Root CA
TPM vendor
TPM
Seed
User
Create Primary Endorsment Key
Key Deriviate Function
Private and public key generated
EK public key
Create EKcert Sign EKcert (CSR) with CA Root key
Signed EKcert
Store EKcert in NVRAM
Manufacturing Done TPM shipped
Create EK
Key Deriviate Function
Private and public key generated
Return Public EK
Read EKcert
Return EKcert
Verify certificate public key
Get Root cert
Root cert
Verify certification chain

Figure 8. Creation and verification of EK certificate (TCG TPM v2.0 Provisioning Guidance 2017).

Introduced here is a corporate example where the platform is provisioned to a remote attestation service. This is done before it is handed over to the end user. Before provisioning, the owner verifies the trust certificates of the platform. In many corporate cases, the owner is an IT administrator. The number of certificates available depends on the manufacturer and the supply chain process.

If the owner considers the platform trusted, he generates an AK for the platform and provisions the platform to the remote attestation service. In a simple setup, the AK is generated under the EK with the attributes shown in Table 2. The attribute userwithauth is a policy method that is not further discussed in this study.

Table 2. List of requires AK attributes (Trusted Platform Module Library: Part 1: Architecture 2019).

Attribute	Description
fixedtpm	The private key can not be extracted from the TPM.
fixedparent	Parent key (EK) is not extractable from the TPM.
sensitivedataorigin	Parent key derived from a seed. Restricts usage to protect the seed value.
userwithauth	User role authentication can be provided by password, HMAC or policy.
restricted	Key functionality restricted.
sign	Key restricted to signing only.

In this thesis, only a few generic features of the provisioning process are examined and more detailed methods are omitted. The endorsement certificate of the device requesting enrolment is validated by the provisioning service. This proves that the requester uses a certified TPM.

The AK attributes are verified by the provisioning service, and a challenge is generated together with the attestation key name. This is then encrypted with the public part of the requesters EK and sent back. A certified TPM will only decrypt the challenge if the AK is loaded into the requester TPM and the AK name matches the response. (TCG TPM v2.0 Provisioning Guidance 2017.)

It should be noted that this requires a TPM that is developed by TCG standards. The EK certificate is crucial because otherwise there is no guarantee that the TPM or some other entity will be able to accept an AK generated outside of a

TPM. The generic provisioning process is shown in Figure 9. (TCG TPM v2.0 Provisioning Guidance 2017.)

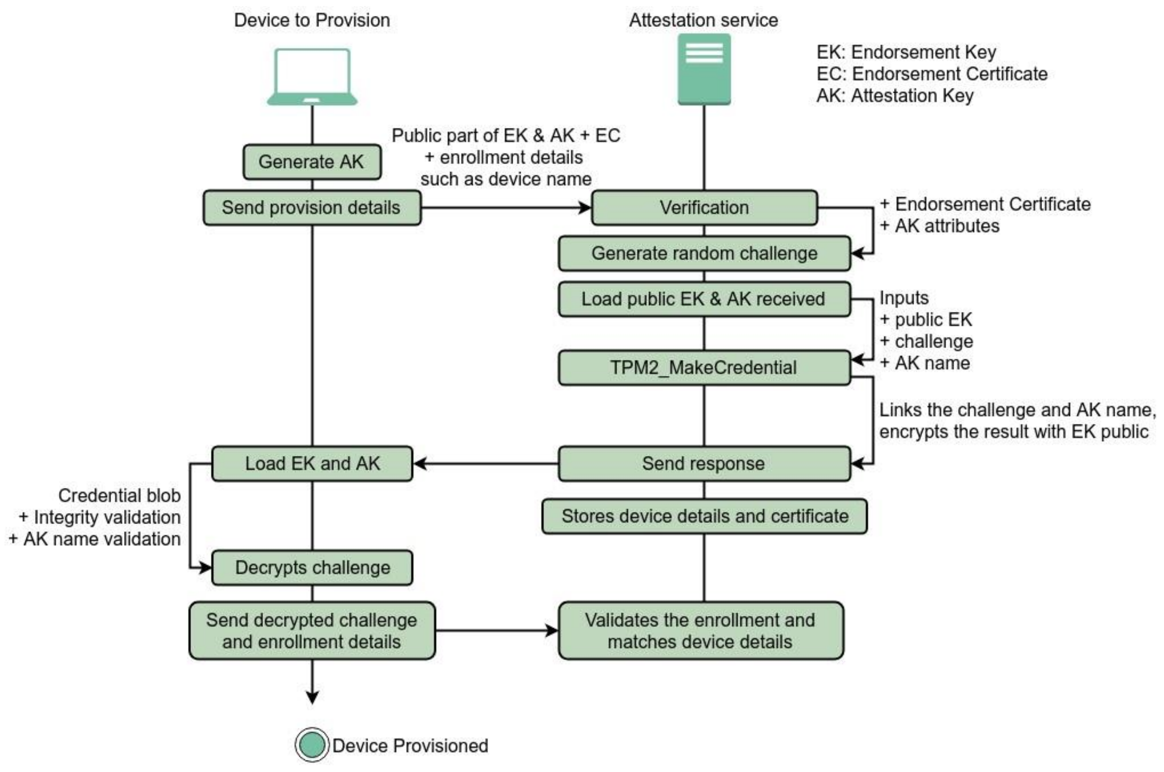

Device to Provision
Attestation service
EK: Endorsement Key
EC: Endorsement Certificate
AK: Attestation Key
Generate AK
Public part of EK & AK + EC + enrollment details such as device name
Send provision details
Verification
+ Endorsement Certificate
+ AK attributes
Generate random challenge
Load public EK & AK received
Inputs
+ public EK
+ challenge
+ AK name
TPM2_MakeCredential
Links the challenge and AK name, encrypts the result with EK public
Load EK and AK
Send response
Credential blob
+ Integrity validation
+ AK name validation
Stores device details and certificate
Decrypts challenge
Send decrypted challenge and enrollment details
Validates the enrollment and matches device details
Device Provisioned

Figure 9. Device provisioning to attestation service (TCG TPM v2.0 Provisioning Guidance 2017).

4.4 Platform boot

A Platform Configuration Register (PCR) is a storage that contains one hash of a supported hashing function by the TPM. Per supported hashing algorithm, there are at least 24 PCRs. SHA-1 and SHA-256 are mandatory in a TPM 2.0 implementation. (TCG PC Client Platform TPM Profile Specification for TPM 2.0. 2020.)

When a platform is powered on or reset, it starts the CRTM instruction set which provides functionalities to measure itself and the next set of instructions. Depending on the TPM implementation, different hardware elements are interacting during the CRTM execution. Before the CRTM completes, it shall extend the measurement hashes into a PCR. (TCG PC Client Platform Firmware Profile 2019.)

An extend function takes the old hash value, adds the new and hashes it. The sequence of extending a PCR needs to be the same throughout resets, because extending A before B would not bring the same results as extending B before A. The extend function is presented below.

$$PCR_{new} := H_{alg}(PCR_{old}||digest)$$

The PCRs is started with a default initial condition upon platform reset. The initial condition for a PC platform is PCRs 1-16 and 23 all bits zero and PCRs 17-20 all bits one. PCR 0 is a special case and acquires an initial value between 4–0 on a PC platform. (TCG PC Client Platform TPM Profile Specification for TPM 2.0. 2020.)

CRTM starts a process of Transitive Trust by measuring itself and the next application before handing over execution. No other application measures itself than the CRTM. As long as the measured values are represented by trusted applications, the software stack can be trusted. (TCG PC Client Platform Firmware Profile 2019.)

An ideal situation would be that all software developers publish trusted values of their products so that measurements could be verified. This is, however, not the case and therefore it is necessary to establish a trusted baseline after installation or configuration. If measurements deviate from the previous software installation or configuration, it can be assumed that unwanted changes have occured.

When deviation along the chain is spotted, no further software measurements can be trusted. This is due to the fact that the previous stage is responsible for measuring the next. It has therefore the ability to tamper with the measurement.

Table 3 illustrates the aspects that are measured into each PCR on a PC platform. The PCRs 0-7 are considered the Static Root of Trust for Measurement (S-RTM) and are only re-producible on a system reset (TCG PC Client Platform Firmware Profile 2019).

Table 3. PCR usage on the PC platform (TCG PC Client Platform Firmware Profile 2019).

PCR Index	Static-Root of Trust for Measurements
0	CRTM, BIOS, Host Platform Extensions, Embedded Option ROMs and PI Drivers
1	Host Platform Configuration
2	UEFI driver and application Code
3	UEFI driver and application Configuration and Data
4	UEFI Boot Manager Code and Boot Attempts
5	Boot Manager Code Configuration and Data and GPT or Partition Table
6	Host Platform Manufacturer Specific
7	Secure Boot Policy
8-15	Defined for use by the Static OS

Figure 10 aims to map different boot specifications to the boot process. It shows that the TCG specified Measured Boot covers the whole boot process. Verified Boot and UEFI Secure boot usually function together and can verify the firmware components utilizing PKI. These methods usually halt the boot process if verification fails. (Bratus et al. 2019, 338–339.)

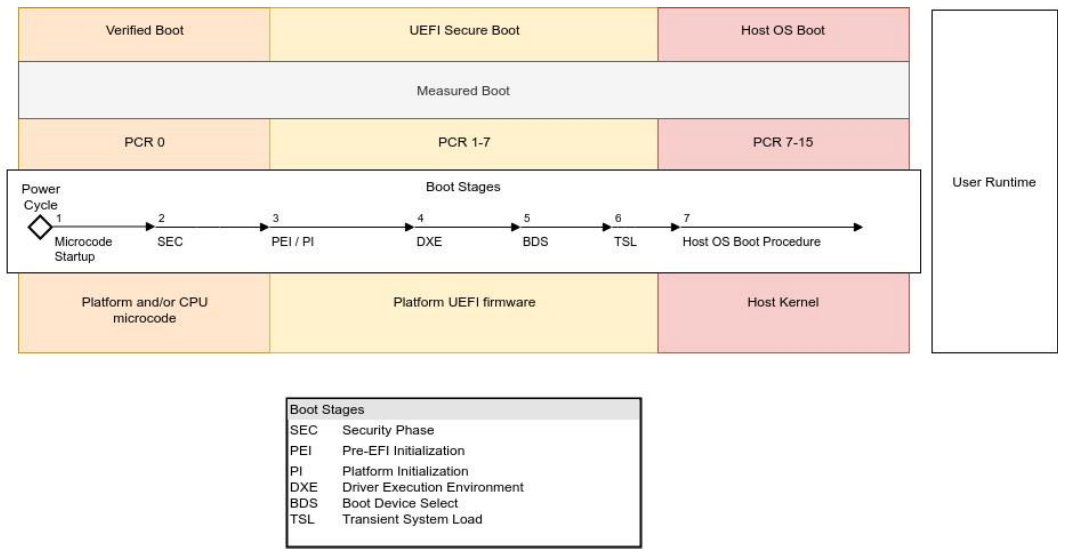


Figure 10. Different boot types protecting the software stack (Bratus et al. 2019).

4.5 Attestation

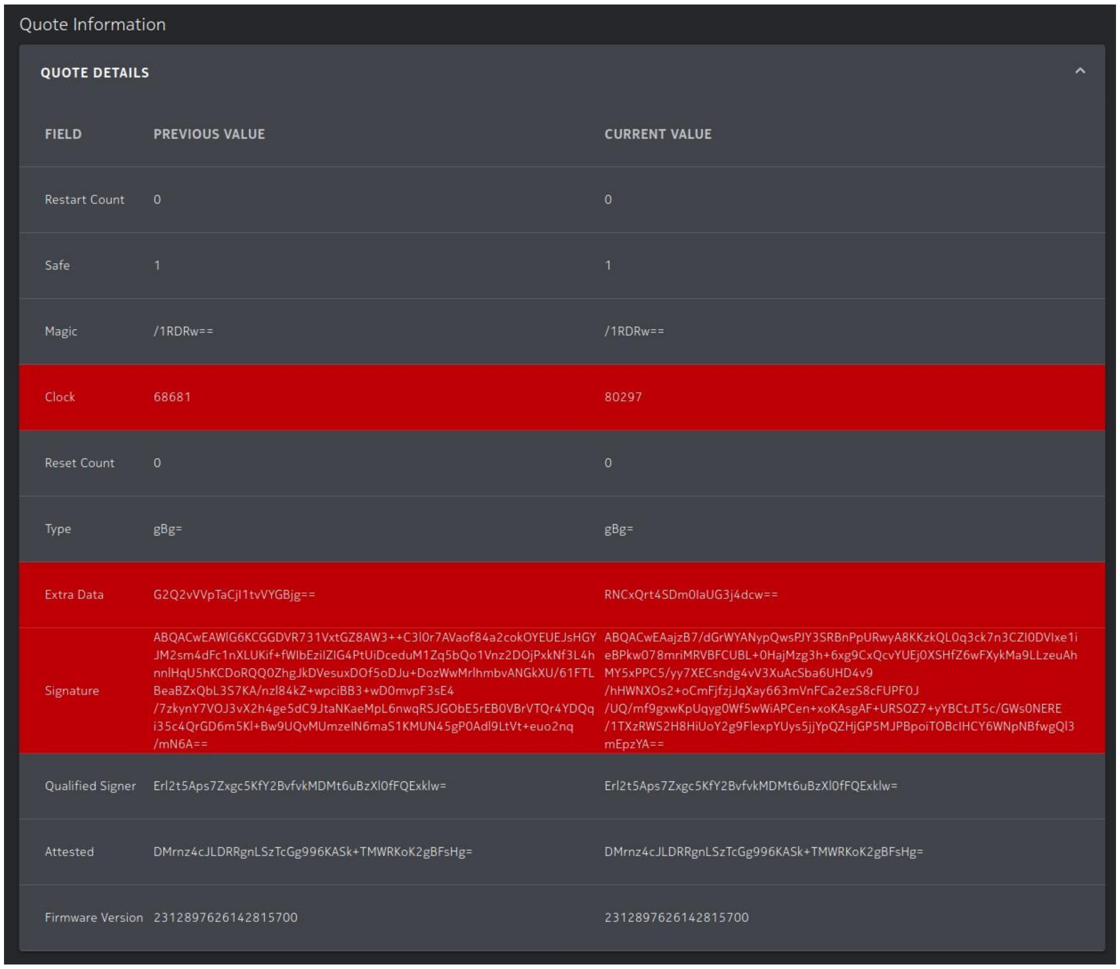
In this study, the TPM is used to attest PCR values. The TPM uses an attest structure that is signed by the provisioned AK. The structure contains a single hash of one or more PCRs referred to as a quote. (Trusted Platform Module Library: Part 1: Architecture 2019.)

The actual PCR hashes are not included in the quote. PCR hashes can be provided in plain text and then verified by the attestation service through the quote. The attestation service takes the plain text hashes and hashes them. If the hash matches the quote hash the plain text hashes can be considered same as on the attested TPM. (Trusted Platform Module Library: Part 1: Architecture 2019.)

Table 4. TPM ATTEST structure (Trusted Platform Module Library: Part 1: Architecture 2019).

Field	Description
Magic number	Prevents external data to be forged as an attest struct.
Quote type	Type of the attestation structure.
Qualified signer	Certifies the environment the signature was made
Extra data	Functions as an anti-replay nonce.
TPM clock states	Clock, resetCount, restartCount and Safe.
TPM firmware version	Can be used to only allow certain versions.
Attested	Signed hash of selected PCR values.

Table 4 shows the fields used by the TPM in the attest structure (Trusted Platform Module Library, Part 2: Structures 2016, p. 110). A quote from a simulated device on Nokia’s attestation service can be seen in Figure 11.


Quote Information

QUOTE DETAILS

FIELD	PREVIOUS VALUE	CURRENT VALUE
Restart Count	0	0
Safe	1	1
Magic	/1RDRw==	/1RDRw==
Clock	68681	80297
Reset Count	0	0
Type	gBg=	gBg=
Extra Data	G2Q2vVVpTaCjI1tvVYGBjg==	RNCxQrt4SDm0IaUG3j4dcw==
Signature	ABQACwEAWlG6KCGGDVR731VxtGZ8AW3++C3l0r7AVaof84a2cokOYEUEJsHGY JM2sm4dFc1nXLUKif+fWlbEziIZIG4PtUiDceduM1Zq5bQo1Vnz2DOjPxkNf3L4h nnlHqU5hKCDoRQQ0ZhgJkDVesuxDOf5oDJu+DozWwMrlhmbvANGkXU/61FTL BeaBZxQbL3S7KA/nzl84kZ+wpciBB3+wD0mvpF3sE4 /7zkynY7VOJ3vX2h4ge5dC9JtaNKaeMpL6nwqRSJGObE5rEB0VBrVTQr4YDQq i35c4QrGD6m5Kl+Bw9UQvMUmzeIN6maS1KMUN45gP0Adl9LtVt+euo2nq /mN6A==	ABQACwEAajzB7/dGrWYANypQwsPJY3SRBnPpURwyA8KKzkQL0q3ck7n3CZI0DVIxe1i eBPkw078mriMRVBFCUBL+0HajMzg3h+6xg9CxQcvYUEj0XSHfZ6wFXykMa9LLzeuAh MY5xPPC5/yy7XECsndg4vV3XuAcSba6UHD4v9 /hHWNXOs2+oCmFjfzjJqXay663mVnFCa2ezS8cFUPF0J /UQ/mf9gxwKpUqyg0Wf5wWiAPCen+xoKAsgAF+URSOZ7+yYBCtJT5c/GWs0NERE /1TXzRWS2H8HiUoY2g9FlexpYUys5jjYpQZHjGP5MJPBpoiTOBcIHCY6WNpNBfwgQl3 mEpzYA==
Qualified Signer	Erl2t5Aps7Zxgc5KfY2BvfvkMDMt6uBzXl0fFQExklw=	Erl2t5Aps7Zxgc5KfY2BvfvkMDMt6uBzXl0fFQExklw=
Attested	DMrnz4cJLDRRgnLSzTcGg996KASk+TMWRKoK2gBFsHg=	DMrnz4cJLDRRgnLSzTcGg996KASk+TMWRKoK2gBFsHg=
Firmware Version	2312897626142815700	2312897626142815700

Figure 11. Quote details shown in Nokia Bell Labs Attestation service.

5 SIMULATION ENVIRONMENT

The aim of simulation in this study is to reproduce a view of the traffic management system, including a way of monitoring the device integrity. In this study, integrity monitoring is included in another user interface (UI) but could be integrated to the traffic operators view in the future.

In order to be able to see changes in device integrity, the device is provisioned to an attestation service. The TPM attest feature is then used to report on PCR values.

Railway signals must only be tampered with, in a controlled environment, so as not to cause real incidents when testing. It is more secure, cost friendly and safer to simulate a rail security system virtually than using real equipment. When tampering with firmware on real devices, there is a great risk that the system will not be recoverable.

This chapter introduces the different roles implemented for future testing of the framework. In this chapter, only the simulation environment administrator’s role is introduced in detail. The other roles are introduced in Chapter 7.

In this chapter, the hardware and software needed to run the simulation are studied. The purpose is also to examine what services are needed in the process and how a particular device is simulated. This chapter ends with the documentation on how a small simulation without frontends can be run.

5.1 Simulation Environment Roles

Since the aim is that this study could be utilized as a test framework in the future, different roles are introduced for training, showcasing and testing. The end user in this test scenario is the traffic control operator. A simulation administrator is defined as the training instructor implementing the scenario for the traffic operator.

In addition to these roles, a simulation environment administrator may be necessary. This can, of course, be the same person as the simulation administrator if he has the required competence.

Simulated scenarios are launched and controlled interactively by the simulation administrator. Simulated railway traffic scenarios should be specified and implemented according to training purposes.

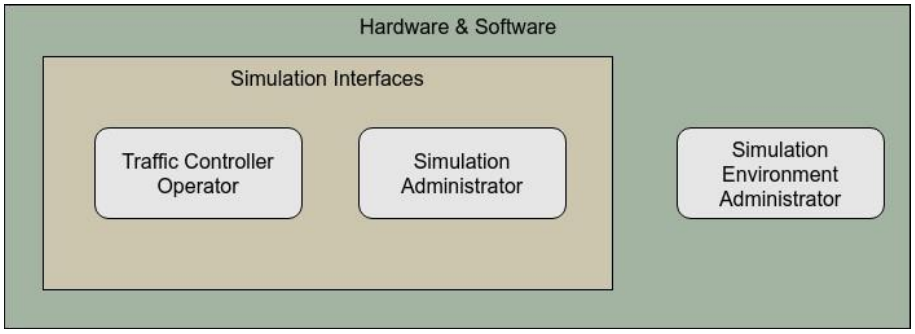


Figure 12. Simulation environment roles.

The infrastructure is managed by the environment administrator. Once the scenario is running on a system the administrator’s presence is not necessary during the simulation.

5.2 Simulation environment administrator

For managing the framework that the simulation run on, the simulation environments administrator uses common IT administrator tool’s. An overview of a framework example can be seen in Figure 13.

This environment can be run on a single computer with an installation of Docker, Node.js and Angular. However, the best practice would be to run Docker and the simulation admin interface on separate virtual machines.

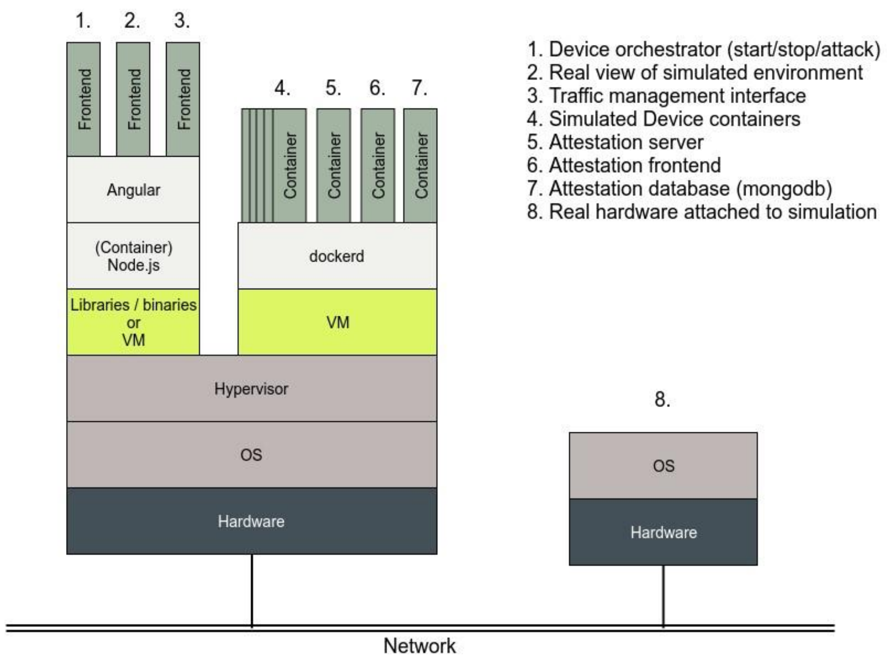

1.
2.
3.
Frontend
Frontend
Frontend
Angular
(Container) Node.js
Libraries / binaries or VM
4.
5.
6.
7.
Container
Container
Container
Container
dockerd
VM
Hypervisor
OS
Hardware
8.
OS
Hardware
Network
1. Device orchestrator (start/stop/attack)
2. Real view of simulated environment
3. Traffic management interface
4. Simulated Device containers
5. Attestation server
6. Attestation frontend
7. Attestation database (mongodb)
8. Real hardware attached to simulation

Figure 13. Overview of the simulation environment.

The simulation admin interface benefits from communicating directly to the host machine where the dockerd daemon is running. From the administrator interface, the simulation admin can create, start, stop and delete device containers.
It is also possible to run the admin interface as a container on a hypervisor separate from the Docker machine. This is how the simulation is performed in this thesis.

5.2.1 Containerization

Containerization was chosen as the base method for simulating devices. Fully virtualized computers need more resources and simulate the underlying hardware. Containers use the host system kernel and simulate only the operating system, behaving in the same manner independently of the hardware. Figure 14 demonstrates the difference between a container and a virtual machine (Schenker 2020.)

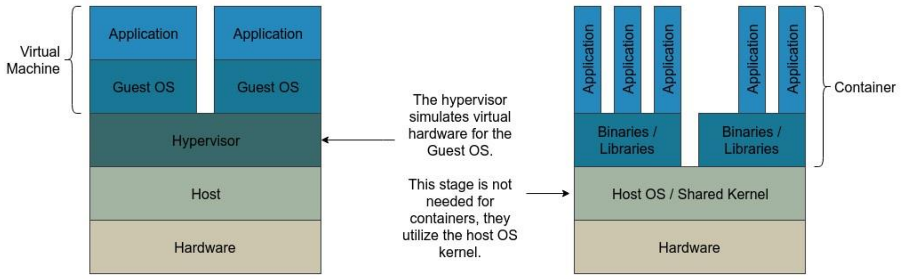


Figure 14. Containerization (Schenker 2020).

Docker was chosen as the containerization technology, but other technologies would also be applicable. Docker has good documentation online, and a software TPM simulation had already been made earlier by the author of the thesis.

Containers are made into run time environments from images by the dockerd service which uses Linux technologies as namespaces, control groups and the union file system to isolate the environment from the host operating system and other containers (Docker 2020). Containers are started from images that consist of at least one data layer with needed binaries. In its simplest form, an image can consist of one executable binary (Create a base image 2020).

Users mostly utilize Linux operating system images to build Docker containers. This thesis utilizes Ubuntu 20.04 Focal as its parent image. In order to be able to easily utilize TPM 2.0 tools, they are extended on top of the parent image.

The best practice of building Docker images is specifying a Dockerfile. This consists of instructions to run inside the parent image. If all instructions are successfully executed, the building process returns an image of the environment state. (Schenker 2020.)

The device Docker basic build file is available at Nokia GitHub (TPMCourse 2020). Table 5 illustrates the used tools for this study that are installed on top of the parent image.

Table 5. Software included in the device container.

Software	Version	Developer
Ubuntu binaries	20.04 Focal	Canonical
dbus	1.12.x.	freedesktop
tpm2-tss tpm2-abrmd tpm2-tools	2.3.2 2.3.1 3.2.1	tpm2-software community
IBM’s software TPM 2.0	1563	IBM
Trust Agent		Nokia

5.2.2 Hardware in the loop

In order to implement a one-to-one simulation of a platform boot, virtualized machines can be used running Open Virtual Machine Firmware (OVMF) with tianocore EDK II (OVMF FAQ 2019). Another option is to use real hardware.

Both options can be directly connected to the simulation environment. In this study, the use of real TPMs is fully supported by the Attestation service and the simulation environment. TPM EK certificate validation is also supported by the attestation service.

The benefits of trusted computing can, however, be showcased with a simplified platform boot simulated inside containers. This approach makes the simulation easy to showcase anywhere on a normal laptop. The speed of starting and resetting a simulation is very fast through device simulation. The whole simulation environment can be launched and reset in 2 minutes.

5.2.3 Simulation services

Docker needs a host with the simulation environment Docker images and a running daemon. In order to control the containers from the simulation admins frontend, the REST API needs to be opened for the daemon on the host. This configuration step is well defined in Dockers documentation (Docker 2020).

In order to connect the simulation containers a Docker created network is used. Docker’s internal Domain Name Server is used to automatically translate container hostnames to IP addresses. A container hostname can be given when starting the container.

By using DNS, it is trivial to automate configuration when starting up the simulation environment since no IP addresses need to be manually added or edited. Hostnames for attestation server, mongo database, device controller are some used and added to all config files in the building process.

The Node.js development server where the Angular simulation frontend is served need to have the Docker daemon host IP address implemented inside the Angular proxy.json file. In this study, we are using plain HTTP traffic between the frontend and the Docker server.

The test scenario containing device parameters is defined inside the Angular application folder. A config.json file describes all simulated containers and also the attestation containers. Different scenarios can be implemented with different config files.

Connections between rail devices are simulated by a Message Queuing Telemetry Transport (MQTT) broker. State information and commands from the security devices are transported with MQTT protocol version 3.1.1. The service is its own container running on the same Docker host.

5.2.4 Attestation service

Nokia Bell Labs in Espoo have been developing their own attestation server to showcase the benefits of trusted computing. This thesis is an extension of their work, targeting the rail industry.

The developed attestation server is implemented in Python and can be executed inside a Docker container. It interacts with attested devices through a REST API and utilizes a separate Mongo database. A user interface has been developed with Angular and runs on a Node.js development server. All of these services are usually supplied from three different Docker containers that can reside on the same dockerd server as the device containers.

The best way to begin a simulation is to first start the attestation services after which devices can be provisioned to it. The IBM’s TPM simulator does not have an Endorsement certificate. In this thesis, the IBM TPM implementation is considered trusted, and no endorsement certificate is implemented or checked during provisioning.

5.3 Simulating a Trusted Device

When a container is started, no processes are running as a default. Execution can be specified to one executable on the start-up. A container will also stop running if it does not have a task to do.

For simulating a device with a container, an execution script is needed to start-up more processes than one. Job execution needs to be considered because the container must stay alive even if it has no active task.

Because the aim is to simulate to some extent a real platform firmware three bash scripts execution is chained on the container start-up. The scripts are:

- crtm.sh
- uefi.sh
- application.sh

The crtm.sh starts execution on the start-up and initiates the software TPM. Then it measures itself, extends the hash to PCR 0. Next it moves on to measure and extend the hash of uefi.sh to the same PCR. Only after this, it hands over the execution to the uefi.sh script.

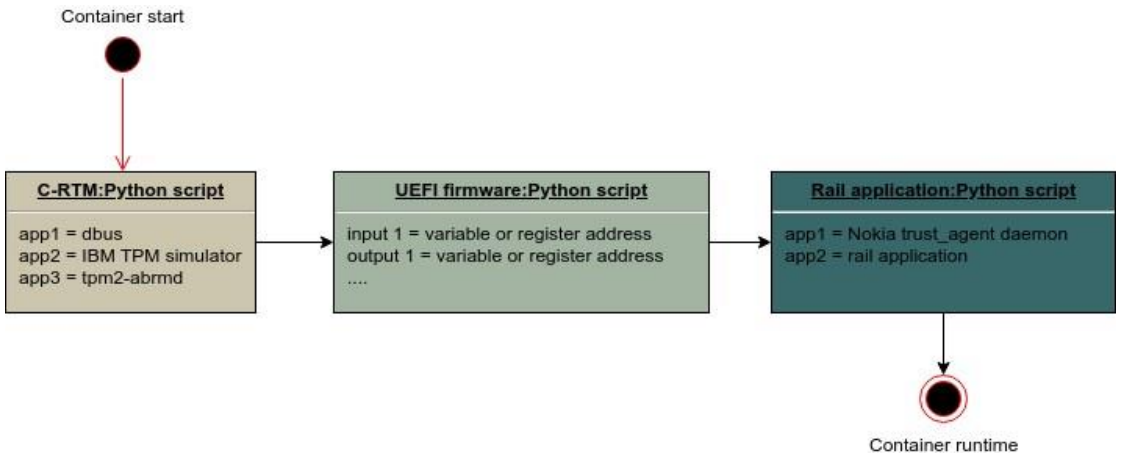

Container start
C-RTM:Python script
app1 = dbus
app2 = IBM TPM simulator
app3 = tpm2-abrmd
UEFI firmware:Python script
input 1 = variable or register address
output 1 = variable or register address
....
Rail application:Python script
app1 = Nokia trust_agent daemon
app2 = rail application
Container runtime

Figure 15. Container start up process.

The uefi.sh initializes bash input and output variables and extends them to PCR 1. These are later tied to the application that simulates the ICS device functionality. The ICS software is represented by the application.sh script which is measured in uefi.sh and extended to PCR 2. The chained execution ends with the application script that leaves the ICS software running.

The PCR values monitored in the simulated devices are PCRs 0-2. This range can simulate the CRTM, firmware and the application and was deemed sufficient to reach the aim of this thesis.

Inside application.sh, an automated attestation provision sequence is started upon the first start-up of the container. The sequence will first generate an EK and AK for the device. Secondly it will try to connect to the attestation service for provisioning. After a successful initial provisioning, the device will attest it’s firmware by reporting the PCR value range 0-2. Lastly the device generates a policy that considers the attested values trusted and attaches the policy to itself in the attestation service.

This automatic provision process helps rapid testing. Changes can be made to the image and it will be automatically trusted on the start-up. Table 6 shows the used environment variables for automatic provisioning. These are implemented in the scenario config file, inside the Angular application.

Table 6. Environment variables for device provisioning.

Variable	Usage
DEVICENAME	Name of the device in the attestation service
ATTESTATIONCLOUD	Cloud is the higher hierarchy to which the device belongs to. Policies can be attached to this group of devices.
ATTESTATIONSLICE	Lower hierarchy than cloud which the device can belong to. Policies can be attached to this group of devices.
ATTESTATIONPCRS	TPM PCR definition for the view in the frontend.
ATTESTATION-OPENSTACK	Not used in this simulation

The Nokia Trust Agent that communicates with the attestation server is also started at the end of every start-up.

A device reset is also needed for proper simulation. A reset is triggered by running a different script called restart.sh. The script generates a few empty files in the filesystem before it kills all other processes inside the container, before it hands over execution to crtm.sh. By looking up the generated files during start-up change the behaviour from initial start up to reset. This is important because we do not want the device to provision itself again on a reset.

The software TPM uses a file called NVCHIP as persistent memory if the file is found during TPM start-up. This way, seeds and keys can be stored between resets.

5.4 Starting an example environment

A description of the procedure for starting a small-scale environment with the attestation service and a simulated trusted device will conclude this chapter. All the services will be started on the same dockerd host and attached to a Docker network.

All containers will be started from the command line and not the simulation admin interface. The step of downloading or building the containers is omitted from this example. The sequence of starting the small-scale environment is the following:

1. Create the Docker container network.
2. Start the mongo database.
3. Start the attestation server.
4. Start attestation frontend (optional).
5. Start the device container.

Docker daemon start-up commands are in the same order as in Listing 1.

The name of the container image is in the end of the run command, this is usually specified as <image>:<version> or <framework>:<type>. The last part of the command specifies the command executed inside the container on start-up. The command is not needed if the container is built with it or an entry point. By specifying a command on start-up, the built-in command is ignored. In listing 1 the command is added for clarification of the command started.

Listing 1. Command examples.

```
docker network create simnet

docker run --name mongodb --network simnet --network-alias
   mongo_db --hostname mongo_db mongo:latest

docker run --name attestation --network simnet --network-alias
   server --hostname server --env "DATABASEFILE=/root/configs/
   db_rail_data.py" thesis:attestation runserver.sh

docker run --name frontend --network simnet thesis:
   attestationfrontend npm start

docker run -it --name device --network simnet \
    --env "POLICYSET=Basic policy set 0-2 pcrs" \
    --env "DEVICENAME=Axlecounter1" \
    --env "ATTESTATIONCLOUD=Rail Cloud" \
    --env "ATTESTATIONSLICE=Low Speed Slice" \
    --env "ATTESTATIONPCRS=Rail Pcrs" \
    --env "ATTESTATIONOPENSTACK=123456789" \
    thesis:device crtm.sh
```

With the network option in Listing 1, every container is attached to the simnet network. Both hostname and network-alias option is needed for DNS to work in lookup mode. The “env” option adds environment variables to the container which are used to specify attestation options on client provisioning.

6 CASE STUDY: RAIL TRAFFIC MANAGEMENT SYSTEM

The simulation environment described in this study is built to simulate a generic interlocking model. The Finnish railway security system is studied as a basis for the model.

The railway control system usually consists of sensors, controllers, turnouts, traffic lights and radios. The interlocking system is a controller containing a protective logic which is responsible for providing safe movement on track sections. Integrity failure on the device can have great negative impact on security.

This chapter gives an understanding of the Finnish electronic interlocking system, and how it protects train environment. The Finnish Rail Traffic Management System is studied from the book published by the Finnish transport infrastructure agency, this book is consistently used as a reference in this whole chapter (Kantamaa & Sorsimo 2018).

The first sub-chapter introduces the railway security terminology. The interlocking devices tasks are examined before introducing the track section used in this study.

6.1 The Finnish rail traffic management system

The Finnish rail traffic management system consist of four individual systems that work together:

- Interlocking
- Track vacancy monitoring
- Automatic Train Protection (ATP)
- Remote control system

The interlocking system is connected to all other systems and by a predefined logic developed through a strict standardized process, it controls other equipment. In this study, traffic signals are considered a part of the interlocking, because it controls the aspect of the lights.

The track vacancy monitor system provides input signals to the interlocking controller. Signals tell the interlocking system which track sections are occupied. Vacancy implementation is either done by track circuits or axle counters.

The Finnish Automated Train Protection (ATP) systems abbreviation is ATP-VR/RHK, this comes from the words valtion rautatie (VR) and ratahallintokeskus (RHK) in this study the system will be referred to as ATP. It consists of track side devices that can forward messages to a train integrated device, which is able to limit the train movement.

Track reservations for train movement are requested by the remote-control system from the interlocking system. The remote-control system cannot normally override the interlocking systems reservations.

6.2 Interlocking

Safety is the main purpose of the interlocking system. It takes vacancy input signals and grants access to remote-control requests if safe passage can be routed. The interlocking controls track side traffic signals and data sent to the ATP system.

The protective logic in the interlocking system is implemented as an application by device specific procedures and tools. The application development process is highly standardized, with steps that review, verify, simulate and test the logic before it can be used to protect a real track environment.

The interlocking device firmware initializes and maps the input and output interfaces to either variables or registers, which the protective logic then uses to interact with the track environment. Integrity verification during production use of an interlocking device, should consider measuring the firmware and the protective logic.

An overview of the interlocking interfaces is displayed in figure 16.

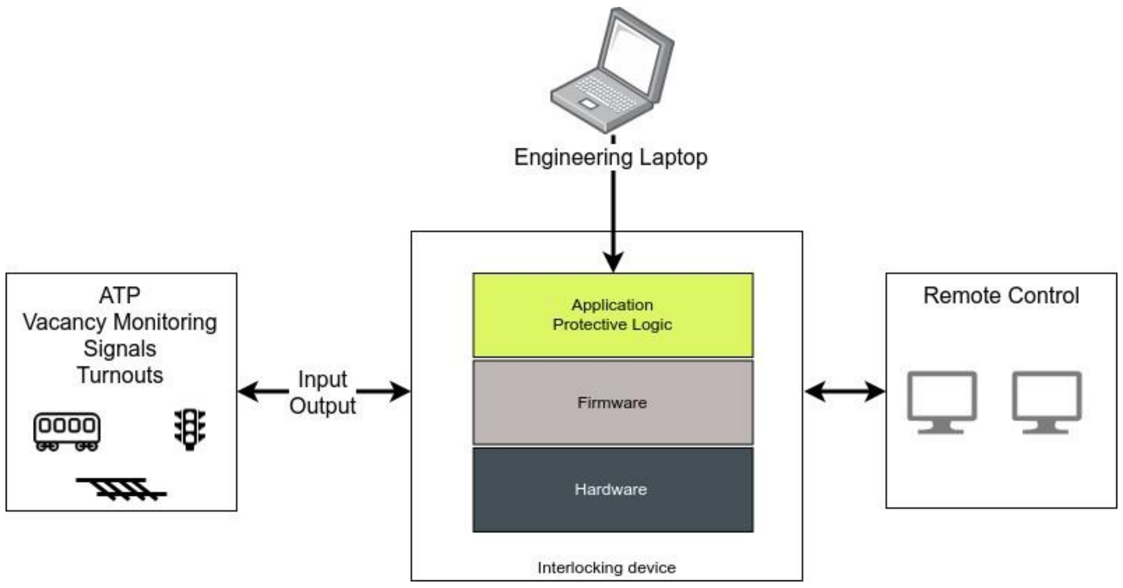


Figure 16. Interlocking devices interfaces (Kantamaa & Sorsimo 2018).

Interlocking device integrity failures during production use today, can only be spotted by the train driver (Hidden 1989). This can generate accidents before the integrity failure is mitigated.

Some interlocking devices are accessible from a Control System network for maintenance, this is an attack vector for the attacker (Stumpp 2019). By propagating through the IT and Control layer the interlocking device can be accessed. Without integrity checks no one can be sure if a device firmware or logic has been changed from the network.

6.3 Track section protection

The track section that is use throughout this study, is introduced in this sub-chapter together with some of the protective logic. The protection implemented is significantly stricter than Finnish track protection standards and only allows few input combinations for granting movement (Kantamaa & Sorsimo 2018).

Figure 17 shows the track section and enumerates all track side equipment. Every track block is assigned a plain number and can have the status of occupied or free. In this study the axle counter input is generated directly to a block.

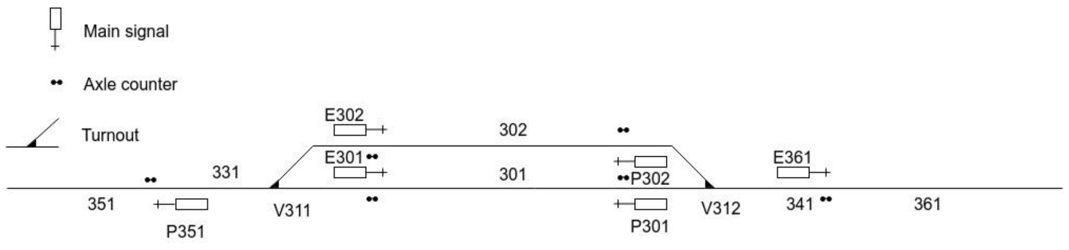


Figure 17. Track section.

In Figure 18 the train T100 occupies block 351 and train T200 block 302. In the figure a reservation is done for T100 on the main track, across the track section. As shown in this example of reservation, the interlocking device outputs signals to control the physical equipment in order to protect every train in the track section. The interlocking device controls in this scenario among other things (Kantamaa & Sorsimo 2018):

- the signal aspects.
- the turnout devices.

The interlocking device have means of verifying that the outputs are according to its sent commands, these are mostly implemented with analog signals. Different signals are used to verify that traffic lights are showing the right aspect and turnouts are pointing in the right direction.

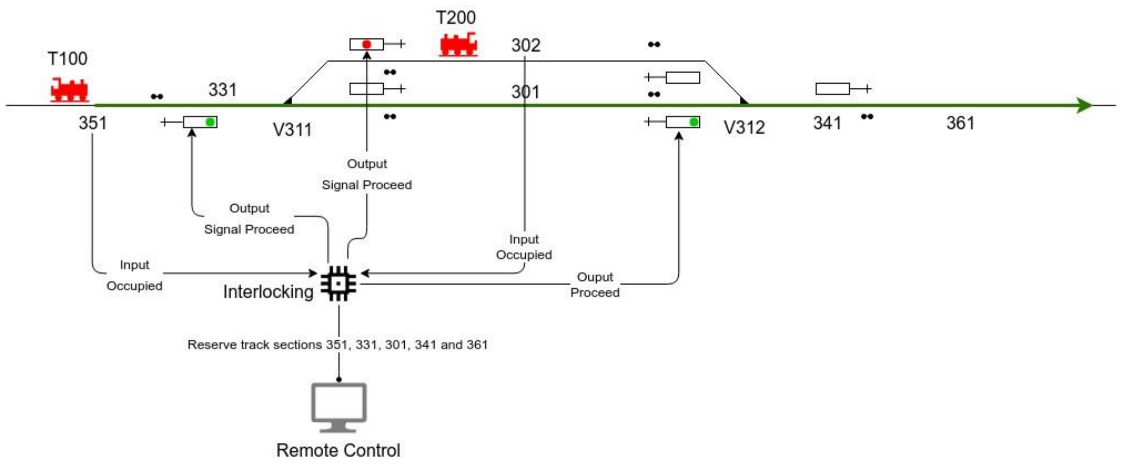


Figure 18. Reserved track section.

Figure 19 illustrates that the turnout is wired with inputs from the turnout that can signal through a closed circuit the position of the turnout. The outputs in this figure could be wired to separate relays controlling a motor, able to switch the turnout position from track block 301 to 302 and vice versa. These illustrated inputs and outputs will be used in the simulation to demonstrate what can happen when they are changed in the firmware mapping phase.

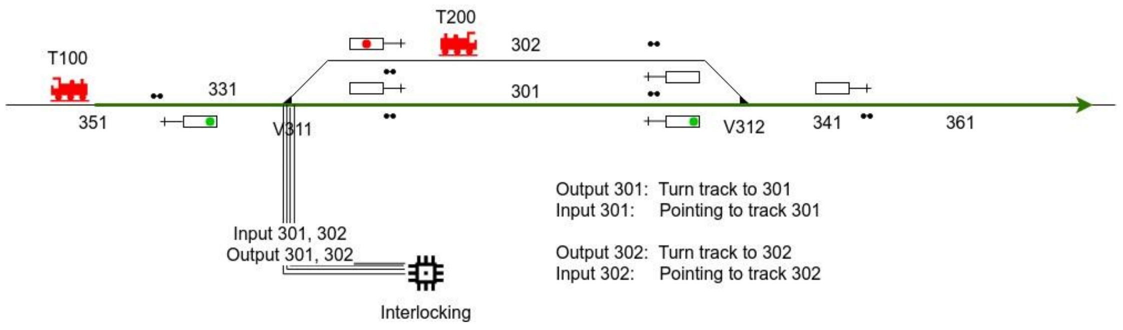

T200
T100
302
331
301
351
V311
V312
341
361
Output 301: Turn track to 301
Input 301: Pointing to track 301
Output 302: Turn track to 302
Input 302: Pointing to track 302
Input 301, 302
Output 301, 302
Interlocking

Figure 19. Turnout input and output.

A signal shows danger for T200, this signals the train not to proceed past the traffic light. In Finland there are three main signals which of two are used in this scenario, used aspects are proceed (green) and danger (red). (Kantamaa & Sorsimo 2018.) The interlocking protective logic or application restricts T200 to enter the main track section reserved for T100 movement with the aspect.

The implemented protective logic in the simulation only accepts green aspects for traffic lights showing proceed in Figure 19, they change to red after T100 has passed the signal. The turnout V311 is in all situations pointing to block 301, this is hard coded into the logic.

7 SIMULATED TRUSTED TRAFFIC MANAGEMENT SYSTEM

This chapter introduces the method of starting the simulation. Controls accessible to the simulation administrator and traffic management operator are examined.

The attestation service for assessing system integrity by the traffic management operator is examined in the last sub-chapter.

7.1 Simulation administrator’s perspective

The simulation administrator specifies the scenario, devices and track section before the simulation is launched. If necessary, this can be done together with the environment administrator. In this study, the scenario introduced in the previous chapter is prepared and starts automatically upon simulation start-up. Configurations are propagated automatically to the following services and devices:

- Administrator frontend.
- Attestation services.
- Interlocking device.
- Other device containers.

It is recommended to start the mongo database service first, a few seconds before the attestation container. The attestation container will however delay the start-up until it can acquire a DNS response from the mongo_db host.

The simulation administrator can start all device containers from the User Interface (UI) in Figure 20. In this example, all visible containers are already started except the MQTT broker. In order to help resetting the scenario, buttons for controlling all devices containers are generated under the device container heading.

A Docker container status update is requested once a second from the dockerd daemon, this helps the administrator to monitor the state of the containers. The update can tell the administrator if a container stopped for an unknown reason, which helps troubleshooting containers when developing scenarios. The update feature can be toggled on and off from the refresh button.

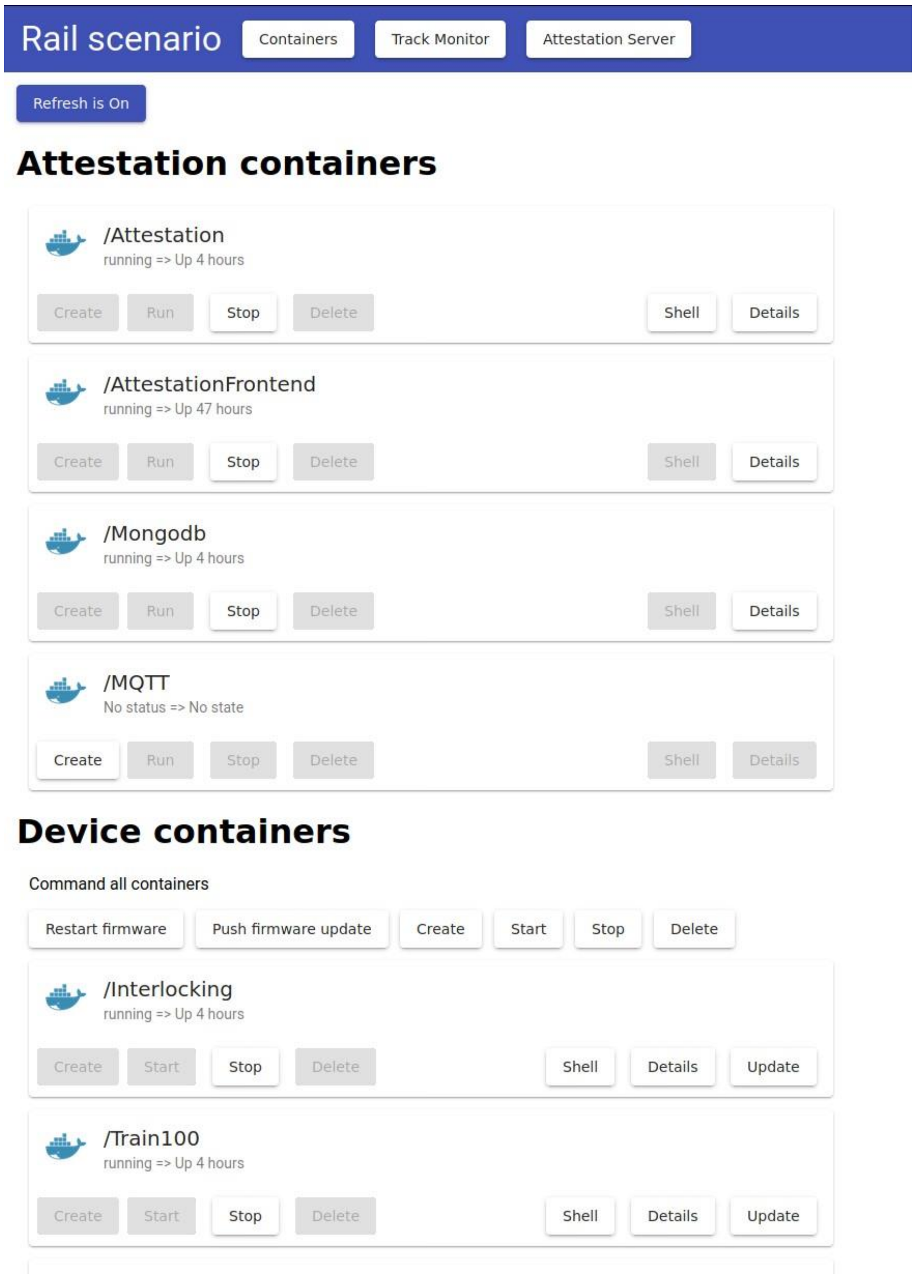


Figure 20. Simulation administrator UI.

The simulation administrator has another UI to control the scenario and monitor the track situation, this shows the real view of the track section. The UI view can be seen in Figure 21 and is otherwise the same as the operators but shows the real state of track devices. Operators UI shows the track state interpreted by the interlocking device. Controls for setting the scenario and moving the train is included in the administrators UI.

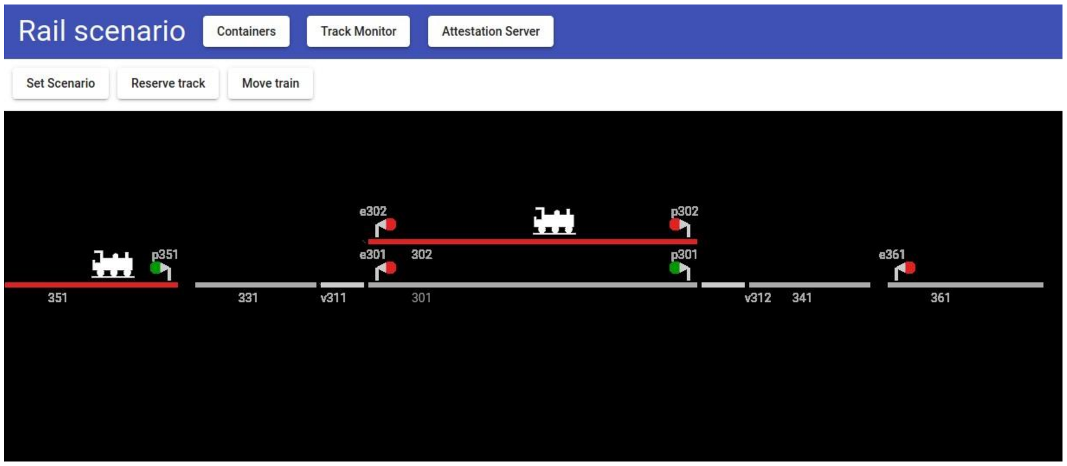


Figure 21. Simulation administrator real view.

By changing the inputs from a turnout device vice versa in the interlocking containers application or firmware changes the perception of the track state for the operator. The interlocking device believes that the turnout points to track block 302 when it in fact points to 301, this deviation can be seen by examining both UIs.

The train scenario simulation can be set and controlled in the real track view. The button, set scenario, initializes the simulation scenario and sets trains in the starting position as introduced. The administrator can move the train T100 forward to simulate movement across the track section.

7.2 Traffic management operator’s perspective

The traffic operator has a similar track section UI as the simulation administrator but without the control buttons and the real track view. The view is displayed in Figure 22.

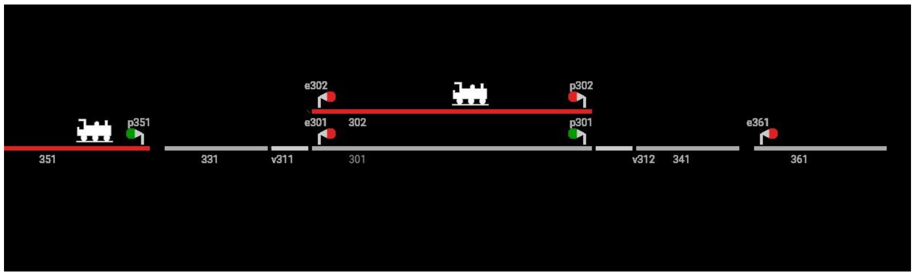


Figure 22. Operator view of the track section.

Displayed in Figure 23 is the attestation UI for integrity monitoring and validation, this is accessible for the operator. The figure shows the main view were devices are assigned to a cloud and a trust slice, which is done automatically when the devices are provisioned.

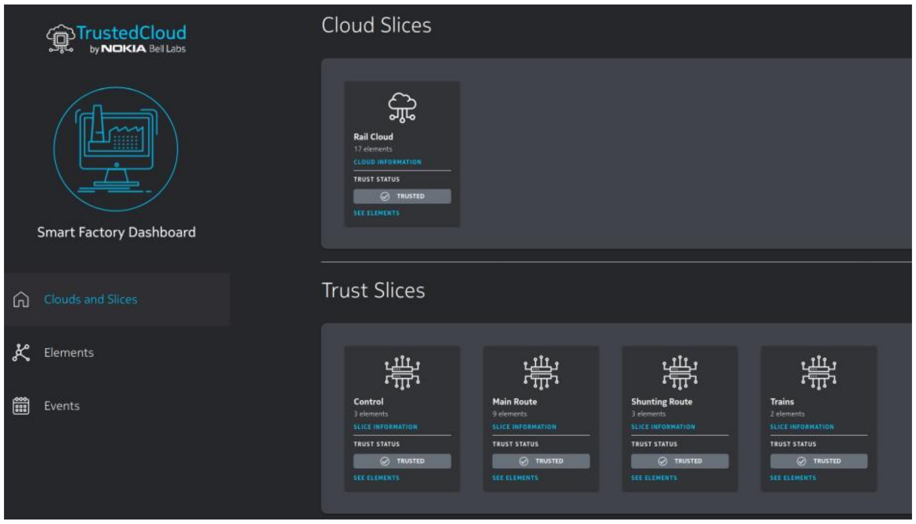


Figure 23. Operator view of attestation UI.

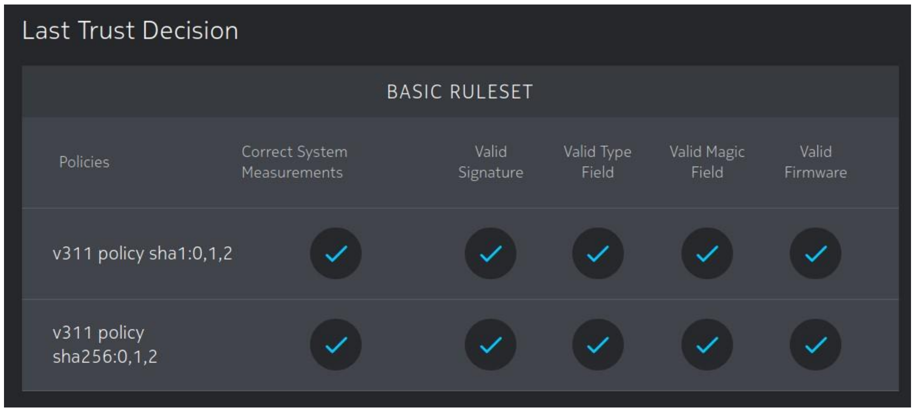
Depending on the ruleset different properties are measured when attested, a basic ruleset is used in this study for the devices. Each trust slice can have its own ruleset and rules. The rules attached to the simulated devices in this study are:

- Correct System Measurements
- Valid Signature
- Valid Type Field
- Valid Magic Field
- Valid Firmware

Correct system measurements are validated if the monitored PCRs contain values that are considered trusted, this is measured by the attest hash which is a hash of the monitored PCR values. In the study simulation PCRs 0-2 are values monitored.

The signature is valid if the attest structure is signed by the corresponding AK that the device was provisioned with. Type validation reflects on the attestation structure and attested parameters. The magic field indicates that the attest structure begins with a predefined value which restricts the TPM from only signing an attest structure generated inside the TPM. Firmware validation verifies the version of the TPM in the attested device (Trusted Platform Module Library: Part 1: Architecture. 2019.)

Validated trust decisions displayed in Figure 24 can be explored in the attestation UI by going into an elements details.



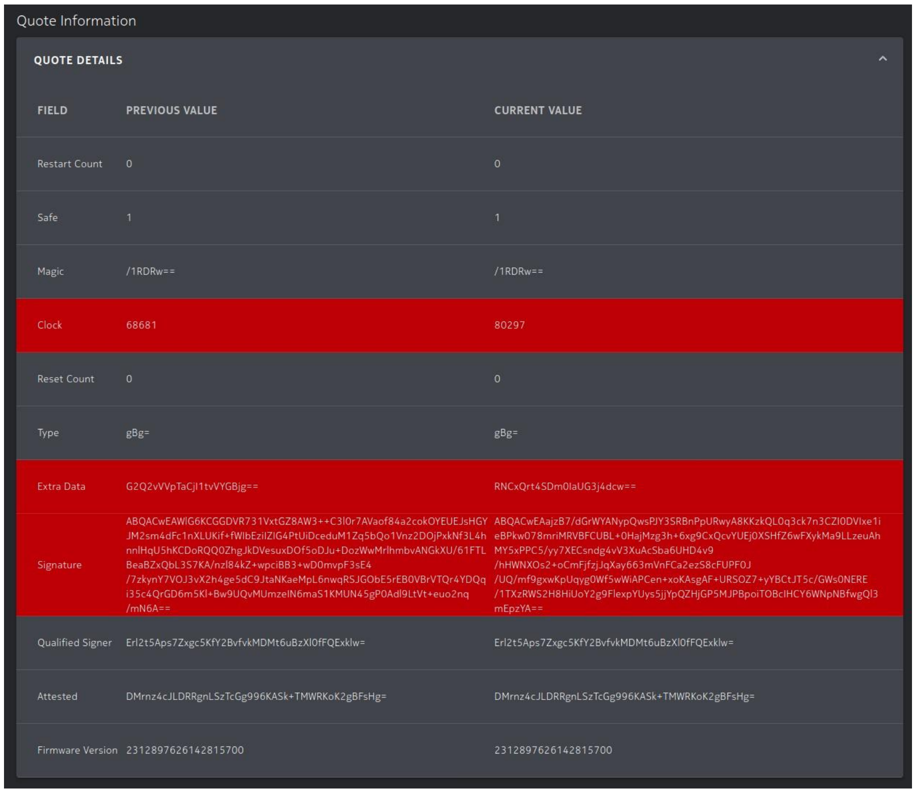
Last Trust Decision

BASIC RULESET

Policies	Correct System Measurements	Valid Signature	Valid Type Field	Valid Magic Field	Valid Firmware
v311 policy sha1:0,1,2	✓	✓	✓	✓	✓
v311 policy sha256:0,1,2	✓	✓	✓	✓	✓

Figure 24. Last trust decision

Attesting a device can be done from the UI, every device in the study simulation automatically attests itself on reset. When a device is attested it will return a quote with information to validate the ruleset. Quote details are accessible from the UI, these are displayed in Figure 25.


Quote Information

QUOTE DETAILS

FIELD	PREVIOUS VALUE	CURRENT VALUE
Restart Count	0	0
Safe	1	1
Magic	/1RDRw==	/1RDRw==
Clock	68681	80297
Reset Count	0	0
Type	gBg=	gBg=
Extra Data	G2Q2vVVpTaCjI1tvVYGBjg==	RNCxQrt4SDm0IaUG3j4dcw==
Signature	ABQACwEAWlG6KCGGDVR731VxtGZ8AW3++C3l0r7AVaof84a2cokOYEUEJsHGY JM2sm4dFc1nXLUKif+fWlbEzilZlG4PtUiDceduM1Zq5bQo1Vnz2DOjPxkNf3L4h nnlHqU5hKCDoRQQ0ZhgJkDVesuxDOf5oDJu+DozWwMrlhmbvANGkXU/61FTL BeaBZxQbL3S7KA/nzl84kZ+wpciBB3+wD0mvpF3sE4 /7zkynY7VOJ3vX2h4ge5dC9JtaNKaeMpL6nwqRSJGObE5rEB0VBrVTQr4YDQq i35c4QrGD6m5Kl+Bw9UQvMUmzeIN6maS1KMUN45gP0Adl9LtVt+euo2nq /mN6A==	ABQACwEAajzB7/dGrWYANypQwsPJY3SRBnPpURwyA8KKzkQL0q3ck7n3CZI0DVIxe1i eBPkw078mriMRVBFCUBL+0HajMzg3h+6xg9CxQcvYUEj0XSHfZ6wFXykMa9LLzeuAh MY5xPPC5/yy7XECsndg4vV3XuAcSba6UHD4v9 /hHWNXOs2+oCmFjfzjJqXay663mVnFCa2ezS8cFUPF0J /UQ/mf9gxwKpUqyg0Wf5wWiAPCen+xoKAsgAF+URSOZ7+yYBCtJT5c/GWs0NERE /1TXzRWS2H8HiUoY2g9FlexpYUys5jjYpQZHjGP5MJPBpoiTOBcIHCY6WNpNBfwgQl3 mEpzYA==
Qualified Signer	Erl2t5Aps7Zxgc5KfY2BvfvkMDMt6uBzXl0fFQExklw=	Erl2t5Aps7Zxgc5KfY2BvfvkMDMt6uBzXl0fFQExklw=
Attested	DMrnz4cJLDRRgnLSzTcGg996KASk+TMWRKoK2gBFsHg=	DMrnz4cJLDRRgnLSzTcGg996KASk+TMWRKoK2gBFsHg=
Firmware Version	2312897626142815700	2312897626142815700

Figure 25. Quote details

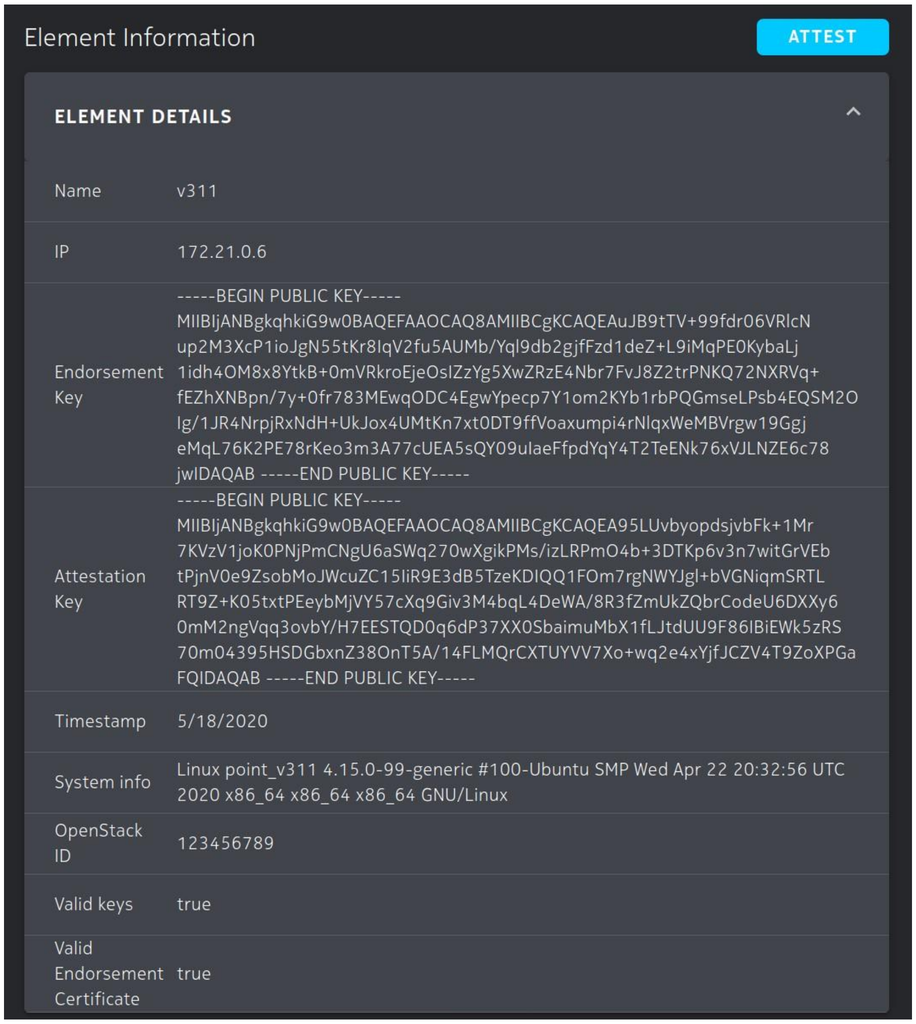
Restart count reflects on device restart, this value is not used in the study. Safe and extra data are other values omitted from this study. The purpose of the magic, type, signature and firmware fields were explained, in Figure 25 the values can be studied.

Fields in red contain values that have changed between quotes. Clock value should increase which makes the signature also differ between quotes. Extended rules can be generated on values in the quote, for example a change in the clock backwards could easily be detected by a rule.

Reset count reflects on TPM reset which happens on a device restart. On a simulated restart this value increases, the restart count is omitted. Qualified signer is the AK name in this study, this is supplied to the attestation service when the device is provisioned.

The most interesting value in the quote for this study is attested, this is the hash of the monitored PCRs. During the last phase of device provisioning in this simulation, the device measures itself and generates the attest hash in the rule. This is the trusted value when the simulation is started and reflects on the correct system measurement rule.

General element details are accessible from the attestation UI, these are displayed in Figure 26. The details contain the name and IP address that was applied during device provisioning. Openstack ID can be used for additional identity information but omitted in this study. Both EK and AK public keys are displayed in the figure which after a timestamp is listed from the day the device was provisioned.



Name	v311
IP	172.21.0.6
Endorsement Key	-----BEGIN PUBLIC KEY----- MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuJB9tTV+99fdr06VRlcN up2M3XcP1ioJgN55tKr8IqV2fu5AUMb/Yql9db2gjfFzd1deZ+L9iMqPE0KybaLj 1idh4OM8x8YtkB+0mVRkroEjeOslZzYg5XwZRzE4Nbr7FvJ8Z2trPNKQ72NXRVq+ fEZhXNBpn/7y+0fr783MEwqODC4EgwYpecp7Y1om2KYb1rbPQGmseLPsb4EQSM2O lg/1JR4NrpjRxNdH+UkJox4UMtKn7xt0DT9ffVoaxumpi4rNlqxWeMBVrgw19Ggj eMqL76K2PE78rKeo3m3A77cUEA5sQY09ulaeFfpdYqY4T2TeENk76xVJLNZE6c78 jwIDAQAB -----END PUBLIC KEY-----
Attestation Key	-----BEGIN PUBLIC KEY----- MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA95LUvbyopdsjvbFk+1Mr 7KVzV1joK0PNjPmCNgU6aSWq270wXgikPMs/izLRPmO4b+3DTKp6v3n7witGrVEb tPjnV0e9ZsobMoJWcuZC15IiR9E3dB5TzeKDIQQ1FOm7rgNWYJgl+bVGNiqmSRTL RT9Z+K05txtPEeybMjVY57cXq9Giv3M4bqL4DeWA/8R3fZmUkZQbrCodeU6DXXy6 0mM2ngVqq3ovbY/H7EESTQD0q6dP37XX0SbaimuMbX1fLJtdUU9F86IBiEWk5zRS 70m04395HSDGbxnZ38OnT5A/14FLMQrCXTUYVV7Xo+wq2e4xYjfJCZV4T9ZoXPGa FQIDAQAB -----END PUBLIC KEY-----
Timestamp	5/18/2020
System info	Linux point_v311 4.15.0-99-generic #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
OpenStack ID	123456789
Valid keys	true
Valid Endorsement Certificate	true

Figure 26. Element information.

The validation of keys and the EK certificate is displayed in the end of the information in Figure 26. In this study scenario keys are valid if they have the correct attestation key attributes explained in Chapter 4. The server can check the EK certificate if it has access to the CA chain, this feature is omitted from the study.

In the home view of the attestation UI displayed in Figure 26, the operator can with a single view acknowledge that every device attached to the simulation is trusted. The traffic management system can be considered trusted according to the ruleset applied, because every validation check assigned has been successfully validated. In the next chapter an untrusted view is examined.

8 ATTACKING THE TRUSTED RAILWAY SIMULATION

Attacking the simulation is introduced in this chapter, this is done through the simulation administrator role. How to tamper with the interlocking firmware and generate a critical flaw, that will generate a dangerous situation is explained in this chapter.

The chapter concludes with explaining how the firmware change can be discovered with the attestation UI and what kind of mitigation procedures could help fixing the critical situation.

8.1 Tampering a simulated interlocking device

The attacks studied in Chapter 3 indicate that propagation through the IT layer can give access to the ICS layer for an attacker. No simulation of propagation is implemented in this study, the aim is to discover attacks after tampering has happened. In order to tamper with the interlocking device, the simulation administrator directly edits the firmware and triggers a restart.

The uefi.sh script acquires virtual inputs and outputs during start-up, which are mapped to variables that the security application use. In Figure 27 input 1 is mapped to the application variable input 301 which represents the connection to track block 301 in turnout V311. When turnout V311 is pointing to track block

301, an active signal is generated from the turnout device to the application variable.

The interlocking operates its outputs according to the application logic.

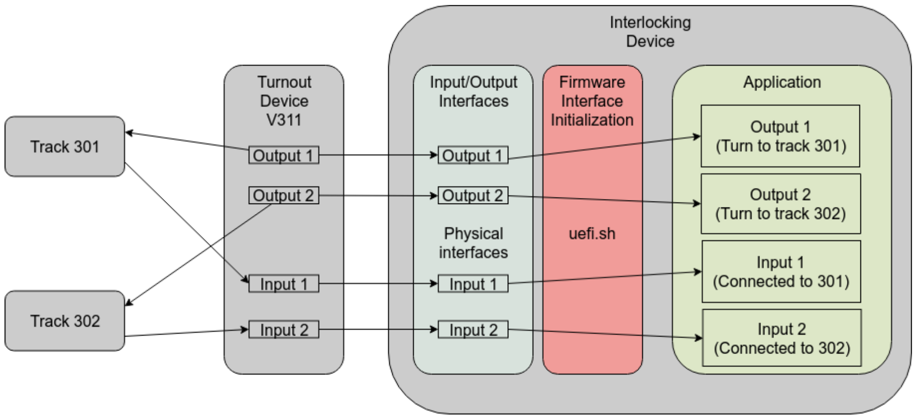


Figure 27. Initial uefi.sh mapping.

When the interlocking requires to switch a turnout, it generates a signal on the corresponding output to which it wants the turnout to point. In this simulation an input will immediately indicate that the turnout has switched to another track block. An overview of the track section and the connections from the turnout is displayed in Figure 28.

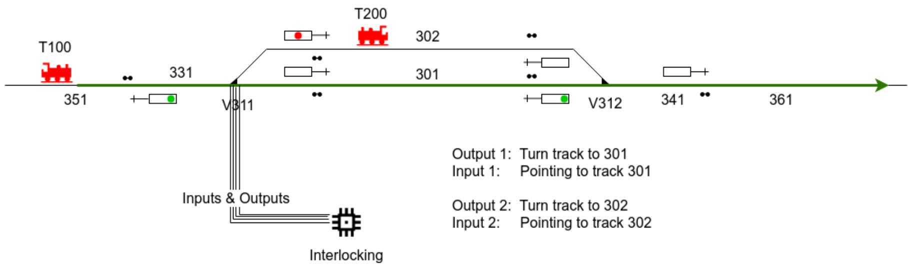

T200
T100
302
301
331
351
V311
V312
341
361
Inputs & Outputs
Interlocking
Output 1: Turn track to 301
Input 1: Pointing to track 301
Output 2: Turn track to 302
Input 2: Pointing to track 302

Figure 28. Inputs and outputs from interlocking to V311.

In order to simulate an error that is not visible in the traffic operators normal track UI, the input and output mappings are changed in the uefi.sh script. Variables represent the physical interfaces in the script, the simulation administrator can directly edit the variable values inside uefi.sh which resides in the interlocking containers file system.

Table 8 shows the original variable values and how they can be flipped to introduce a dangerous situation for the simulated track environment. Figure 29 illustrates how the flipped mappings connect to the application logic.

Table 7. uefi.sh mapping variables.

Variable	Original Value	Edited value
INPUT1	301	302
INPUT2	302	301
OUTPUT1	301	302
OUTPUT2	302	301

After the file has been edited by the simulation administrator, a restart is required for the changes to take effect. Restarting can be done from inside the interlocking container through the restart.sh script or starting the container after it has been stopped. Crtm.sh starts the boot process and hands over execution to the uefi.sh script which will extend the variable values to PCR 1 before mapping them to Linux environment variables.

Last step in the boot sequence starts after the application.sh is extended to PCR 2. The interlocking application is the last to start after a successful restart.

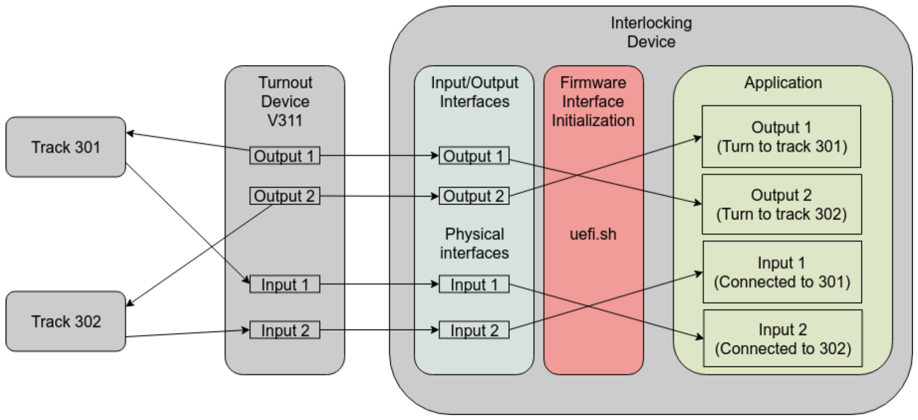


Figure 29. Flipped uefi.sh mapping.

The uefi.sh is the glue between the physical interface and the application interface. The interlocking software loads its protective logic last with the variables provided by uefi.sh. No change is done in the protective logic inside the train application to affect the end functionality.

8.2 Characteristics of attacks

The PCR measurements of register 1 is incorrect after the explained tampering and restart of the interlocking device. The reported value does not match the initial measurement made on the containers first start up.

The change in mappings cannot be spotted by the traffic operator in his UI shown in Figure 30. However, the turnout V311 is pointing to track block 302 even if it is showing it as pointing to 301, this is confirmed in the simulation administrator's UI in Figure 31.

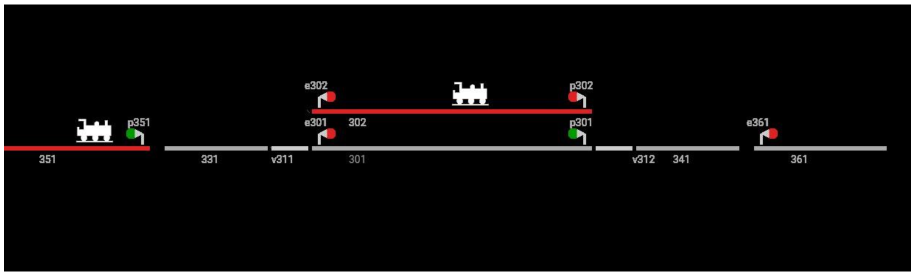


Figure 30. Operator view of turnout V311.

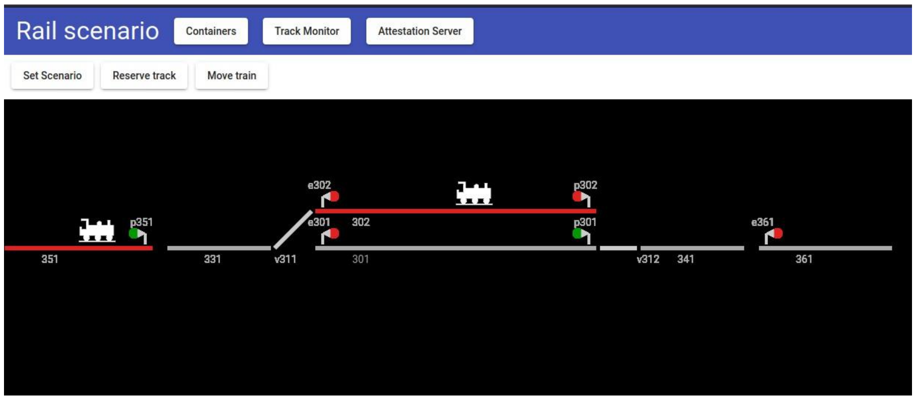


Figure 31.Real view of turnout V311.

By using the attestation UI, the traffic management operator can spot an integrity change in the interlocking device. The interlocking device belongs to the Rail Cloud and the Control slice, both instances are displayed in Figure 32 as untrusted with a red color.

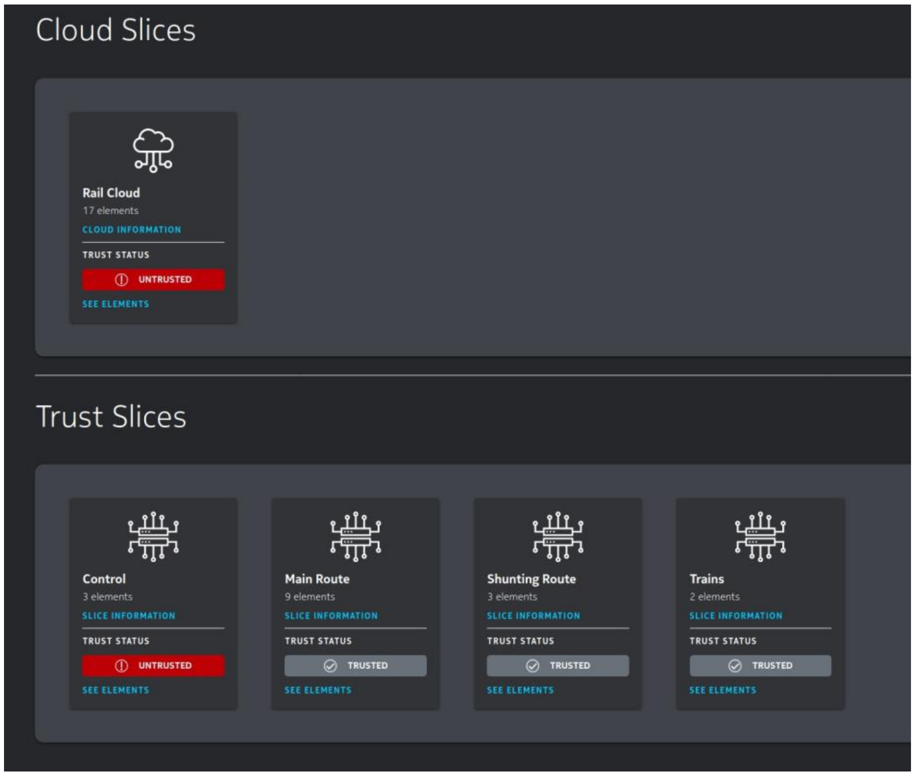
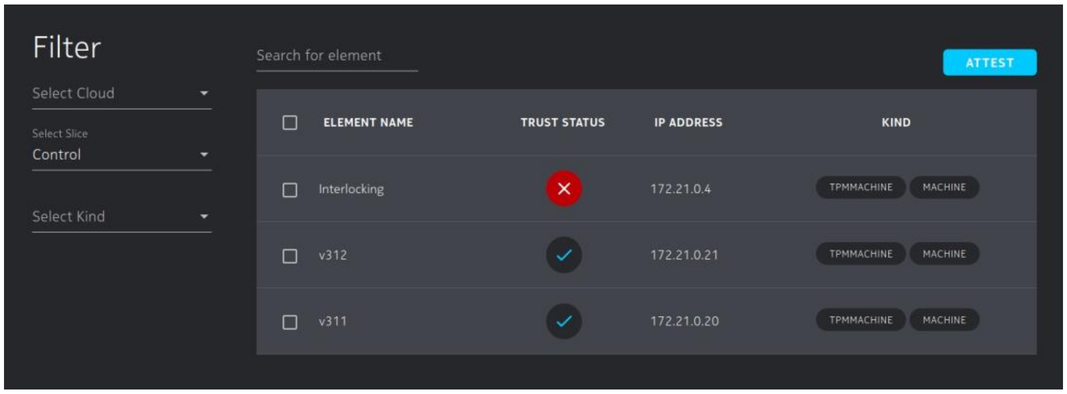


Figure 32. Untrusted control slice.

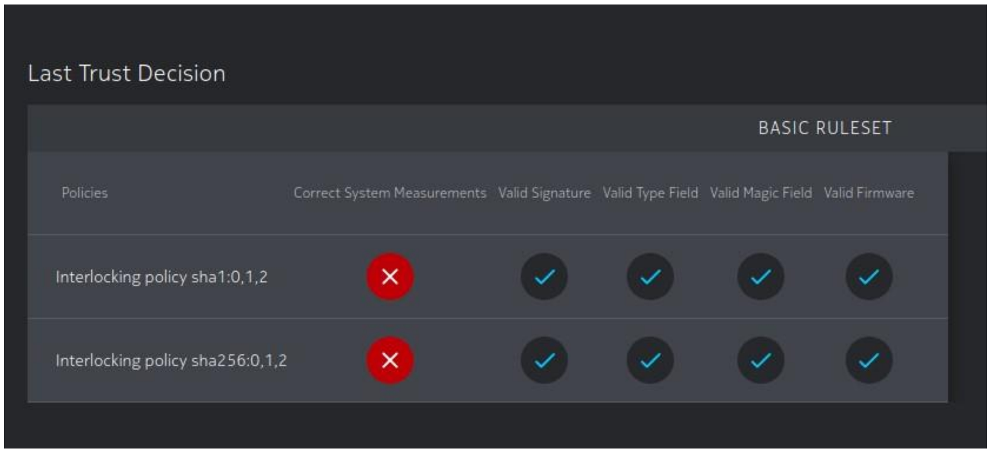
The operator can open the slice and view all devices trust state attached to it. A red indicator shows that there is an issue with the interlocking device in Figure 33. Details of the device can be studied by clicking on the name of the device.



ELEMENT NAME	TRUST STATUS	IP ADDRESS	KIND
Interlocking	✗	172.21.0.4	TPMMACHINE MACHINE
v312	✓	172.21.0.21	TPMMACHINE MACHINE
v311	✓	172.21.0.20	TPMMACHINE MACHINE

Figure 33. Untrusted interlocking container.

The last trust decision is displayed in the details in Figure 34. The correct system measurements rule has failed validation in the figure, this means some of the PCR values are not considered trusted.

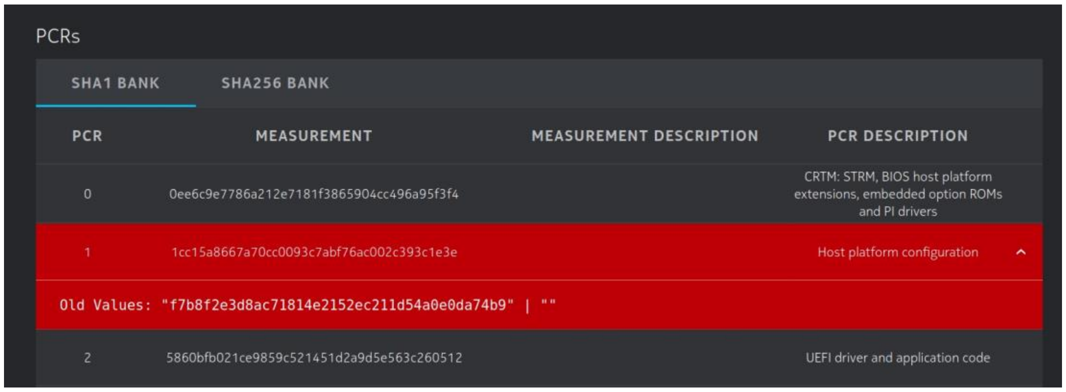

Last Trust Decision

BASIC RULESET

Policies	Correct System Measurements	Valid Signature	Valid Type Field	Valid Magic Field	Valid Firmware
Interlocking policy sha1:0,1,2	✗	✓	✓	✓	✓
Interlocking policy sha256:0,1,2	✗	✓	✓	✓	✓

Figure 34. Last Trust Decision.

By opening the PCR values as displayed in Figure 35, the operator can spot that the issue is generated from an untrusted value in PCR 1. This relates back to the flipped inputs and outputs. Even if the operator do not identify the source of the problem he identify at which software level the mitigation has to start.


PCRs

SHA1 BANK | SHA256 BANK

PCR	MEASUREMENT	MEASUREMENT DESCRIPTION	PCR DESCRIPTION
0	0ee6c9e7786a212e7181f3865904cc496a95f3f4		CRTM: STRM, BIOS host platform extensions, embedded option ROMs and PI drivers
1	1cc15a8667a70cc0093c7abf76ac002c393c1e3e		Host platform configuration
Old Values: "f7b8f2e3d8ac71814e2152ec211d54a0e0da74b9" | ""			
2	5860bfb021ce9859c521451d2a9d5e563c260512		UEFI driver and application code

Figure 35. Host Platform measurements untrusted.

The attestation trust view could be propagated onto the traffic management control UI. This could show device integrity failures immediately when the attestation service spots them and would not require the operator to open a different UI.

Continuing operations without spotting an integrity failure presented, the train would most probably derail at V311 or crash into the other train on track block 302. This would depend on the speed of the train and the length of the V311 turnout.

Turnouts can have a lower speed limit than the one on the main track when turning the train to another track. Depending on the length of the turnout the speed limit can be as low as 35km per hour. The main track in the presented scenario could still support speeds up to 160km per hour. (Kantamaa & Sorsimo 2018.)

There are previous incidents where train accidents have happened due to flipped analog signals. Without integrity monitoring, only the train driver can spot the turnout fault introduced in this study. (Hidden 1989)

8.3 Mitigation

Mitigation of an integrity failure of a device depends on the security procedure. When a device fails the rule check a strict protection policy could be put in place to only allow manually controlled movement. However, the attestation feature introduced in this study could provide a tool to impose finer policies on failure.

Testing and verifying that certain elements can provided sufficient protection even with a certain failure, could speed up traffic movement during mitigation.

The TPM approach never leaves the device inoperable even if integrity tests fail, this is the opposite of what PKI verification provides. This is especially an advantage for remotely located devices, which still could be updated remotely and attested after failure.

9 CONCLUSION

Presented in this chapter are the answers to the thesis questions:

- How can a firmware integrity failure in the security system lead to a dangerous situation in the rail environment?
- How can a railway traffic operator validate the firmware integrity of the security system?

The questions aim to find a solution for the problem on noticing and mitigating integrity failures in ICS-systems. The first question seeks answers of the necessity in integrity measurements in the rail environment. The second question seeks a solution for noticing integrity failures and help in mitigation of the problem.

Indications can be made by studying the attacks presented in the background literature, that there is a need for firmware integrity monitoring in the ICS layer. In the case of Stuxnet and Triton, it would have provided an indication of a deviation from the normal software.

In the simulation an attack similar to Stuxnet were implemented into the rail environment, this changed interface mappings in the firmware before the protective software is executed. Even if the protective applications integrity would be monitored and intact, the cyber-physical functionality would have changed. This indicates that the integrity monitoring should be implemented in the lowest level possible.

The simulation results show that by manipulating the firmware, in the railway security system dangerous interlocking outputs can be produced. These can be discovered by the implemented TPM and attestation solution. This only considers checking the integrity during device start up, no dynamic checking is implemented in this study.

By restricting firmware and software changes to only take affect after a reboot, integrity monitoring with the presented solution will work without dynamic integrity monitoring. ICS devices mostly function in a predefined which fits the static TPM measurements.

Validation and maintenance of trusted values was omitted from this thesis. In the study the operator was monitoring trusted values that were established by reasoning. It is not the optimal way to establish trust, but it still provides means of discovering tampering and it can be extended later to reflect on trusted values reported by developers or manufacturers.

Additional personnel could be alerted to evaluate the trust state of a device which validation fails against the device ruleset. By using attestation, the devices that fail integrity checks are noticed, it can even pinpoint the software level where integrity have degraded. This helps alerting the right maintenance for mitigation.

10 DISCUSSION

A discussion on the general success of the study, reliability analysis and some feedback on the theory studied in this thesis are included in this chapter.
Future research and study topics conclude this chapter.

Combining all pieces together for this study was a challenging task, Trusted Computing and the TPM alone includes a large amount of information. Implementing the TPM to the rail environment simulated in this thesis is possible but going forward needs more insights from the industry. The interaction with the assigner of the work and the railway industry was crucial for this work to succeed.

10.1 Reliability analysis and feedback

Interlocking devices were studied in a general aspect in this thesis. The tampering procedure where implemented with background information from previous ICS attacks and PC platform knowledge from the UEFI specification. Therefore, it is not known how many interlocking devices could be affected by an attack aiming for changing interface mappings.

Additional security implementations than the interlocking system were not simulated, these could provide features that would mitigate the dangerous situation presented in this study. However, the interlocking system is complex and can consist of hundreds of inputs and outputs. Changes or failures in the firmware or software can most probably in some case present an unwanted

Publications on ICS attacks could state more clearly the affected layer, in most cases it is unclear if the firmware level where compromised or only controlled through the IT layer. This tendency can make companies, industries and organizations focus on wrong security implementations. The ICS kill chain can be used to describe the compromised layer in detail (Assante & Lee 2015).

10.2 Suggestions for further study

Extensions on this thesis could include:

- Availability of TPM implementations in interlocking devices
- Simulation with TPM implementation in a real rail simulator

First the TPM availability should be studied of the devices in the industry, if there are no implementations the following studies could be done.

One study could be conducted to explore if existing equipment could be replaced by other including a TPM. Another study could be made where the measurement method would be changed from the TPM to another model supporting attestation.

Once hardware with needed features are acquired, simulating the integrity measurements with equipment in a proper rail simulator could be studied. Railway interlocking simulators exist and are a part of the software logic verification and testing procedures. If the TPM exist in some of the equipment this study could easily be made.

When the measurements are technically possible to do on a device in the rail environment next steps could be considered:

- Evaluating the effectiveness of static firmware measurements on interlocking devices software stack
- Communication method for value reporting
- Procedure for establishing trusted values in the rail industry
- Trust policy generation and mitigation procedures

The first study should evaluate the protection provided by the measurements. If the logic can be tampered with during runtime, additional tools are needed to protect the integrity.

Reporting the measurements requires a communication method. As shown in this study, reporting does not need additional security it is protected by the attestation key. If IP connectivity is provided, measurements can be reported to an attestation server.

Establishing trusted values is a larger task which expands back to the supply chain and most probably includes many different vendors. These are all responsible for different stages of production and software development. Establishing trust backwards to the supply chain will probably take time. A study could be made on considerations for a fully trusted supply chain. Another study could evaluate the options today, for establishing a trust state without considerations on the supply chain. From these two studies a third could be made to outline the steps from today to acquire a fully trusted software stack with values provided from vendors in the supply chain.

By being able to pinpoint the integrity measurement in a certain device and a certain layer of software, different mitigations could be established for different situations. This could span into different restriction levels depending on the failure. Traffic flow would not be restricted in a minor failure to the same extent as in a major one. This requires however an extensive amount of study and testing to be able to assure secure implementation.

REFERENCES

Adams, C., Lloyd, S. 2002. Understanding PKI: Concepts, Standards, and Deployment Considerations. Pearson Education: Boston.

Analysis of the Cyber Attack on the Ukrainian Power Grid. 2016. Electricity Information Sharing and Analysis Center. PDF document. Available at: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf [Accessed 18 May 2020].

Arthur, W. & Challener, D. 2015. A Practical Guide to TPM 2.0. Apress.

Assante, M. J. & Lee, R. M. 2015. The Industrial Control System Cyber Kill Chain. SANS Institute.

Azad, S. & Pathan, A-S. K. 2014. Practical Cryptography. Auerbach Publications.

Bratus, S., Matrosov, A. & Rodionov, R. 2019. Rootkits and Bootkits. San Francisco: No Strach Press Inc.

Brubaker, N., Caban, D., Glyer, C., Krotofil, M. & Scali, D. 2017. Attackers Deploy New ICS Attack Framework TRITON and Cause Operational Disruption to Critical Infrastructure. FireEye. Available: https://www.fireeye.com/blog/threatresearch/2017/12/attackers-deploy-new-ics-attack-frameworktriton.html [Accessed 19 May 2020].

Buurmans, K., Es, A., Koopmans, M., Vliet, M. & Rijlaarsdam, R. 2018. Feasibility Study Reference System ERTMS: Final Report, Digitalisation of CCS (Control Command and Signalling) and Migration to ERTMS. PDF document. European Railway Agency. Available at: https://www.era.europa.eu/sites/default/files/library/docs/studies/ccs_migration_study_arcadis_report_en.pdf [Accessed 18 May 2020].

Caracano, A., Dragoni, Y. & Pinto A. D. 2018. Triton: The First ICS Cyber Attack on Safety Instrument Systems. Nozomi Networks. WWW document. Available: https://i.blackhat.com/us-18/Wed-August-8/us-18-Carcano-TRITON-How-It-Disrupted-Safety-Systems-And-Changed-The-Threat-Landscape-Of-Industrial-Control-Systems-Forever-wp.pdf [Accessed 19 May 2020].

Ceruzzi, P. E., 2003. A History of Modern Computing. London:The MIT Press.

Cherepanov, A., 2017. WIN32/Industroyer: A new threat for industrial control systems. WWW document. ESET. Available at: https://www.welivesecurity.com/wp-content/uploads/2017/06/Win32_Industroyer.pdf [Accessed 18 May 2020].

Chien, E., Falliere, N. & Murchu, L. O. 2011. W32.Stuxnet Dossier. PDF document. Symantec. Updated September 2010. Available at:

https://www.wired.com/images_blogs/threatlevel/2010/10/w32_stuxnet_dossier.pdf [Accessed 18 May 2020].

Cimpanu, C. 2018. Security flaw in AMD’s secure chip-on-chip processor disclosed online. WWW document. Updated 6 January 2018. Available at: https://www.bleepingcomputer.com/news/security/security-flaw-in-amds-secure-chip-on-chip-processor-disclosed-online/ [Accessed 18 May 2020].

Das, A. & Madhavan, C. E. V. (2009). Public-key Cryptography: Theory and Practice. Pearson Education, South Asia. ISBN: 978-81-317-0832-3. New Delhi.

Docker. 2020. Documentation. WWW document. Available at: https://docs.docker.com/ [Accessed 18 May 2020].

EN 50126-5:en. 2014. Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 5: Functional Safety - Software

Fukuyama, F. (1995). Trust: The social virtues and the creation of prosperity. Free Press.

Goldman, K. 2016. IBM’s software tpm 2.0. Sourceforge. Blog. Updated 20 December 2019. Available at: https://sourceforge.net/projects/ibmswtpm2/ [Accessed 22 May 2020]

Greenberg, A. 2017. Unprecedented Malware Targets Industrial Safety Systems in the Middle East. WWW document. Updated 14 December 2017. Wired. Available: https://www.wired.com/story/triton-malware-targets-industrial-safety-systems-inthe-middle-east/ [Accessed 18 May 2020].

Harari, Y. N. 2014. Sapiens: A Brief History of Humankind. Signal.

Hidden, A. 1989. Investigation into the Clapham Junction Railway Accident. Department of transport. PDF document. Available at: http://www.railwaysarchive.co.uk/documents/DoT_Hidden001.pdf [Accessed 19.5.2020]

IEC 60870-5-101:en. 2020. Telecontrol equipment and systems - Part 5-101: Transmissionprotocols - Companion standard for basic telecontrol tasks.

IEC 60870-5-104:en. 2020. Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standardtransport profiles.

IEC 61850:en. 2020. Communication networks and systems for power utility automation.

TPMCourse. 2020. Github repository for TPMCourse. Nokia. Updated 18 May 2020. Available at: https://github.com/nokia/TPMCourse [Accessed 22 May 2020].

Kananen, J. 2017. Kehittämistutkimus interventiotutkimuksen muotona: Opas opinnäytetyön ja pro gradun kirjoittajalle. Publications of Jyväskylän ammattikorkeakoulu 2017:232. Jyväskylä: Suomen Yliopistopaino Oy.

Kantamaa, V-M. & Sorsimo, T. 2018. Rautatieturvalaitteet. 2nd edition. Keuruu: Otavan Kirjapaino Oy.

Langner, R. 2013. To Kill a Centrifuge: A technical analysis of what stuxnet's creators tried to achieve. The Langner Group. Available: https://www.langner.com/wp-content/uploads/2017/03/to-kill-a-centrifuge.pdf [Accessed 19 May 2020].

Lewis, D. J. & Weigert, A. 1985. Trust as a Social Reality. Social Forces 63 (4), 967–985.

Nason, R. 2017. It's not complicated: The Art and Science of Complexity for Business Success. Rotman-UTP Publishing.

OVMF FAQ. 2019. TianoCore. Available: https://github.com/tianocore/tianocore.github.io/wiki/OVMF-FAQ [Accessed 23 April 2020].

Platform Security Architecture Overview White Paper 2019. ARM. WWW document. Available at: https://pages.arm.com/PSA-Building-a-secure-IoT.html [Accessed 19 May 2020].

GlobalPlatform. 2017. Root of Trust Definitions and Requirements. WWW document. Available at: https://globalplatform.org/wp-content/uploads/2018/06/GP_RoT_Definitions_and_Requirements_v1.0.1_PublicRelease_CC.pdf [Accessed 19 May 2020].

Roseveare, H. 1991. The Financial Revolution 1660-1760. New York: Routledge.

Ruan, X. 2014. Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine. Apress.

Schenker, G. N. 2020. Learn docker – fundamentals of docker 19.x. E-book. 2nd edition. Packt Publishing.

Soanes, C. & Stevenson, A. 2005. The Oxford dictionary of English. 2nd edition. Oxford University Press.

Stumpp, K. 2019. Draft of the Security-by-Design and of Railway Cyber Security Management System Standards. European Union Funding for Research and Innovation.

TCG PC Client Platform Firmware Profile. 2019. Version Level 00 Revision 1.04. Family 2.0. Trusted Computing Group.

TCG PC Client Platform TPM Profile Specification for TPM 2.0. 2020. Version 1.04 Revision 37. Updated 3 Feb 2020. Trusted Computing Group.

TCG TPM v2.0 Provisioning Guidance. 2017. Version 1.0. Revision 1.0. Trusted Computing Group.

TCG. 2019. TPM 2.0: A brief introduction. PDF document. Available at: https://trustedcomputinggroup.org/wp-content/uploads/2019_TCG_TPM2_BriefOverview_DR02web.pdf [Accessed 22 May 2020].

TPM recommendations. 2018. Documentation. Microsoft. WWW document. Available at: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/tpm-recommendations [Accessed 22 May 2020].

Trusted Platform Module Library, Part 1: Architecture. 2016. Version Level 00 Revision 01.38. Family 2.0. Trusted Computing Group.

Trusted Platform Module Library, Part 2: Structures. 2016. Version Level 00 Revision 01.38. Family 2.0. Trusted Computing Group.

Trusted Platform Module Library: Part 1: Architecture. 2019. Revision 01.59. Trusted Computing Group.

Vacca, J. R. 2004. Public Key Infrastructure: Building Trusted Applications and Web Services. Boca Raton: CRC Press.

Zetter, K. 2019. Hackers Hijacked ASUS Software Updates to Install Backdoors on Thousands of Computers. Vice. Available at: https://www.vice.com/en_us/article/pan9wn/hackers-hijacked-asus-softwareupdates-to-install-backdoors-on-thousands-of-computers [Accessed 12 May 2020].