Fachbereich Informatik
Fachgebiet Sicherheit in der Informationstechnik
Fraunhofer Institut für Sichere Informationstechnologie SIT
Prof. Dr. Claudia Eckert

Diplomarbeit

Securing Digital Evidence

Jennifer Richter

Betreuer: Prof. Dr. Claudia Eckert
Dipl. Inform. Nicolai Kuntze

30.04.2009

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, April 2009

(Jennifer Richter)

Kurzfassung

Trusted Computing wie es durch die Trusted Computing Group definiert wird, hat zum Ziel die Sicherheit von Plattformen zu gewährleisten. Insbesondere bietet die Möglichkeit sich gegenseitig als vertrauenswürdig zu authentisieren erhöhte Sicherheit gegenüber Dritten. Die Funktionalität dazu liefert ein Hardwarebaustein, das Trusted Platform Module (TPM). Die Funktionen, die das TPM bereitstellt, werden in dieser Arbeit verwendet, um vertrauenswürdige digitale Beweise in einem automatisierten Prozess zu sammeln. Digitale Beweise wie Messwerte, Fotos oder Dokumente werden allerdings nichts von vornherein bei Gericht als glaubwürdig angesehen. Sie sind einfach zu fälschen oder zu modifizieren. Da sie nur in binärer Repräsentation existieren, ist es schwierig vorgenommene Änderungen festzustellen und diese zu beweisen. Verfahrensfehler, beispielsweise aufgrund fehlerhafter Beweise, sind schwer zu handhaben. Weshalb von vorne herein versucht werden muss diese auszuschließen. Dies und die leichte Fälschbarkeit von digitalen Beweisen führt zur Geltendmachung von strengen Sicherheitsrichtlinien. Deshalb fordern Gerichte, die heutzutage immer häufiger mit digitalen Beweisen konfrontiert werden, stichhaltige Beweise für die Authentizität und Integrität der vorgelegen digitalen Beweismittel. Eine eindeutige Identifizierung einer Person oder eines Objektes, welche eine Aktion ausgeführt haben, muss gewährleistet sein. Weiterhin muss sichergestellt werden, dass die gemessenen Daten ab dem Zeitpunkt ihrer Erzeugung nicht mehr verändert oder gefälscht werden können. Da insbesondere die Handhabung von digitalen Beweisen viele Probleme verursachen kann, werden geeignete Methoden benötigt, die sichere Informationen beschaffen, aufzeichnen und aufbewahren können. Weiterhin benötigt man geeignete Methoden um digitale Beweise adäquat vor Gericht zu präsentieren.

Diese Arbeit hat zum Ziel ein Konzept einer sicheren Umgebung zu erstellen, die in der Lage ist automatisch Information zu sammeln und so zu sichern, dass sie als zulässige digitale Beweise verwendet werden können. Um die Akzeptanz der gemessenen Daten vor Gericht zu erreichen, wird mit Hilfe des TPM die Integrität des Systems überprüft und dokumentiert, sowie Sicherheitsinformationen erzeugt und den gesammelten Daten beigefügt. Sicherheitsinformation sind z.B. Hashwerte, Signaturen, Systemzustandsinformationen und zurückverfolgbare, zertifizierte Zeitstempel. Ein wichtiger Punkt hierbei ist die Entwicklung einer Methode, um die Daten mit einem systemunabhängigen, zertifizierten Zeitstempel auszustatten. Desweiteren werden die Authentizität und die Integrität der Daten durch die Erzeugung von Signaturen und Hashwerten gesichert. Diese und weitere Trusted Computing Maßnahmen zusammen bilden akzeptierbare zulässige digitale Beweise. Bei entsprechender Auswertung der gesammelten Daten, zusammen mit den zugefügten Sicherheitsinformationen, beweisen diese, dass ein Ereignis zu einer bestimmten Zeit an einem bestimmten Ort stattgefunden hat. Die Hauptfunktionen des entwickelten Konzepts wurden in einer Referenzimplementierung verwirklicht. Die Implementierung zusammen mit dem Konzept zeigt die Realisierbarkeit der automatischen Beschaffung von digitalen Beweisen.

Abstract

Trusted Computing as defined by the Trusted Computing Group is aimed to provide a trusted platform that can attest to its current configuration to a third party. The functionality behind is a Trusted Platform Module (TPM). We use this functionality to provide trustworthy digital evidence collected in an automated process. Digital evidence like measurements, photos, or documents is not a priori seen as trustworthy since it is only existent in binary representation. Therefore, modifications and tampering of the collected data is easy but also hard to detect. Faults in trial are difficult to cope with and so lead to an enforcement of high security policies. Since courts are now dealing more and more with evidence in digital form they require for integrity and authenticity of digital evidence. The unambiguous identification of a person or entity performing an action must be warranted. Moreover, it should not be possible to tamper data at any time since its creation. Because handling of digital evidence can cause many problems it requires appropriate methods to securely acquire information and present them in court.

In this thesis, a concept of a protected environment that automatically collects and secures digital evidence is created. In order to achieve the admissibility of digital evidence in a court of law, the integrity of the system is checked and document and the collected data is furnished with additional security information. Security information comprising for example hash values, signatures, configuration status values and retraceable certified time stamps. In particular, a method for the acquisition of a certified platform independent time stamps is developed. Moreover, authenticity and integrity of the collected data is accomplished through the creation of hash values and signatures. These and additional Trusted Computing methods constitute admissible digital evidence. During a subsequent verification the collected data together with the security data prove that a certain event happened at the specified time and location. The main functions of the developed Trusted Computing concept are implemented in a demonstrator application. Both, the implementation and the concept show the feasibility of an automatic collection of admissible digital evidence.

Contents

1 Introduction **13**

2 Digital Evidence **17**

3 Trusted Computing **19**
3.1 Introduction . . . 19
3.2 The Trusted Platform . . . 19
3.2.1 Identity and Ownership . . . 20
3.2.2 Roots of Trust . . . 20
3.2.3 Trusted Boot . . . 21
3.3 TPM . . . 22
3.3.1 Functionalities . . . 22
3.3.2 Architecture . . . 23
3.3.3 Message exchange . . . 24
3.4 Cryptographic Components of a TPM . . . 25
3.4.1 Keys . . . 26
3.4.2 Certificates . . . 28
3.4.3 Attestation . . . 29
3.5 TPM v1.2 Specification Changes . . . 30
3.6 Attacks on a TPM . . . 32
3.6.1 TPM Reset Attack . . . 32
3.6.2 Software Attack . . . 33
3.6.3 Platform Reset Attack . . . 34
3.6.4 Remote Attestation Attack . . . 34
3.6.5 Other Vulnerabilities . . . 35

4 Traffic Monitoring Systems **37**
4.1 Use Case . . . 37
4.2 Threats and Requirements . . . 41
4.2.1 Threats . . . 41
4.2.2 Requirements . . . 48
4.3 Definition of Trust . . . 57
4.4 State of the Art . . . 58
4.4.1 Relevant Laws for this matter . . . 59
4.4.2 Relevant Standards . . . 62
4.4.3 Trusted Computing Group . . . 63
4.5 Concept . . . 63
4.6 Solutions . . . 71
4.6.1 Trusted Boot . . . 72
4.6.2 Measurement Records . . . 79
4.6.3 Time Stamping . . . 92
4.6.4 Transmission . . . 104
4.6.5 Miscellaneous . . . 106

5 Implementation and Security Analysis **109**
5.1 Implementation 110
5.1.1 Used Components 110
5.1.2 Classes 112
5.2 Security Analysis 126

6 Conclusion **129**

7 List of Abbreviations **131**

A XML Schema **141**

B XML Output Example **147**

List of Figures

1 Chain of Trust [70] . . . 22
2 TPM Architecture . . . 23
3 Volatile and Non-volatile TPM Storage . . . 27
4 Attestation with Privacy Certification Authority . . . 30
5 Direct Anonymous Attestation . . . 31
6 Roles . . . 39
7 Roles and their Tasks . . . 41
8 Transitive Trust Model . . . 58
9 Concept . . . 65
10 Concept with added Security Goals . . . 67
11 Chain of Trust [64] . . . 73
12 TPM_QUOTE_INFO structure [87, page 99] . . . 84
13 TSS_PCR_EVENT structure [86, page 98] . . . 85
14 Signing Process . . . 87
15 TPM_KEY structure [87, page 88] . . . 88
16 TPM_CERTIFY_INFO structure [87, page 96] . . . 90
17 TPM_CURRENT_TICKS structure [87, page 117] . . . 94
18 General Time Stamping Procedure . . . 97
19 TPM Time Stamping . . . 97
20 Association of Real Time to TPM Time Stamping . . . 101
21 Remote Attestation with PCA . . . 105
22 Implemented Components . . . 109
23 TSS Layers . . . 110
24 TA Protocol . . . 117
25 TA Protocol with Remote Attestation . . . 120

List of Tables

1 PCR usage of static root of trust for measurement as defined through the TCG [76] . . . 74
2 PCR usage of dynamic root of trust for measurement as defined through the TCG [76] . . . 75
3 PCR usage of TrustedGRUB [76] . . . 77
4 PCR usage of GRUB-IMA [75] . . . 78
5 Event types of GRUB-IMA [75] . . . 78
6 TPM_QUOTE_INFO Data Structure [87, page 99] . . . 85
7 TSS_PCR_EVENT Data Structure [86, page 98] . . . 86
8 TPM_KEY Data Structure [87, page 88] . . . 89
9 TPM_CERTIFY_INFO Data Structure [87, page 96] . . . 91
10 TPM_CURRENT_TICKS Data Structure [87, page 117] . . . 95
11 MeasuringInstrument Parameters . . . 114
12 TAServer Parameters . . . 116
13 TAClient Parameters . . . 116
14 TAServer_RA Parameters . . . 119
15 TAClient_RA Parameters . . . 119
16 MyXMLBuilder Parameters . . . 125

1 Introduction

The penetration of computers or other electronics into daily life increasingly influenced laws and jurisdiction the last years [62]. Evidence, especially paper-based evidence, like that used in courts is no longer the only kind of evidence used. Through the development of computer based technology, the importance of digital evidence like measured values, photos or videos has increased. However, digital evidence is regarded sceptically since it only exists in a binary representation. The collection, the handling, the storage and the presentation of digital evidence has met some problems in the past. For those reasons, the courts now demand stringent requirements on the admissibility of digital evidence at trial. As a result, digital evidence requires appropriate methods to prove and preserve their significance.

This thesis designs an embedded system that is able to collect admissible digital evidence through an automated process. The measurement process is mainly performed by a machine and human collection is kept to a minimum to ensure automation. For digital evidence to be admissible in a court of law, the system must be secure and the data produced must have integrity and authenticity. This process must be done automatically since the system will not have any human inputs. Consequently, a shielded environment must be created to enforce the high restrictions the courts place on admissibility. This environment must have security functions that increase the protection of the collected digital evidence. Moreover, additional security information that attests to the trustworthiness of digital evidence is produced to provide for admissible digital evidence. The verification of this security information must be done by a fair-minded third party and later be reproducible at court.

The motivation behind this project is to combine the automated collection of evidence with a practical application. One issue discussed in recent years is the introduction of various kinds of Traffic Monitoring Systems (TMS). In almost every European country, charging vehicles for exceeding the speed limit, or using highways is considered an acceptable practice. Traditional charging methods like badges and taxes are not appropriate anymore since the traffic load has increased substantially. The traffic load and the amplified environmental pollution require new charging schemes. In England, congestion charging schemes were developed to decrease the amount of cars driving through the inner areas of many cities, most notably the city of London [38, 82]. In Germany, satellite-based and distance oriented truck tolls were discussed and implemented in 2005 [24]. Additionally, the Netherlands is currently discussing the introduction of a road fee for all public roads. In December 2007, the Dutch government decided to introduce a national road toll. In July 2008, parliament voted this toll into law [51]. Apart from congestion charging, monitoring the speed limit is also a big issue. Speeding accidents are increasing but traditional techniques at reducing them do not seem to discourage drivers. Drivers, knowing the location of fixed speed measuring instruments, slow down in this area but accelerate right after they pass the instrument. Therefore, new techniques must be found to control speeding. One solution might be the controversial speed measuring with distance control [57]. The distance control tech-

nique documents each vehicle passing a special section and calculates the average speed out of the time needed for this distance. This new method needs new data processing techniques to handle the arising amount of data. Particularly, the privacy of personal data as well as the integrity and the authenticity of such devices are a concern. Moreover, Blythe [38] said 'the lack of appropriate technology will not be the constraint in implementing road-use charging in the near future' [38, page 356].

In recent years, software based techniques have proven to be vulnerable to attacks and failures threatening the security of the produced measurements. Software solutions are more easily modified than their hardware equivalents. This can be advantageous in terms of easy development and maintenance but it can also be disadvantageous because of their ability to be compromised. In contrast, hardware based security solutions are considered to be more secure since they are not as error-prone. For example, code burnt onto a hardware chip cannot be modified to malfunction or leak critical data. Additionally, software is usually weak in providing appropriate storage capabilities for keys. Hardware solutions burn keys into the hardware or store them in non-volatile storage made inaccessible with other hardware means. Both methods make it difficult to access the keys protected by hardware. Traffic Monitoring Systems are pretty well-defined in their purpose. There is no need for dynamic changes of their system functions. Therefore, the software can be kept to a minimum of instructions. On the other hand, the hardware can be enlarged to a maximum to provide protected capabilities for safety-critical data. Hence, the device can use the advantages of a hardware solution while only exposed to a minimum of software vulnerabilities.

The purpose of this thesis is to secure digital evidence so that it is admissible in court. Presuming that software solutions are not producing admissible digital evidence on their own a mainly hardware based solution for a TMS is contemplated. Therefore, the technical hardware basis offered by Trusted Computing (TC) is used to realise the concept for the creation of digital evidence through an automated process dealing with the problems of the integrity of digital evidence. Trusted Computing as defined by the TCG provides us with hardware implementing several cryptographic modules and other advanced security features. For example, this hardware component is able to attest its current state to a third party and store keys for a longer period of time. This component is a so-called Trusted Platform Module (TPM). The TPM offers all required features and on that account is an ideal hardware security basis. For this reason, the TPM is used to create the desired embedded system approaching the challenge of integrity protection of digital evidence. The goal is to offer a new hardware based technique anchored in Trusted Computing which is able to replace traditional techniques.

Chapter 2 gives an introduction to digital evidence and the related problems. Followed by Chapter 3 which describes the basic concepts of Trusted Computing in terms of the TCG. Chapter 4 forms the main part of this thesis. Here, a Traffic Monitoring System design is presented using TC concepts. A detailed use case is described as well as, the intimidations threatening the design. In the end, the concept and its TC solutions are depicted. Next, Chapter 5 covers the implementation part of this thesis.

A demonstration application realising the developed concepts is implemented. Then, a security analysis of the presented concept is presented reconsidering occurring weak points. Eventually, the idea is reviewed and concluded with Chapter 6.

2 Digital Evidence

This chapter describes digital evidence and its admissibility in legal actions. There exist a variety of articles, books, guides, laws etc. dealing with digital evidence, its seizure and computer forensics in general (e.g. [49,62,93,94], and much more). Because of the range of this topic, in particular digital evidence, it is not possible to consider all peculiarities and requirements but only give an introduction to this topic.

Digital evidence is defined as any probative information transmitted and stored in digital form that can be demonstrated in court. Digital evidence can be measurements, photos, e-mails, or other electronic documents. In recent years, courts have allowed the use of digital information as piece of evidence in trial. The use of digital evidence is increasing as e-mails, digital signatures, digital photographs, instant message histories, the contents of computer memory, log files, digital video or audio are allowed to be used in court as key pieces of evidence [62, 65]. An historical perspective of digital evidence is given in [99]. Whitcomb [99] explained the development of digital evidence and listed some standards concerning digital evidence and its seizure. In addition, on June 30, 2000, the U.S. Congress enacted the Electronic Signatures in Global and National Commerce Act. This federal law facilitates the use of electronic records and signatures in commerce by assuring the validity and legal effect of electronic contracts. The first section (101.a) denotes the general intention that 'a signature, contract, or other record relating to such transaction may not be denied legal effect, validity, or enforceability solely because it is in electronic form' [19]. Electronic signatures are now considered as valid as paper documents. This act is a result of the arising need to use digital evidence in legal action and it necessitates for restrain integrity and authenticity of electronic data. Electronic data as paper-based information will become inadmissible as evidence if not complying with several requirements.

Evidence must meet certain legal requirements before being produced in court. Braid [39] defined five rules of evidence in order for evidence to be considered useful. Evidence must be admissible, authentic, complete, reliable and believable. Whereas, admissible means that the evidence must conform to legal rules. Authentic implies that an incident must be able to be relevant to the evidence. Moreover, Braid [39] said that evidence must be complete. This means that the whole story must be told, not just one special perspective of the case. It is necessary to consider and evaluate all evidence available. Evidence that tries to prove a crime as well as evidence that tries to disprove it must be collected. The first, evidence that goes to prove a crime, is known as inculpatory evidence. While the latter, evidence that goes to disprove a crime, is known as exculpatory evidence. Exculpatory evidence is collected to eliminate other suspects and to strengthen the inculpatory evidence. Additionally, evidence needs to be reliable. Nothing about how the evidence was collected or handled shall cast doubt on its authenticity or veracity. Finally, evidence must be readily believable and understandable to members of a jury. A jury not understanding the supplied evidence will not incorporate them in their contemplations.

In comparison to more established and traditional evidence, courts have noted that digital evidence can cause some problems. Digital evidence tends to be more difficult to destroy, potentially more expressive and possibly more readily available. Moreover, digital evidence can be more voluminous, easily duplicated, modified and forged. Another problem with digital evidence is its volatility. Accidental handling can alter digital evidence as well as intended attacks on the data. Specifically, digital evidence can be easily altered undetected compared to traditional evidence. OS-dictated overwriting and recycling, mishandling during acquisition process and unintentional damage are also problems courts have to deal with when they use digital evidence. In consequence, it is important to secure the integrity of data used as digital evidence. Usually, integrity of digital evidence associates events with time and thus methods are required that bind time and identity [71]. For that reason, methods used to assemble and protect digital evidence themselves must be trusted.

There exist a range of methods to collect digital evidence and to protect it from faulty handling. Nowadays, electronic systems that are able to automatically collect information are used to acquire evidence about an offence or crime. That is the reason why it becomes most important that those systems run correctly even though they are not under permanent observation. The correct gathering of information is part of a machine. Only, evaluating information, proving illegal behaviour and presenting the evidence precisely to a court of law will be the responsibility of a person. Consequently, when collecting evidence no alteration, virus introduction, damage or data corruption must be prevented. As this could cast doubt on the assembled evidence. The integrity, authenticity and security of probative information should always be granted. Moreover, it must be considered that each country has its own legislation. Consequently, digital evidence is handled differently around the world. Different countries have different laws. Therefore, restrictions on how digital evidence can be collected and used for court proceedings vary from each country. The collection of evidence can be significantly complicated by laws. For example, in [65] laws that affect the monitoring and collection of digital information especially in the U.S. are lengthy explained.

Mason et al. [62] gave a good introduction to digital evidence. They sorted digital evidence in a number of categories. One of these categories is presented by 'records generated by a computer that have not had any input from a human' [62, page xiii]. These records chiefly match with systems automatically collecting evidence. As Mason et al. pointed out, the 'main evidentiary issue with such records may be to demonstrate that the computer program that generated the record was functioning properly at the material time.' [62, page xiii] Therefore, records generated automatically with no human interaction must prove that the system was functioning correctly at the recorded time. According to Patzakis [68] the digital chain of custody must always be warranted. 'Once compromised, either during the collection or analysis process, the evidentiary integrity of the data is lost.' [68, page 1]. Considering the whole lot brings us to the following chapters.

3 Trusted Computing

Trusted Computing establishes a basis for this work. Therefore, the main topic of this chapter will be Trusted Computing. Subsequently, an excerpt of all necessary Trusted Computing information will be given. This chapter will first give an introduction to the organisations involved in the creation of Trusted Computing specifications. Then, the principals of Trusted Computing and the basic features of a TPM platform are described. Moreover, the TPM v1.2 specification changes are listed. And last, some of the known attacks on TPMs are explained.

3.1 Introduction

Traditional hardware components do not provide adequate security concepts. For trustworthy security solutions a combination of secure software and hardware solutions is needed. Especially because software security cannot be granted constantly, the hardware part of the architecture needs to be secured. The Trusted Computing Initiative was founded to develop methods for a secure attestation of the platform. Together with hardware and software manufacturers the Trusted Computing Initiative develops a security architecture which tries to meet the security requirements and attests the trustworthiness of a platform. To establish trust into a system there needs to be a component which forms a basis of trust. Software is constantly exposed to threads arising from security vulnerabilities such as malware. On the contrary, hardware components are not threatened constantly because physical access is needed to harm them. Because software has proven not to be reliable enough to build a trust basis a hardware component is needed. Therefore, a hardware chip - the Trusted Platform Module (TPM) - was designed by the Trusted Computing Group (TCG).

The TCG is the official subsequent organisation of the Trusted Computing Platform Alliance (TCPA). First, in 1999 the TCPA was founded. This group consisted of members of Microsoft, Intel, IBM, Compac and HP. They invited more companies to join their project. In 2002 more than 150 companies participated in the development of specifications and made them an open industry standard. But the TCPA turned out to be incapable of any action because of the growing number of members with right to veto. In consequence, the Trusted Computing Group (TCG) was formed in April 2003. The TCG adopted the TCPA specifications and their further development. The TCG turned their main attention on the development of the Trusted Platform Module (TPM). Specification upgrading and usage of the TPM in different environments is only a small part of their whole field of activity. [52, 53, 64]

3.2 The Trusted Platform

A trusted platform as defined by the TCG must at least offer protected capabilities, integrity measurement and integrity reporting. To realise these three features a trusted platform is equipped with a unique identity and other security functions. A perfect

basis for the features realised by a trusted platform like a TPM is provided through the unique identity of the platform, the Roots of Trust and the trusted boot process.

3.2.1 Identity and Ownership

The Trusted Platform Module provides the owner of the TPM with a **unique identity**. This identity is obtained by integrating information about the chip inside the TPM. Information included are the specially created Endorsement Key (EK) and the associated Endorsement Key Credential. Through that information it is possible to prove the identity and the correctness of the manufactured chip to a third party. Nevertheless, using a unique identity can lead to privacy problems. Therefore, the TPM implements a possibility to camouflage the identity through the usage of Attestation Identity Keys (AIK). Furthermore, the TPM specification 1.2 provides the opportunity to change the initial identity information. Even though, this is not implemented in the architecture yet.

The TPM is a passive chip which only carries out activities if the owner of the platform explicitly activates it. In general, the TPM is deactivated per default. Only after activation of the TPM the security features are available for usage. To prevent others from activating the TPM via remote access the owner must explicitly activate the TPM with the *Take_Ownership command*. After the activation, the owner is able to use and manage the TPM functions. For special functions that need owner authorisation the authentication password needs to be proven. Thus, platform usage is fully controlled by the user.

3.2.2 Roots of Trust

The TPM consists of three 'roots of trust' namely, the Root of Trust for Measurement (RTM), the Root of Trust for Storage (RTS) and the Root of Trust for Reporting (RTR). Those roots of trust establish a basis for trusting the whole system. It is necessary to rely on these roots of trust so that later a chain of trust can be developed. The enlarged chain of trust permits the user to trust the whole system. To enhance security of the roots of trust, all functions of these computing engines are provided by the TPM.

The **Root of Trust for Measurements** is a computing engine which is responsible for determining the integrity of a configuration during the boot process. Typically, the computations are performed by the Core Root of Trust for Measurement (CRTM). During each restart the CRTM will be executed first. As mentioned above, the TPM provides functions of the RTM like the calculation of hash values (SHA-1) or the secure storage for calculated integrity values. Secure storage of integrity values is realised through special registers, the Platform Configuration Registers (PCRs). The **Root of Trust for Storage** protects keys and data which are trusted by the TPM. In addition, this engine is responsible for the whole key management. Meaning the creation and the maintenance of cryptographic keys. The RTS realises a special key hierarchy to securely store cryptographic keys. Consequently, all safety-critical data is protected. The key

hierarchies root is the Storage Root Key (SRK) which never leaves the TPM. Further information about the SRK will be given later. The **Root of Trust for Reporting** reliably issues information about the integrity of data protected by the RTS. Its functions are based on the attestation principles which will be discussed later in this chapter.

The Roots of Trust may not only have shielded locations or protected capabilities. The parts which are not protected but trusted are known as the Trusted Building Blocks (TBB). At this point, 'trusted' means that these parts behave in an expected way and do not compromise the trusted platform goals [89].

3.2.3 Trusted Boot

The TCG concepts of secure booting imply to defend a system against attacks and to prevent a system from loading a mistrusted system configuration. In consequence, the system always has to be booted from an unchanged basis. Because only the state of the system is documented the implemented method is on no account a secure boot process but rather a **trusted boot**. The integrity reports documented must be evaluated separately. How evaluation is done in detail is not further considered in the TCG specification. The integrity measurement reports are stored in a protected volatile storage of the TPM, also known as PCRs. All further integrity measurements are concatenated with an older PCR value, hashed and stored: $PCR[n] \leftarrow SHA{-}1(PCR[n]{+}measureddata)$. The process of acquiring metrics of platform characteristics that affect the integrity and the trustworthiness of a platform and saving digests of those metrics in PCRs is known as **integrity measurement** [89].

Transitive trust plays an important role in determining if a system can be trusted or not. Transitive trust is relevant in a TPMs trusted boot process. Transitive trust is based upon one trusted root which is always been trusted without limitations. Then this root ascertains the trust of the next trustworthy component. After the root of trust assured that the first loaded component was reliably and correctly measured a transitive trust relationship can be established between those two components. The first component then hands control over to the second component that in turn measures the next component. Passing this kind of integrity check control is handed over. Then, this component starts measuring a subsequent component and so on. Each component must be trusted to the effect that it measures the subsequent component correctly. This sequence of integrity checks is also known as **chain of trust**. Finally, the environment has to trust the system that the trusted boot process was executed correctly and that the chain of trust was established properly.

Reid and Caelli [70] described how the chain of trust is established using a TPM. The root is the CRTM which is either part of the TPM or the whole BIOS. In general, out of flexibility reasons the CRTM is not included in the BIOS. The CRTM is in a predefined state. First, the CRTM will be loaded into the storage and executed. Then, the CRTM hashes the BIOS code and stores the hash value in a PCR. The CRTM now passes control to the BIOS - the chain of trust is expanded. Then, the BIOS hashes

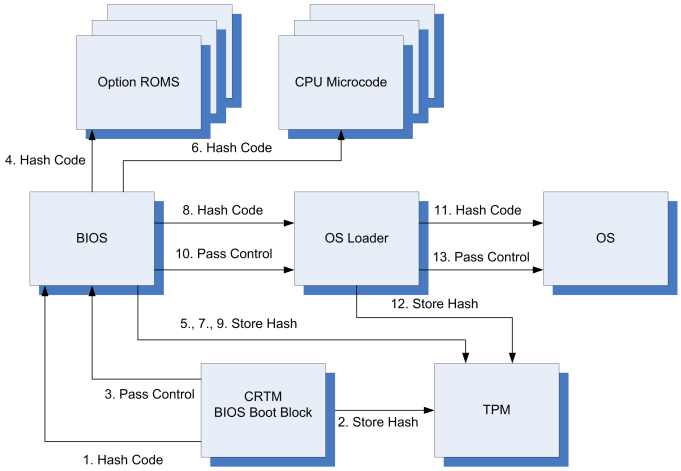

Figure 1: Chain of Trust [70]

and stores measurements of Option ROMS, CPU microcode updates and Operating System (OS) Loader. Control then is given to the OS Loader. The OS Loader hashes the Operating System and finally, passes control to it. The whole process is depicted in Figure 1.

3.3 TPM

Having heard already a lot about the Trusted Platform Module (TPM) and its trust basis, it can now be described in more detail. A Trusted Platform Module (TPM) is a chip which covers the hardware part of the Trusted Computing Architecture. Since October 2002 a TPM protection profile is published after the Common Criteria. This design determines TPM requirements with trust level EAL4+ [44]. (The Evaluation Assurance Level (EAL) define precise requirement for IT security tests. A growing EAL number increases the needed requirements for the inspected scope, the test depth and the testing methods [3, 42, 43].) As a result, it can be seen as providing a secure hardware environment for a diversity of cryptographic functions. A TPM is a passive chip which only reacts on commands and does not carry out any actions it is not defined to. During the start up the BIOS starts the TPM which then executes a self test. This self test secures that all functions work correctly.

3.3.1 Functionalities

In general, the functions of a TPM are the following:

- Offering an Endorsement Key for its identification.
- Generation of cryptography keys, symmetric or asymmetric.
- Creation of signatures, calculation of hash values and asymmetric encryption.

- Encryption of cryptographic keys (binding).
- Secure storage of small objects (e.g. keys) and hash values.
- Signed reporting about calculated hash values.
- Owner functions to seize the platform and to (de)activate the TPM.
- Offering a trustworthy timer.

Nevertheless, the TPM is kept simple meaning that not all probably useful capabilities are realised. This is done out of security reasons, less components needed to control and needed to protect against unexpected behaviour mean less security risks.

3.3.2 Architecture

The TPM is built out of a variety of components each of them carrying out some of the previously defined functions. The internal architecture of the TPM is illustrated in Figure 2 showing its components. Subsequently, these components and their functions are briefly explained (cf. [89]).

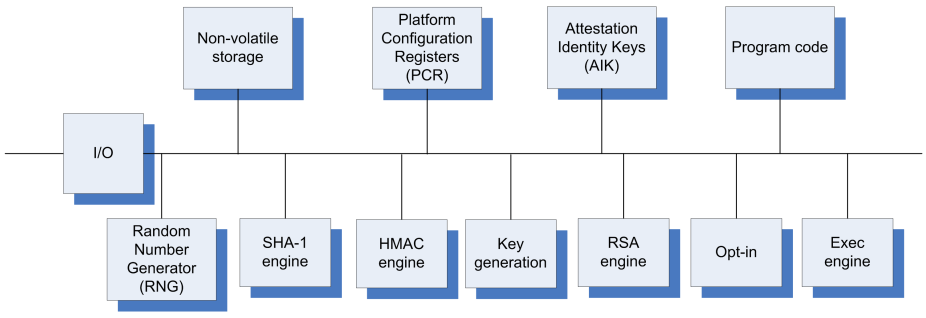

Figure 2: TPM Architecture

All communication between the TPM components takes place through the *I/O component*. Therefore, this component routes the messages to the correct components while enforcing protocol and access policies. Specifically, access policies related to the Opt-In component must be put into effect. Whereas, the *Opt-In* component implements customer desires whilst minding TCG policies. TPM components utilised for the measurement of system components are the execution engine and the program code component. The *execution engine* runs program code and performs the initialisation of the TPM and the measurements which are stored in the Platform Configuration Registers. And the *program code* component holds firmware needed for measuring platform devices. It is reasonable that this part is represented by the CRTM but in some cases it might be necessary to locate the CRTM outside the TPM.

Moreover, the TPM owns a variety of cryptographic components. The *Random Number Generator* is one of this components. It is used to provide randomness of data to the TPM. The produced random numbers are used inside the TPM to generate cryptographic keys or outside the TPM for example as input for software random number generators. Another cryptographic component is the *SHA-1 engine* which implements a cryptographic hash function. This hash function is in particular used for the updating process of PCR values. The returned value is a 160 Bit value. The *HMAC engine* creates Message Authentication Codes (MACs) which can be used to verify the integrity of messages. A MAC is based of a cryptographic hash function (SHA-1) and a symmetric key. A TPM especially uses the HMAC functionality for establishing a secure communication between its components. Additionally, the TPM possesses components used for symmetric and asymmetric key utilization. These components include an engine for the symmetric and asymmetric *key generation*. Symmetric encoding can only be deployed by TPM internal operations so that critical data like keys or passwords are secured. The *RSA engine* implements the RSA algorithm which is used for the creation of RSA signatures, encryption, decryption etc. A key length of 512, 768, 1024 and 2048 Bit is supported. Out of performance reasons a validation mechanism for RSA signatures cannot be included. *Attestation Identity Keys*, another part of the TPM and a special kind of keys applied for anonymisation, are thoroughly explained later in sections 3.4.1 and 3.4.3.To store safety-critical data like the EK or other keys the *non-volatile storage* is used. Details about the non-volatile storage are described later in 3.4.1.

Another important part of the TPM are *Platform Configuration Registers (PCRs)*. PCRs pertain to the volatile storage and are used to securely store 160 Bit hash values. The integrity of the system is determined through those hash values. Among other values hash values are computed during the trusted boot process or during the execution of security services. Every TPM owns at least 16 PCRs. No faulty integrity values shall be stored in already occupied registers. To correctly store the hash values inside the PCRs a special method is applied. This method concatenates all update values with the old hash values and then a new hash value is computed [89]:

$$PCR[n] \leftarrow SHA-1(PCR[n] + measureddata)$$

This enables the TPM to store an almost infinite number of hash values. For further security, the order of the measured integrity values is stored in a log file. This log file is known as the Stored Measurement Log (SML). Each sequence shares a common measurement digest. Measured values are attached to the common measurement digest and re-hashed. Extending the digest ensures that measured values are not ignored and that the order of operations is preserved.

3.3.3 Message exchange

Secure distributed systems while exchanging information need to identify the endpoints of communication. For this, it is equally important to know that communication takes place between the right communication partners as well as to know the state of the

system. A TPM improves security in combining key management features and configuration management features. The TCG defines four classes for protected message exchange:

- Binding
- Signing
- Sealed-Binding or Sealing
- Sealed-Signing

Binding is the operation of encrypting a message using a public key. Assumed, that a traditional asymmetric algorithm is used and the non-migratable private key is managed by the TPM then only the TPM is able to use the private key. In consequence, a message encrypted with a public key relating to a non-migratable private key is bound to this certain TPM.

Signing is the traditional operation of linking a message's integrity to a key used to generate the signature. A TPM specially generates signing keys that can only be used for signing purposes and thus cannot be misused as for example as encryption keys.

Sealing is a binding operation restricted to a special platform configuration. That is, a message can only be encrypted if a specific platform configuration state is achieved. Therefore, sealing associates an encrypted message with a non-migratable key and a set of PCR values. In choosing PCR values the sender defines the platform in functioning in an intrinsic and known configuration. The selected PCR values as well as the symmetric key used to encrypt the message are asymmetrically encrypted. Henceforth, the TPM can only decrypt the symmetric key if the state of the platform is consistent with the PCR values defined by the sender.

Sealed-Signing associates PCR registers to a signing operation similar to sealing. It provides assurance that the platform used to sign a message was in a specific and trusted state. The signer must include some PCR values in the message defined by the verifier. Then, the verifier is able to check whether the platform was in the right configuration state at the time of the signature or not.

3.4 Cryptographic Components of a TPM

Secure key and certificate management and the principle of attesting the TPMs integrity are essential features constituting the security of a TPM. Keys as well as certificates must be handled correctly by their users to avoid any compromise. Nevertheless, the desired treatment is often neglected since it is seen as complicated and not recognised as important by their users. Out of these reasons the TPM provides protected creation, handling and storage capabilities for keys and certificates. Moreover, the attestation

feature helps to protect against fraudulent systems trying to communicate with other systems. Communication with a compromised system can be prevented in first instance. Subsequently, the cryptographic TPM components are thoroughly described.

3.4.1 Keys

Keys used or created by a TPM are managed differently depending on their designated usage. A TPM consists of a volatile key storage and a non-volatile key storage. The non-volatile key storage is regarded as secure and protected. Three safety-critical keys are stored in the non-volatile storage:

- **Endorsement Key (EK)**: This key is a 2048-Bit RSA-key pair and its main purpose is to confirm others that the TPM is authentic. The EK is generated by the manufacturer during the fabrication of the chip. Moreover, the manufacturer issues a signed Endorsement Key Credential to attest the authenticity of the EK. The private portion of the EK should never leave the TPM. Otherwise the EK would not be clearly dedicated to the TPM. Another important use of the public portion of the EK is to sign Attestation Identity Keys (AIKs) so that they are bound to one specific TPM. The public portion of the EK is only published to those Privacy Certification Authorities (PCAs) which are classified as trustworthy by the owner.

- **Owner Authentication Key**: This key is a 160-Bit SHA-1 hash value of a password which is defined by the TPM owner. This password is the authentication password of the owner and is necessary for execution of safety-critical functions. This hash value is encrypted with the public portion of the EK and stored in the non-volatile storage. In consequence, the hash value is bound to the platform and the authentication password is only valid on this platform. If the execution of a TPM function needs owner authentication then the owner has to prove the knowledge of the password.

- **Storage Root Key (SRK)**: This key is also a 2048-Bit RSA-key pair. After the owner executed the *Take_Ownership command* to seize the TPM platform the SRK is generated. The SRK is stored inside the TPM and needs password authorisation for its usage. The private portion never leaves the TPM. The public portion is used to encrypt private keys of the first hierarchy level so that they can be stored outside the TPM. The keys of the first hierarchy level can encrypt keys of the second hierarchy level and so on. Hence, the SRK forms the root of a dynamic key hierarchy. This hierarchy is established out of limited storage reasons. Keys or even larger security relevant data must be stored outside the TPM without loosing its safety which is realised through this key storage hierarchy. All newly created keys get assigned a usage purpose and a key which is responsible for its wrapping.
  Based on privacy reasons, Storage Root Keys can be deleted by the owner and every time a new owner takes over the ownership of the platform a new hierarchy

is created. Hence, the new owner cannot access the former owners keys. Disadvantageously is that this makes the recovery of the stored keys impossible if no recovery provisions have been made before.

If applicable another kind of keys can be stored in the non-volatile storage part, the Attestation Identity Keys. **Attestation Identity Keys (AIKs)** are 2048-Bit RSA-keys which can only be created after the appropriation of the TPM. An infinite number of AIKs can be created but the owner must control the creation and activation of every AIK. AIKs are used for signing data which is created by the TPM for example a set of PCRs or keys. On that account, AIKs can be used to approve a TPMs authenticity to another party but do not reveal the identity of the TPM. This principle is known as Direct Anonymous Attestation and is discussed later in section 3.4.3.

The volatile storage is used as persistent secure storage for the Root of Trust for Storage (RTS). The RTS uses the keys for calculating all cryptographic functions. The management of this storage is done by the Key Cache Manager Module. In addition, the volatile storage is used for storing hash values in the PCRs for the Root of Trust for Reporting (RTR). The PCRs can also be managed outside the volatile storage of the TPM. Figure 3 outlines the mentioned storage design.

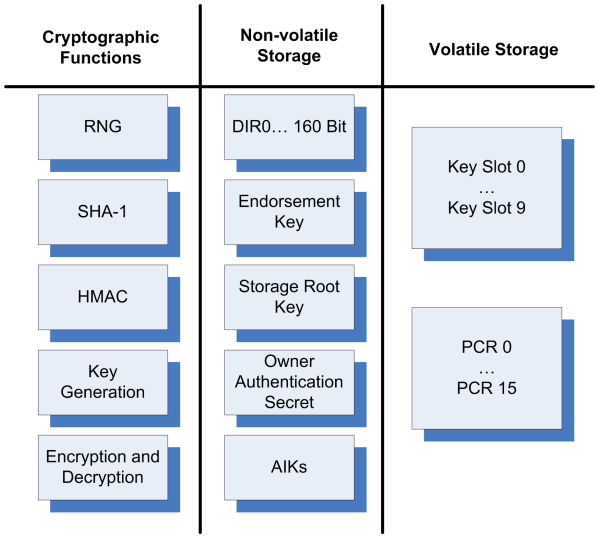

Figure 3: Volatile and Non-volatile TPM Storage

Key Types

TCG defines seven key types which limit the use of the different keys. One broad classification is as signing or storage keys. Additionally, the keys can be categorized as platform, identity, binding, general and legacy keys. Symmetric keys are classified separately as authentication keys [89]. Furthermore, the keys have special attributes which are to defined during the creation of the key. A key which is created by a TPM can be migratable or non-migratable. Migratable keys can be exported to other storages or TPMs so that the user is able to decrypt his encrypted data on other media

than only the TPM. In consequence, migratable keys are not trustworthy because they loose the hardware protection of the TPM. Thus, they should not be used for signing purposes. In contrast, non-migratable keys are always bound to one distinct TPM. Hence, non-migratable keys can be used for the authentication and identification of a TPM.

Following, the seven key types with their attributes are listed:

- ***Signing keys*** are asymmetric keys which are used to sign data as well as messages. They can be migratable or non-migratable. If the key is defined as migratable key then the user might enforce migration restrictions.

- ***Storage keys*** are asymmetric keys which are used to sign data and other keys managed outside the TPM.

- ***Identity keys (or AIK)*** are non-migratable signing keys. Those keys can only be used to sign data which originate from the TPM.

- ***Endorsement Key (EK)*** is an asymmetric non-migratable key. It is used to decrypt the owners authentication data and messages related to the creation of an AIK.

- ***Bind keys*** are used to encrypt small amount of data like symmetric keys on one platform and to decrypt the data on another platform.

- ***Legacy keys*** are migratable keys created outside the TPM. They can be migrated to the TPM to encrypt or sign data.

- ***Authentication Keys*** are used for Transport Sessions. They are symmetric keys which secure the transport sessions involving the TPM.

3.4.2 Certificates

In this section certificate types so-called credentials are briefly summarized. TCG defines five different credentials. Credentials are used to attest a TPM platforms or an AIKs identity. The credentials are described below (for more information see [89]):

- Endorsement Credential:
  This Credential is issued and signed by the manufacturer of the TPM who also creates the Endorsement Key. It documents that the key creation and the transmission of the key was carried out securely.

- Conformance Credential:
  This Credential can be issued and signed by any accredited instance. With his signature an issuer confirms that the design and the implementation of the Trusted Building Blocks comply with the specification.

- Platform Credential:
  This credential is issued by the platform manufacturer, the vendor or a trusted instance. It identifies the manufacturer and some properties of the platform. In addition, it contains a reference to the Endorsement Credential and the Conformance Credential. Its purpose is to approve that the platform contains a TPM as it is described in the Endorsement Credential.

- Validation Credential
  This credential can be issued by a manufacturer who measured safety-critical components as proper working. Not every component gets a Validation Credential issued only those which might pose a thread to the security of a system but are valued as trustworthy. Example components that might have Validation Credentials included are video adapters, disk storage adapters, memory controllers, communications controllers, network adapters, processors, keyboard, mouse and software.

- Attestation Identity Credential
  This credential is issued by a privacy CA which is trusted by the owner of the TPM and preserves privacy policies of its client. The privacy CA validates if an AIK which is used to sign TPM values is authentic and is related to the users TPM. Thus, the platform is authenticated by proving facts about the TPM. The information contained in the credential can be used to trust the platform via Attestation protocol.

3.4.3 Attestation

The variety of credentials mentioned in the previous section increase the security, the integrity and the trust of a module or a whole platform. Besides, the credentials are used to measure integrity statements about a platform and attest its integrity. This is also known as **integrity reporting**. The TCG specification describes two principles of attestation, a general model using a Privacy Certification Authority (PCA) and a model using Direct Anonymous Attestation (DAA). The latter was developed to shelter the privacy of the user.

An AIK credential confirms the validity of the public AIK key. This credential is issued by a PCA. But, instead of using one AIK for each request the owner is able to use different AIKs certified by different PCAs to obfuscate his identity. This method of pseudonymity prevents others from developing an owner profile. Figure 4 describes how attestation with an intermediary PCA takes place.

First, if the platform or the owner of the platform respectively wants to use a service then it has to query the Privacy Certification Authority. The EK and a specially for this session created AIK key are sent to the PCA. Second, the Privacy Certification Authority insures itself if the querying platform is a trusted platform. If the platform is trusted the PCA signs the AIK and sends the credential back to the platform. As a last step, the trustworthiness of the platform has to be proven to the verifier. For this,

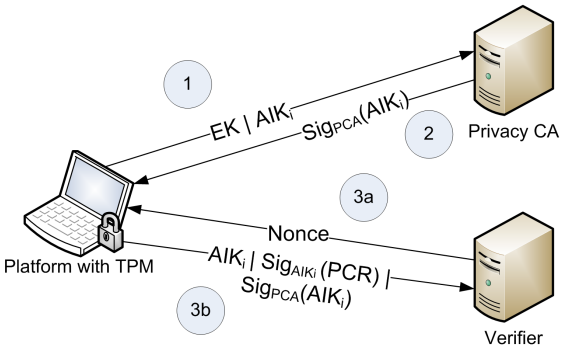

Figure 4: Attestation with Privacy Certification Authority

the verifier needs to check the configuration values of the platform through an attestation request. The Root of Trust for Reporting (RTR) is responsible for effecting an attestation. Accordingly, the RTR signs the integrity describing configuration values of the platform with the previously signed AIK. A Nonce prevents the reuse of this attest. The Nonce can be sent by the verifier to challenge the platform. The attest and the belonging AIK credential are sent back to the verifier. Then, the verifier checks the trustworthiness of the PCA which issued the AIK Credential. If it is a trusted PCA then the public portion of the AIK extracted out of the credential is used to encrypt the configuration values of the platform. If the configuration values meet the requirements the communication can take place.

The traditional way of attesting the integrity of a platform still entails the risk of revealing information about the TPM. The usage of the Privacy Certification Authority is privacy critical. The PCA can link the EK to an AIK and consequently, the TPM to an attestation. But to provide privacy to the user, the verifier must only learn that a TPM was used but not which particular one. That is the reason why the TPM specification 1.2 adopts a new concept, Direct Anonymous Attestation (DAA). This concept intensifies the privacy properties of a platform while not divulging the public portion of the Endorsement Key to a Privacy Certification Authority. DAA is based on zero knowledge proofs which is an interactive method for proving the authenticity of a TPM to an enquirer without revealing anything other than the veracity of the statement. I.e. no information that clearly identifies the TPM is revealed. Figure 5 describes how Direct Anonymous Attestation proceeds. So far, these are the considerations of the TCG. However, in [55] an attack on the remote attestation procedure is discussed. In the described scenario an attacker is able to receive a certificate for his key by an operation actually initiated by the TPM owner. For more information about the attack and its prevention see 3.6.

3.5 TPM v1.2 Specification Changes

This section provides a summary of the v1.2 specification changes with respect to the v1.1b TPM specification [84]. Following the changes are briefly discussed:

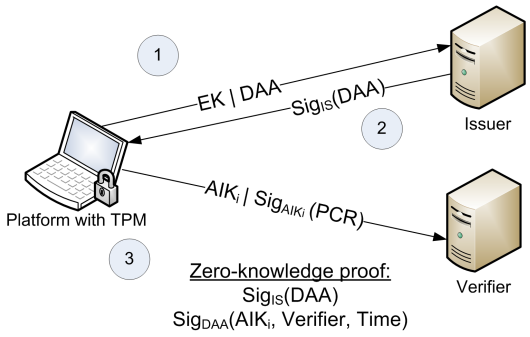

Figure 5: Direct Anonymous Attestation

- **Direct Anonymous Attestation** - see Section 3.4.3 about Attestation
- **Locality** - A concept that considers that there are hardware and software-based processes external to the TPM with different security or trust levels. Locality-specific PCRs are introduced to save the characteristics of external software processes. Permission can be granted to external software processes like a trusted operating system based on the locality-specific PCRs.
- **Delegation** - A feature that allows the owner to have fine-grained control over the use of owner-authorized TPM commands. In version 1.1 software modules which need owner authorisation need to be provided with the owner password. This gives the software the undesirable possibility to perform any owner authorised actions. In version 1.2 the owner has the ability to give a software module the ability to use any individual owner authorised TPM command or subset of commands, without granting permission to use any other TPM commands.
- **NV Storage** - The v1.2 TPM Specification defines the usage of the Data Integrity Registers (DIR) - 20-bytes non-volatile storage locations. The owner of the TPM now controls the write-access to the DIRs. They potentially can be used for privacy sensitive data like the EK certificate.
- **Transport Protection** - A facility that improves the security of communication channels between the TPM and trusted processes. Version 1.2 protects the integrity and confidentiality of commands sent to the TPM. All commands sent to the TPM can be logged with newly designed Transport Sessions. The logs offer reliable evidence that a sequence of operations using the TPM occur. User privacy can be enlarged because confidentiality of data exchanged in a transport session is ensured. Furthermore, it can be assured that a particular desired sequence of commands to a TPM was actually executed as intended.
- **Monotonic Counters** - This function helps to prevent replay attacks. It increases the security of the system and so the privacy of a system. Attacks against TPM protocols which can bear the risk of revealing personal data are prevented. For example, a Monotonic Counter is included in a message and each time an

action takes place it is incremented. If the message with the counter value is replayed then it can be discovered through comparing the actual Monotonic Counter value with the value in the message. A 1.2 TPM provides a limited number of Monotonic Counters.

- **Tick Counter** - This function is a compromise of implementing a time source into a TPM and saving costs. A real time clock would be too expensive so the Tick Counter is included. It counts the ticks and the tick rate since the counter is started. So, it is possible to calculate real time information with some external support.

- **Context Save and Restore** - This function realizes efficient shared use of a TPM by different software layers. If a TPM user encounters a resource problem the appropriated objects can be saved through a Context Save and then the resource can be released and used. Later, restoring of the state of the TPM is done by using the Context Restore function before dispensing the TPM to another user.

- **Clear Endorsement Key** - In version 1.2 it is possible to erase the Endorsement Key on the platform. This results in, loosing its identity and thus cannot create AIKs until a new EK is created and validated.

- **General Purpose I/O Function** - A low-bandwidth access-controlled physical communication channel can be implemented between the TPM and other hardware components in the platform. This channel protects the access of public keys of trusted hardware elements by preventing the uncontrolled exposure of the identities.

- **Key Migration Changes** - Key migration in version 1.1 could only take place with loosing security of the key. In version 1.2 migratable keys are introduced which only can be ported from and to protected environments. By this means, those keys are seen as more trustworthy than general migratable keys. But still, loose some security compared to non-migratable keys.

3.6 Attacks on a TPM

There exists a variety of attacks on TPMs. Attacks on TPMs have been described in various publications like [41, 45, 55, 58, 78, 91]. In the next sections a number of selected attacks on TPMs and some possible countermeasures are described.

3.6.1 TPM Reset Attack

Kauer [58] described a TPM Reset attack on a v1.1 TPM. The attack is based on setting the reset bit of a PCR. In doing so, the TPM is reset without resetting the whole platform. This is inconsistent with the assumption that PCRs can only be reset if control is passed to trusted code. As in [58] said the Low Pin Count (LPC) bus is a good point of attack. The bus owns a separate reset line and most TPMs

are connected through it. Sparks [78] adopted this attack. He states that the current design of TPMs allowed them to easily monitor the LPC bus and to enter the TPM. Commonly, a TPM daughterboard is connected through the LPC bus permitting the easy swapping of TPMs and the easy programming of the LPC bus in comparison to other busses. By first recording a trusted boot process he and his team were able to compromise the whole system. The chance of recording a trusted boot process was already demonstrated in [61]. In the next step they swapped the trusted hard drive against another hard drive running a different operating system. The TPM will notice this change and not disclose any secrets. Then they sent a reset signal to restart the system. The reset was carried out by connecting the LRESET line of the LPC bus to GROUND using a piece of wire. They brought in the recorded measurements and thus convinced the TPM to be in a trusted state. But actually being in the same untrusted state as before. The detailed attack is described in [78].

In order to avoid the possibility to reset the PCRs without resetting the whole platform it would be desirable to integrate the TPM with e.g. the BIOS. Joining both BIOS and TPM makes it more difficult to physically access the lines leading to the TPM. Lines to the LPC bus are not accessible as easy as before minimizing the risk of this attack. Another possibility to avoid this attack is to implement encrypted communication between the individual components. So, the components are able to distinguish whether they are communicating with the correct communication partner or not. A secure communication can be for instance realised through the in v1.2 TPM Specification newly adopted TPM transport sessions. Transport sessions serve to protect the confidentiality of data exchanged.

3.6.2 Software Attack

In addition to the TPM reset attack Sparks [78] described in his technical report a software attack which renders the measurements of the TPM ineffective. His key issue is that the remote attestation is not reliable. This is because the requestor cannot make sure that the system configuration has not changed since it was last measured. In particular, Sparks [78] criticised IBM's IMA which is described later in 4.6.1. He explained that IMA measures binaries at load time but if it changes after it has been loaded, the system state would not match the actually measured configuration. For demonstrating this vulnerability Sparks [78] changed a process's page table. They copied the page table's entry which maps to the targets virtual address into the kernel's page table at an arbitrary address. Then, they marked the entry as writable and wrote data to the corresponding page in physical memory. So, they modified the way a program executes when it arrives at an instruction in the modified address range. In consequence, if no continuous measurements are carried out the measurements of a TPM are useless. The complete attack is depicted in [78].

The problem with Remote Attestation is that it cannot be guaranteed that the system configuration has not changed after the TPM_Quote was sent to the requestor. In [78] it is explained that a mitigation method could be to create a system that

memory cannot be changed after it was measured. There it is also pointed out that this approach would be difficult to realise since a computer needs to write memory to function properly. Instead it is proposed to monitor the state of the memory. If memory changes this must be included in the measurements or actively signalled for example to the requestor.

3.6.3 Platform Reset Attack

The TCG itself discusses a Platform Reset Attack in their publication [91]. The idea of this attack is based on the problem that the contents of the RAM are not immediately lost when a platform is rebooting or shutting down. A so called cold boot is executed to carry out this attack. Cold boot means that the computer is turned off and on while not shutting down properly. Malicious software can be booted during the next start up for example from a removable file. The malicious boot software copies the contents of the RAM and tries to extract security data like keys. Another step taken could be to remove the memory hardware and place it into another computer without security mechanisms implemented. So, it is possible to read the RAM in circumventing security mechanisms. Unfortunately, the copied contents could bear keys or other secrets. All this steps must be carried out quickly since the data vanishes after a short time span. Cooling down the memory chip prevents the data from disappearing immediately and the attacker gains some time carrying out his attack. This attack has been verified in [56] to be successful against TPM secured systems. In [56] it is said that a TPM can stop a key from being loaded into memory but if it once resides in memory there is no possibility for the TPM to prevent the capture of this key. 'Notably, using BitLocker with a Trusted Platform Module (TPM) sometimes makes it *less* secure, allowing an attacker to gain access to the data even if the machine is stolen while it is completely powered off.' [56] This is because the system would automatically mount the machine to the previous status as soon as it is powered on.

One countermeasure is defined in [91]. An operation which is initiated by the BIOS overwrites the system memory on the next platform reboot. All information that may be exposed to an attacker must be cleared by zeroing the memory. This does not need to be done at each reboot but the TCG introduces a Memory Overwrite Request (MOR) bit which indicates the overwriting of memory content [91].

3.6.4 Remote Attestation Attack

In [55] an attack on the TPM_CertifyKey command is discussed. The attack is based on the manipulation of TPM key handles or the integrity of blobs. If an attacker gains physical access to the TPM he could interfere the creation of a certificate for a key and insert key data for a key in his possession. Thus, the attacker receives a certificate for his key by an operation actually initiated by the TPM owner. As depicted in [55] the TPM_Certify key command generally includes the following:

- Key handles of the AIK which is used for certification.

- Key handles of the key to be certified.
- A nonce (e.g. from a certificate requestor).
- HMAC based on the usage authorization data of the AIK (not covering key handles).
- HMAC based on the usage authorization data of the key (not covering key handles).

The attacker just needs to block the command and change the HMAC authorising and the key handle of the key against those of his key. Then, the TPM creates a certificate for the key of the attacker instead of creating a certificate for the TPM owner. The response must be blocked again so that the TPM owner does not get hold of the falsely issued certificate and thus notices the attack. The attacker can now use this certificate in relation with his key for disguising his identity and impersonating the TPM owners identity. Another possibility for an attacker entails in exchanging one key by another key that has the same parent. It is claimed that this will not be noticed.

If an attacker gets hold of a key that is bound to the identity of a TPM he owns a powerful tool. For example, the attacker might be able to sign malicious data with this identity and foist this data on someone trusting in this signature. [55] offered a countermeasure against the compromise of the TPM_CertifyKey command. It is to establish a TPM transport session. Transport sessions are newly included in the v1.2 TPM Specification and protect the integrity and confidentiality of commands.

3.6.5 Other Vulnerabilities

Besides, there exist some vulnerabilities of TPMs which are only briefly mentioned at this point. In [78] a timing attack on TPMs is theoretically discussed. This attack is based on Kocher's side-channel attacks on RSA and differential power analysis attacks against RSA engines and Boneh and Bromley's RSA Timing Analysis attack against the RSA engine in a STMicro v1.2 TPM. Furthermore, Kauer [58] analysed in is publication if TPM-enabled boot loaders execute code that is not hashed. He found that GRUB (cf. Section 4.6.1) loads and extracts a kernel image at the same time instead of loading them completely into memory and extracting them afterwards. So, measuring a file separately from loading it already adds TCG support to GRUB. Kauer [58] mentioned that an attacker who has physical access to the network or the disk could insert different data at load time. Consequently, the measured and the executed code can differ. TrustedGRUB, on the other hand, would solve this problem. In [58] is also criticised that current CRTMS of many machines are freely patchable. No signature checking would be performed on updates und this could lead to untrustworthy code running and extending PCRs.

4 Traffic Monitoring Systems

Measuring instrument designers now deal more often with the problem of admissibility of the data collected with the created instrument. In general, measuring instruments are subject to strict controls by the calibration office in order to correctly determine momentous transgressions and to demonstrably expose the actual originator. Occurring difficulties, during the unambiguous mapping of measurement values to another collected value, result in rejecting the measurement. Therefore, it is desirable to diminish the measuring inaccuracy compared to already existing technologies. The instruments must accomplish all requirements given by law and enforce the correctness of all created measurements. This chapter now addresses measuring instruments most notably Traffic Monitoring Systems (TMS) in relation with Trusted Computing. Hence, the designated use of a device automatically collecting digital evidence is explained in detail. Then, considering the security point the potential threats are clarified. So that later on, the security goals and the resulting system requirements can be described. Next a brief summary is given which describes some of the currently available standards and laws that must be obeyed designing a TMS collecting admissible digital evidence. And last, the Trusted Computing solutions are explained forming the most important parts of a TMS.

4.1 Use Case

An embedded system is designed for the automatic collection of measurements. The collected measurements can be used for several analyses. Together with some security information, the measurements provide trustworthy digital evidence admissible in court. The evidence collected points out when and where an event happened. Thus, the measured data shall not be tampered at any time since the time of its creation and its presentation in trial. Since trials can be long lasting or can take place long time after the evidence was collected it is necessary that the evidence is verifiable at any time. Particularly, the contemplated embedded system is designed for measuring traffic data. The embedded system should contain all functionalities that can be used to impose all kinds of traffic information. Usually, there will be specialised systems with limited functionalities to save costs and to increase the systems' performance. This thesis will focus upon Traffic Monitoring Systems in general. The planned traffic monitoring device shall be able to collect different sorts of Measurement Values (MVs). Some possible MVs are:

- Infrared pictures
- Digital pictures
- Time values
- Speed values
- Number of vehicle

- Type of vehicle
- Location values
- Distance a vehicle covered
- Traffic volume values

Coming up with these MVs the system will be able to be adopted for multiple fields of application. According to the system requirements all sorts of TMS can be constructed. All of them holding different system features. Up next some conceivable system features are listed.

- Speed measurement – Different speed measurement techniques can be implemented using laser or radar. Radar guns or stationary speed cameras are probably the most known methods for billing speeders.
- Toll systems – In a variety of countries it is compulsory for every driver to pay a toll charge. Each country applies different methods like point pricing, cordon pricing, zone pricing, distance-based charging etc. Point pricing means that a vehicle is charged a fee when passing a given point. Similar is the cordon pricing method which charges a vehicle when crossing a border. The zone pricing bills a driver when entering or exiting a zone. Whereat, the distance-based charging calculates the fee per mile/kilometre the driver travelled. (For more information about charging types see [50])
- Congestion charging schemes – The last years, congestion charging is in particular introduced in big cities and their surrounding areas. With this step it is tried to reduce the traffic load in cities. Drivers are charged when entering a special zone or must pay once a year designated annual fee. At the moment, London is the model for such kinds of charging schemes [82].
- Traffic analysis – General traffic analysis like calculating the number of vehicles passing or determining the period of time of most traffic can be important for planning, e.g., new transport networks, rout diversions or traffic light circuits.
- Recognition of vehicle types – Sometimes it might be necessary to estimate the traffic load and especially the load of special vehicle types like trucks.
- Automatic Number Plate Recognition (ANPR) — The ANPR equipment identifies the number plate of vehicles passing through the field of view of the sensors. Recognition of number plates can be employed to impose action against drivers who have not paid their vehicle dues [82]. As well, it can be used for distance speed controls [57] to identify the vehicles not maintaining the speed limit.

It should be noted that not all possible MVs as well as purposes are mentioned at this point. Certainly, there exist also other fields of application depending on a variety of MVs.

Defining Roles

At this point, the different roles of the system are described. The description adverts to the interactions between single roles and their tasks. Figure 6 shows the different roles of the system.

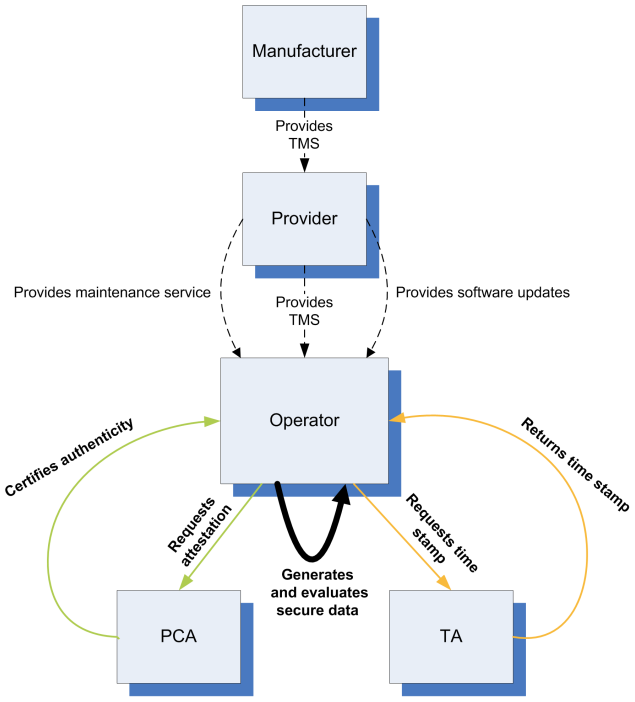

Figure 6: Roles

Following roles can be distinguished in this system architecture a manufacturer, a provider, an operator, a Privacy Certification Authority and a Timing Authority. The **manufacturer** produces the Traffic Monitoring System and delivers it to the provider. In particular, this includes generating the Endorsement Key and issuing an Endorsement Key Credential for the TPM. Later, the credential attests that the EK was properly created and embedded in a valid TPM. Additionally, the manufacturer might issue some AIK Credentials for AIKs created in the manufacturing process. This alternative is optional and depends on the system design.

The **provider** commissions the TMS from the manufacturer. When it is ready for use, the provider places the system to the operators' disposal. It may also be his responsibility to provide an archiving and evaluation possibility. This is optional to the operator. If the operator owns qualitative hardware and software resources for archiving and evaluation then the provider is not needed to offer this kind of hardware and software. Unless, it might be a necessity to make hardware and software available to the operator. Assembling the system, providing software updates as well as providing maintenance and repair services are further responsibilities of the provider. In addition,

the provider might issue some AIK Credentials depending on the system architecture.

The **operator** now possesses and controls the Traffic Monitoring System. His responsibility is the operation of the whole system and evaluating its outcome. Since the operator is running the Traffic Monitoring System and estimates its results he is responsible for the correct storage of digital evidence. It must be taken care that storage conditions appear to be appropriate. As Mason et al. [62] remarked it is possible for data to be rendered unreadable if it was stored for example unprotected from humidity, extreme temperatures and strong magnetic fields. Furthermore, the operator is responsible for the correct presentation of the evaluation results. That will mean, the operator must provide an expert for court who presents the results if necessary. This expert must deliver expert evidence about facts from the domain of their expertise and with primary duty to the court [4, 11, 73].

Out of privacy reasons, a **Privacy Certification Authority (PCA)** provides functionalities to supply privacy on the one hand and traceability on the other. The PCA provides the usual attesting functionalities. Basically, this comprises the issuing of digital certificates. Certificates contain a public key and the identity of the owner. The PCA attests that a public key belongs to an entity with issuing a certificate. In the attestation certificate the public key and the identity of the entity must be noted. Privacy is still warranted as long as the certified identity is a pseudonym. Only the PCA is able to trace the pseudonym back to the actual identity of the key owner. Certificates in a TPM incorporate the attestation that a TPM was generated compliant to the standards and that a key was generated correctly by the TPM. In this context, the PCA functionality is required to issue the credentials stating the origin of the data and binding the data to a particular platform. It attests the authenticity of the system credentials. Another party trusting the PCA verifies the PCA's signature and that a certain public key belongs to the system. In this case, the role of a PCA can be external or internal. External means that a so-called Trusted Third Party (TTP), the PCA itself, is delegated with the mentioned tasks. Internal means that the manufacturer, the provider or the operator may assume the role of the PCA and issue certificates themselves. In this instance, it must always be provided that the whole process takes place in a trusted and protected environment. Compromise of the certification authority leads to loss of security of the entire system.

The **Timing Authority (TA)** offers an adequate time stamping service so that a law conformant time stamp can be obtained. With the help of a TA a real time value can be provided to the TPM which does not own an inherent time but only a tick count. Later, this real time value can be harmonised with the current tick value of the TPM. So that, the actual time of an event can be determined. For a detailed explanation see Section 4.6.3. It is to be noted that also this role must not inevitably be separated in all cases from other roles. It is possible for example that the operator adopts this role.

Depending on the system architecture roles can be merged or separated as well as additional roles can be defined. For instance, a charging unit or billing unit surely

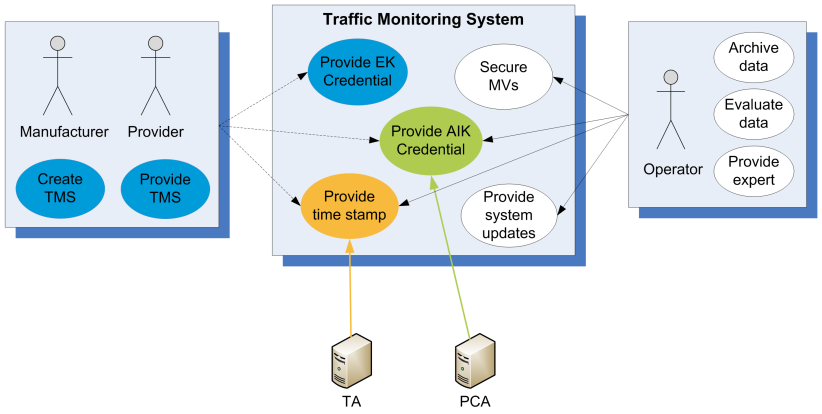

Figure 7: Roles and their Tasks

would fit into the system design of a toll system. Completing, Figure 7 illustrates the roles and their tasks.

4.2 Threats and Requirements

Having defined the purpose of the embedded system generating admissible digital evidence the outside impacts threatening the TMS as well as the requirements improving the trustworthiness of the TMS must be analysed.

4.2.1 Threats

The comprehensive objective is to provide admissibility of measured data produced by a Traffic Monitoring System. Especially, if the Measurement Values serve as digital evidence in court it must be guaranteed that it cannot be disputed. Since digital evidence can be challenged in multiple ways we try to design a secure environment that exactly destroys those challenges. For instance, the reliability of the program generating the data is questioned. Doubt can also arise if it cannot be proven that data has not been altered between it was collected and it appeared in trial. Regardless if data shall be used in court or not, a system not working correctly does not help anyone. Wrong measured data could significantly affect statistics and result in wrong outcomes. The operator just would incur losses. To that effect the system is analysed on threats intimidating the Traffic Monitoring Systems' trustworthiness. In the first place, an overview over the threats is given. Followed by a detailed threat description.

Threat Overview

T1 Attacking the initialisation process.

T2 Malicious or altered software affecting the Traffic Monitoring System.

T3 Attacking the permissions.

T4 Attacking security relevant data.

T5 Attacking the authentication and transmission process.

T6 Attacking the identity.

T7 Attacking the completeness of data.

T8 Inside effects impairing the Traffic Monitoring System.

T9 Outside effects impairing the Traffic Monitoring System.

Detailed Threat Analysis

T1 Attacking the initialisation process.

Description: The time period before system applications are started is known as initialisation process. The initialisation process can be harassed by an attacker. Meaning, from the power on to the time when control is handed over to the applications every component must be checked for failures. The phase of initialising a system exhibits several points of attack. First, during the boot process fraudulent or modified software could be inserted and booted. The inserted malicious software might have different functionalities than the actually installed trusted software. This could change the whole system functionality or only some small parts. Nevertheless, these changes harm the systems' creditability significantly. If no counteractive measures are taken these malicious software changes are hard to detect and can remain undetected. Second, if an attacker is able to partly or completely reset the initialisation values at the activation of a system the system functions can be derogated. Altered initialisation values can produce a different output than the initial correct values would have produced. E.g. different initialisation values like constant variables used in calculations change the result unwanted. Third, considering the last two cases if the attacker is able to cheat software components he might also be capable to change hardware components. Especially, if he is competent to change the software in such a manner that it disguises the replacement of hardware components.

Consequences:

- Malfunction of the system.
- Non-optimized performance of the system.
- Loss of initialisation values and data.
- Attacker might gain unauthorised access to security relevant data.
- Authenticity, integrity and privacy of data not ensured.
- Authenticity and integrity of system not ensured.

Countermeasures: A countermeasure that is taken is to establish a trusted boot as described later in Section 4.6.1. Before handing the control to any application the system is checked for malicious changes by means of hash values. Every component, software or hardware, participating in the initialisation process are examined and checked against known trustworthy elements. All detected intrusions are documented and in special cases, if previously defined, the corrupted parts are not given any control.

T2 Malicious or altered software affecting the Traffic Monitoring System.

Description: This threat in distinction from T1 addresses the software running on the Traffic Monitoring System after the initialisation process. An attacker may alter or replace the present software of the TMS or may introduce malicious software into the TMS at runtime. Gaining access to the device using fraudulent software an attacker is capable of changing the functionalities of the device. There exist quite a number of attack points to change the software. Malicious updates could be introduced from the Archiving and Evaluation Units (AEU) side or through unprotected corrupted interfaces. Furthermore, the operating system loader code, the actual operating system code or the software running on the system could be tampered. Depending on the actual system design different vulnerabilities of the system can be identified.

Consequences:

- Malfunctioning of the Traffic Monitoring System leads to inadmissible data.
- Checking the instruments' software and changing its settings will become necessary.
- Attacker may gain access to keys.
- Stability and functionality of the system may decrease.
- Authenticity, integrity and privacy of data are harmed.
- Authenticity, integrity and privacy of the system are harmed.

Countermeasures: Countermeasures are strong authentication mechanisms. No software updates shall be able to be introduced through unauthenticated services. The system should naturally only accept software updates from predefined entities. The updates must be checked for its authenticity and integrity itself to prevent an attacker from pretending a wrong identity. Besides, dynamic boot of trust (see Section 4.6.1) and constant lifetime checks of the system are needed to add security during runtime. Monitoring the system during runtime and only executing trusted code decreases the possibilities of an attacker to affect the TMS. Apart from this, TPM protected storage of security relevant data helps to defend against fraudulent software changes that could affect this data. If data can only be accessed via trusted processes it can be at least be guaranteed that security relevant data was not impaired or used in an unintended way.

T3 Attacking the permissions.

Description: An attacker is able to change the access control and grants themselves permissions to operations that he is not allowed to execute. Predefined security levels are bypassed. It is to be mentioned, that first the attacker needs to overcome the authentication procedures intrinsic to the system to gain access to system permissions. With compromising the authentication procedures the whole system is threatened.

Consequences:

- Attacker gains administrative control of the system.
- Functionality and stability of system is not warranted.
- Authenticity, integrity and privacy of data is threatened.

Countermeasures: To build a security basis and to avoid permissions to be attacked and changed, strong authentication techniques must be warranted. Only persons with special authorization are allowed to change permission settings. Therefore, these persons must authenticate themselves to the system. Furthermore, hardware protection in terms of a TPM can be used to protect security relevant data like keys. The TPM ensures that only the TPM owner is able to access security relevant data like key. No permission is granted to anyone not knowing the owner secret. Thus, safety critical data encrypted with protected keys cannot be accessed.

T4 Attacking security relevant data.

Description: An attacker is able to access security relevant data like Measurement Values, timestamps and especially signing and storage keys. Any way which allows an attacker to access data he is not allowed to is a security risk. Accessing comprises reading, changing, using and deleting security critical data. Ideally, the attacker needs to overcome the authentication process and permissions to gain access to the system critical data (cf. Threat T3).

Consequences:

- Attacker may gain access to permissions (cf. Threat T3).
- Functionality and stability of system is not warranted.
- Authenticity, integrity and privacy of data is not secured.

Countermeasures: Critical data must be protected separately from any unauthorised access. Special hardware can be used to secure the storage of security relevant data. In designing a TMS, a TPM is used as hardware security anchor providing hardware protection for keys and other security relevant data. This is done via a special key hierarchy implemented in the TPM (cf. Section 3.4.1). For example, keys or timestamp information can be encrypted by a key which then is encrypted

by another key and so on. So, it is possible to store bulky data outside the TPM but still using its hardware protection mechanisms.

T5 Attacking the authentication and transmission process.

Description: During the authentication process the attacker is able to listen to the traffic sent through the network. Either the attacker is only able to listen or he is also able to authenticate as legitimated user. The difference is that the attacker does not only listen but also injects traffic. Anyway, the attacker gains access to transmission data. In the latter case, the attacker pretends to be the correct communication partner. All data, flowing between the victim and the network, could be read. Eavesdropping attacks, man-in-the-middle attacks, spoofing attacks, tampering of data, replay attacks etc. are possible.

Consequences:

- Authenticity of communicating party is not warranted.
- Operational access to data in the name of another person is possible.
- Authenticity, integrity and privacy of sent data is not secured.

Countermeasures: Countermeasures against this type of attacks are strong authentication techniques tailored on the de facto system design. A strong authentication algorithm protects a user from communicating with a wrong communication partner. To ensure the integrity and authenticity of data signing procedures can be introduced. To ensure privacy of security relevant data encryption techniques can be used. For instance, keys can already be exchanged during the manufacturing process. Given that the manufacturing environment is trusted, this is probably the most secure way to exchange signing and encryption keys between communicating parties. Later, the exchange keys are protected against unauthorised usage through the TPM.

T6 Attacking the identity.

Description: An attacker adopting a trusted identity can almost do anything within the scope of this identity. In adopting the identity of a trusted device an attacker can try to foist wrong Measurement Values. This includes false measurements as well as to pretend that no measurements were created. On the other hand, an attacker simulating or undertaking an identity can gain access to security relevant information like private data or software secrets. The difference to Threat T5 is that this threat considers the loss of authenticity and integrity of components and data not just during the transmission process.

Consequences:

- Access to data in the name of another component.

- Non-optimized performance of the system.
- Authenticity, integrity and privacy of data are harmed.

Countermeasures: That an identity can be forged is prevented in embedding a TPM in the Traffic Monitoring System. A TPM does not only offer protected means for cryptographic functions but also offers a distinct identity (see 3.2.1). This identity is protected through the TPM. A TPM is considered to be always trusted complying with all requirements needed for the EAL4+ of a Common Criteria security evaluation [44]. (For a brief introduction to Common Criteria and its security profiles see [42,43].) Moreover, the TPM provides signatures, encryption mechanisms, trusted boot and dynamic trust support. All these functionalities can also strengthen the security of the identity.

T7 Attacking the completeness of data.

Description: Attacking the completeness of data means that a set of measured values is not generated or transmitted. The attacker succeeds in changing the system in a way that no Measurement Values are transmitted or even worse generated. Subsequently some possible scenarios are mentioned. In the case of a TMS, the generation of data can be stopped or interfered by using a special interference pulse. This impulse harms the sensors and causes them to fail. This is likely, since the usage of radar warning or interference systems increases, even though they are permitted by law. Those devices warn the driver acoustically and optically of any radar and laser frequencies or interfere the measuring. Germany, §23 of the StVO [33], disallows drivers to operate or to carry along an operatable technical device which can be used to show or interfere traffic surveillance procedures like speed measurements. Another possibility is that the transmission is delayed and causes the loss of data. It might also be possible that, as described in Threat T9, an attacker covers the sensor or changes its viewing angle and thus, prevents the creation of new values. Certainly, software and hardware changes must also be considered in this threat.

Consequences:

- Malfunction of the system.
- Non-optimized performance of the system.
- Loss of data.
- Worst case: Total rejection of all measured values during a statistical evaluation.

Countermeasures: It cannot be prevented if an attacker physically hinders the Traffic Monitoring System from creating values. This means, if someone curtains, misplaces or interferes the sensors it can be detected but not circumvented instantaneously. The only possibility for the system entails in documenting and setting of a sabotage alarm [32,59]. Documentation comprises reference measurements in periodical intervals (cf. Threat T9) as well as the establishment of a

trusted boot process and dynamic trust (cf. Threats T1 and T2). This should be sufficient for the detection and reporting of any failure. To check actively whether measuring data transmitted was complete and nothing generated got lost a PCR inside the TPM can be allocated to register all created values (see Section 4.6.2 for more information about the completeness value). By now, video cameras or motion detector have already implemented special sabotage alarms which alarm the operator if the sensor was for example covered (cf. Threat T9). This would give the operator a more instantaneous possibility to react than only documenting the failures and remedy them after evaluation.

T8 Inside effects impairing the Traffic Monitoring System.

Description: Inside effects are failures of system components, system software, etc. either caused unintended or intended. Unintended failures can be for example caused by programming or manufacturing faults. Intended failures can be caused by an attacker taking advantage of those vulnerabilities to fail. Buffer overflow or the insertion of wrong input parameters can cause the system to break down or malfunction. There exists a variety of attacks exploiting any kind of weakness.

Consequences:

- Malfunction of the system.
- Non-optimized performance of the system.
- Loss of data.
- Integrity of data is harmed.

Countermeasures: As described within earlier threats the trusted boot process and the dynamic trust document any changes to a trusted system configuration. This also includes unintended failures and failures caused by an attacker (cf. Threats T1 and T2).

T9 Outside effects impairing the Traffic Monitoring System.

Description: Outside effects on the Traffic Monitoring System mean every effect that arises exterior to the instruments' inner workings. Counted are environmental influences as well as vandalism against the measuring system. Environmental conditions can harm the instrument and induce a failure. Alternatively, the human destructiveness must be reflected. There might be attackers trying to damage the TMS forcefully. Damage or devastation of the instrument can occur through random physical attacks. With some preparation time, an attacker might be able to change system components. Another problem would be the covering of the sensor or the relocation of the entire system. This would make the creation of data useless or cease it at all. Altogether, the minimum mischief would be a performance variation or loss of the system.

Consequences:

- Malfunction of the system leads to inadmissible data.
- Failure of the system.
- Cost for rebuilding the system or system components.
- Introduction of some degradation of the Traffic Monitoring Systems' lifetime.
- Authenticity, integrity and privacy of data are impacted.

Countermeasures: As described before, tamper resistant hardware can be used to detect and document any changes in the functioning of a system. But, as this threat is mainly concerned with physical attacks damage resistant hardware is necessary. To protect against human vandalism indestructible materials can be used for building the casing. Also reasonable are for instance seals attached to the casing to detect impacts. If a seal is broken the TMS needs to be examined. The casing shall protect against any outside influences either caused by nature or by persons. E.g. a rain and wind protection is practical to protect against bad weather conditions. As a countermeasure against covering or dislocating of sensors reference measurements can be taken in periodic intervals. If using a camera sensor a reference measurement can be for instance constituted out of a picture. In most cases, reference pictures must be matched manually. If some suspect changes occur someone can be send out to check the situation. Usually, pictures cannot be matched automatically because of environmental changes. For example, weather conditions, construction sides that are installed or trees that are cut change the picture values generally used for comparisons. These changes are regular but cannot be recognized by a general picture analysing software. A more sophisticated countermeasure against outside influences is the active sabotage alarm. This alarm is set off automatically and the operator is alarmed if the lens of the sensor is covered, spray-painted, removed, defocused or if the viewing angle is changed. This kind of alarm was specially developed for areas where vandalism occurs like prisons, schools or public transport. Moreover, it can be employed in areas in that weather, vibrations or dirt can harm the camera functions [32, 59].

4.2.2 Requirements

During the establishment of a system the outside impacts on the system must be measured as well as the countermeasures which can be taken to protect the system. The indicated tasks of a Traffic monitoring System (Section 4.5) are intimidated through the threats described in the preceding section 4.2.1. Consequently, requirements are obliged to be derived concerning the system purpose and threats. This section describes in more detail the requirements for a TMS. First, an overview over necessary requirements is given. Afterwards, all requirements are described in more detail.

Requirement Overview

R1 Providing a trusted boot process.

R2 Providing software attestation.

R3 Protection against intentional changes.

R4 Protection against accidental or unintentional changes.

R5 Securing data after creation.

- Securing completeness of measurement data.
- Securing integrity of measurement data.
- Securing authenticity of measurement data.
- Secure storage and protection.

R6 Securing keys.

- Key protection.
- Confidentiality of keys.

R7 Providing access control mechanisms.

R8 Providing secure authentication.

R9 Providing secure transmission techniques.

R10 Providing a secure hardware environment.

R11 Providing for completeness of data transmitted.

Some potential supplementary requirements which might become obligatory in a different design are fault recovery or back-up facilities. Depending on the effective system design, there might exits additional requirements which are neglected at this point.

Detailed Requirement Analysis

R1 - Providing a trusted boot process.
Legally relevant Traffic Monitoring Systems shall be secured against fraudulent behaviour in any way at the lowest stage of execution. Only by knowing that a trusted environment was booted upper layers of the system can be trusted.

Reasons for an untrusted boot:

- Malicious hardware and lower level software that was inserted by an attacker.
- Exchanged OS inserted by an attacker.

- Malicious kernel inserted by an attacker.
- Failures in hardware or software that unintentionally occurred during runtime of the system.

Provisions: The trusted boot functionality of a TPM is used to implement checks before starting system functionalities. This so-called static boot process is realised by hashing all components and comparing the hash values against reference values before any control is transferred to the next state (compare to Section 3.2.3). At this stage, everything up to the operating system but the application software is checked. Details about the dynamic trust of running software are described in R2 and Section 4.6.1.

Threats: This requirement specially refers to Threat T1. Threats T7 and T8 also can be diminished with this requirement.

R2 - Providing software attestation.
Legally relevant Traffic Monitoring System shall be secured against fraudulent or faulty software execution. The instrument shall be protected against swapping of application software. Only by knowing that trusted software was executed you can believe in the correctness of data generated by a monitoring instrument.

Reasons for insecure software execution:

- Malicious software that was inserted by an attacker.
- Error-prone software.
- Faulty input data produced by erroneous or malicious hardware or software.

Provisions: Measurement data must be protected against unauthorised altering, deleting and reading. Data is believed to be adequately protected if only legally relevant software produces and accesses it. To achieve this, a dynamic boot process is realised. All software applications are checked for correctness during runtime. This is done by comparing predefined values with effectively occurring values. At the moment, this is a critical problem since the concepts of a dynamic boot of trust environment is hardly realised. Dynamic boot of trust is difficult to achieve because checking software during runtime is complex. There is still no consensus agreed on what exactly is checked when and how often. But since the TMS software is limit and it is exactly known what operations are trusted the dynamic trust concept is regarded as being realisable. For more information see Section 4.6.1.

Threats: This requirement addresses Threat T2 but also considers Threats T3 and T4. Since all system modifications during runtime are documented changes concerning Threats T3 and T4 also can be detected. In addition, T1 is referred implicitly. Changes in the upper level of software might not be detected if a malicious system was booted. Threat T7 and T8 also can be reduced by this requirement.

R3 - Protection against intentional changes.
Legally relevant software shall be secured against inadmissible modification. This requirement is related to Requirement R2. The difference is that in R2 entirely new software is inserted. Whereas, in this requirement modifications that are carried out on the already present software and its input parameters are considered. In addition, mainly attacks on the software of a Traffic Monitoring System are observed. For physical attacks turned against the hardware see Requirement R10.

Reasons for intentional changes:

- Manipulated program code inserted by an attacker. This is done either physically or remotely. For example, the memory is physically removed and replaced with one containing fraudulent software or data.
- Attacker introduces malicious input data or alters input data to change output.

Provisions: Compare to provisions made for R2.

Threats: This requirement references Threats T1 to T4.

R4 - Protection against accidental or unintentional changes.
Legally relevant software and measurement data shall be protected against accidental or unintentional changes. Stored measurement data shall be protected against corruption or deletion when a fault occurs. Appropriate provisions shall be made to protect data from unintentional changes that could arise through incorrect program design and programming errors. Functional defects that can falsify Measurement Values shall be detected as fast as possible.

Reasons for accidental changes and faults:

- Error-prone software.
- Residual defects of the software even though state of the art of development techniques have been applied.
- Unpredictable environmental influences.
- Unpredicted effects caused by user.

Provisions: As already described in R2, the program code shall be checked during runtime for any malicious occurrences. This principle will also help in protecting against accidental or unintentional changes since a fault will produce a different value then it was defined. Thus, the failure will be documented. Additionally, it can be ensured that overwriting of data only occurs in the foreseen data storage period. Deleting measurement data shall cause a warning and can only be done by authorised persons and services. Besides, it must be checked that all TPM processes ended correctly and that the Measurement Records are created correctly.

Threats: This requirement considers threat T8.

R5 - Securing data after creation.

- ***Securing completeness of measurement data.***
- ***Securing integrity of measurement data.***
- ***Securing authenticity of measurement data.***
- ***Secure storage and protection.***

Legally relevant measuring data shall be secured against faulty measuring and processing. Furthermore, it shall be protected against any malicious attacks and unauthorised modifications. Non-repudiation of data shall be granted.

Reasons for inadmissible data:

- No secure storage:
  Data is easily accessible as it is not securely stored. An attacker is able to corrupt the integrity or authenticity of data by inserting fraudulent software. Accordingly, measurement data is tampered what makes it inadmissible as evidence. In addition, the attacker might violate the privacy of data stored. As a result, measurement data stored must be protected against intentional changes carried out with
  - Simple common software tools or more sophisticated software tools.
  - Simple common software tools are understood as tools, which are easily available and manageable as e.g. office packages.
  - Sophisticated software tools are debuggers, re-compilers, software development tools, etc.
- Incorrect stored data: The storage of data might be inadequate. High temperature variations, water, magnetic influences etc. can destroy stored data.
- Incomplete stored data:
  First, the measurement data stored must contain all relevant information necessary to reconstruct the measurement. And second, stored measurement data may be needed for reference at a later date e.g. for checking invoices. All data necessary for legal and metrological reasons shall be stored together with the measurement value.
- Incorrect processed data: Errors occurring during processing cause inadmissibility of data.
- Fixed parameters can be changed easily: Fixed physical variables used for calculations that are changed easily result in a totally different outcome than actually expected. Even small changes in the accuracy of the decimal places can harm the correctness drastically.
- Data stored must be capable of being authentically traced back to the measurement that generated them:
  The authenticity of measurement data may be needed for reference at a later

date, e.g., for checking invoices, for presentations at court, for issuing tickets. Authenticity requires the correct assignment of measurement data to the measurement that has generated the data.

Provisions: Keys for signing and encrypting data are used. Moreover, these keys are bound to system state and authenticity. This protects data secured with these keys from being altered or read by an unauthorised person or application. If the system state and the authenticity of the applicant do not conform to the predefined values keys are not released for usage. This is realised through the protected key storage and usage capabilities inside the TPM. Furthermore, the TPM documents any actions in a Stored Measurement Log (SML). In the SML all changes of data and events are logged. Nothing can be deleted or changed from the SML without being also logged. Each entry is generated automatically by the legally relevant software and contains

- the identification of the parameter (e.g. the name),
- the parameter value (the current or the value before),
- the time stamp of the change.

Aside from this, measurement data is protected by attaching an automatic time stamp on creation (see sections 4.6.2 and 4.6.3). Moreover, measurement data should not be deleted without prior authorisation, e.g. a dialogue statement or window asking for confirmation of deletion are implemented. Rights management can be used. An event counter that is incremented always when parameters are changed can be used to survey events. This counter then can be compared to initial values. And last, if applicable, a flag or label stating whether bills were paid (unpaid) can be utilized. A program would only delete files if invoices had been paid or were out-of-date.

Threats: This requirement especially addresses T4 but is closely related to Threats T2 and T3.

R6 - Securing keys.

- ***Key protection***
- ***Confidentiality of keys***

All keys and accompanying data used for securing measurements must be treated as legally relevant data. Thus, they must be kept secret and be protected against compromise and fraudulent use. No attacker shall ever gain access to keys.

Reasons for corrupted keys:

- Attacker gain access to keys because of
  - unprotected storage.
  - careless use and transmission of keys.

- careless authentication and thus, revealing authentication secrets.
- inadequate protection against intentional changes carried out by common simple software tools or more sophisticated software tools (compare Requirement R5).

Provisions: Hardware protection, by means of a TPM, is used to securely store keys. The secret key is stored in a hardware part that can be physically sealed. The software does not offer any features to view or edit these data. For additional security, the TPM implements a special key hierarchy (cf. Section 3.4.1). In addition, the identity of a system can be concealed to protect the authentication process. For this, AIKs (see section 3.4.3) and certificates can be used. Hardware protection with a TPM is regarded as adequate. After [97] a standard technical solution would not be sufficient to ensure high protection level if there were no appropriate hardware protection means for the keys and other secret data. Hardware protection means in the described sense are for instance housing seals. If the access to secret keys is prevented by sealing the casing of the TMS, no additional software protection means would be necessary [97]. Thus, by using a TPM, a sophisticated hardware protection solution, no additional software should be needed to secure keys.

Threats: This requirement especially addresses T4 but is closely related to Threat T3. And if keys are used for authentication Threat T5 is also contemplated.

R7 - Providing access control mechanisms.
All legally and security relevant data needs to be secured against illegitimate access. It must be impossible for an unauthorised person or application to tamper the contents of data.

Provisions: Access control mechanisms are one means to achieve security. Only certain users and applications are able to read, write and delete data. All others are restricted in their access permissions. As well, access can be controlled through the use of cryptographic keys. Who exhibits the key is able to access data or to carry out a special operations like signing. With the help of a TPM, keys can be bound to a special identity and to a specified system state. Only this special TPM gains access to the keys. The utilization of certificates also ascertains the identity of a unit. Thus, if the identity does not comply with the one described in the certificate access is denied.

Threats: This requirement refers to threat T3.

R8 - Providing secure authentication.
For any kind of transmission processes a secure authentication is needed so that it can be precluded that an unauthorised person or application gains access to legally and security relevant data. Or in general, attacker should not be able to intimidate the authentication and transmission process.

Reasons for corrupted authentication:

- Weak authentication algorithms used for authentication of remote communication parties.
- Weak authentication algorithms used for authentication of user.
- Faulty implemented authentication algorithms which lead to compromise of authentication (e.g. replay attacks).

Provisions: For the authentication of any Traffic Monitoring System the TPM provides a unique identity. This identity cannot be compromised. It just needs to be secured that strong authentication mechanisms are implemented. For instance, a Diffie Hellmann key exchange can be used to securely establish a secret between parties. Then, by using this secret, the parties can be ensured that they are communicating with the right partner. The traffic between the parties is encrypted so that only the party who owns the right key can read it. Certificates can be used to provide evidence that the parties' identities are the ones they claim to be. Since the public keys are linked to an identity with a certificate. Digital signatures can be used to implement authenticity. Furthermore, to combine all these approaches with the TPM functionality special non-migratory keys can be used for transmission. Non-migratory keys are certified to be part of the TPM and cannot be migrated. By using non-migratory keys for signing purposes authenticity is warranted. No other TPM or application can use non-migratory keys but the certified TPM.

Threats: This requirement refers to Threat T5 and T6.

R9 - Providing secure transmission techniques.
Secure transmission is needed to provide privacy, integrity and authenticity of data. Legally relevant software that may get tampered on the transmission process is not anymore useful for evaluation and as evidence. This requirement considers internal transmission as well as external transmission to a remote server. Requirement R8 is closely related to this requirement.

Reasons for weak transmission:

- No authentication techniques are applied.
- No secure transmission channels are established.
- Data is sent in plaintext.

Provisions: Secure transmission channels must be established. Therefore, authentication techniques, as they are described in R8, must be adopted to protect the transmission process. Without authentication no guarantee is given that the obtained data is correct and authentic. Hash values and checksums are attached to transferred data to secure its identity. To ensure privacy of data encryption is applied. For further data protection during the transmission process time stamps are added (see Section 4.6.3).

Threats: This requirement addresses Threat T5.

R10 - Providing a secure hardware environment.
This requirement specifically considers the hardware environment of the Traffic Monitoring System. The instrument needs to be secured against any natural impacts as well as intended destruction of the instrument. This means, all outside effects being caused by man or nature must be obstructed.

Reasons for easily damaged or destroyed hardware:

- Casing is not secured against environmental influences like water, wind, etc..
- Casing is opened easily and not sealed what can cause the instrument to malfunction or break down.
- Casing is easily damaged or destroyed because of wrong casing materials.

Provisions: The casing must be secured during the manufacturing process against environmental conditions. The TMS might be deep-seated temporarily or for a longer period of time outside at one place. That is why most importantly rain and wind should not be able to penetrate the inside of the TMS. For example, special glass fibre reinforced plastics can be used to protect against weather conditions. To avoid destruction of the monitoring system caused through vandalism its casing must be built indestructibly. Against vandalism casings out of aluminium die casting are utilized. As another provision, mechanical or electronic seals are tagged to the casing to avoid unnoticed destruction or opening of the instrument. [32, 59]

Threats: This requirement addresses Threat T9.

R11 - Providing for completeness of data transmitted.
The completeness of the transmitted data must be controlled. Otherwise, it might be possible that complete sets of measurements are not transmitted and get lost. Regular control of the transmitted data helps to check the system performance. Depending on time and location, the Traffic Monitoring System shall generate a special predicted amount of data. If this completeness value is incorrect or is not observed there might be a failure in the system.

Reasons for the incompleteness of data:

- Failures or attacks on the network prevent the system from sending or generating data.
- Failures or attacks on the system prevent the system from sending or generating data.
- Failures or attacks on the sensors prevent the sensor from sending or generating data.

Provisions: For the regular control of the completeness of data sets reference measurements like pictures showing the correct situation are used. A reference value is generated to manually overlook if the TMS is still in the right working order. With a reference value on the one hand sensor failures can be detected. On the other hand, the network connection can be checked. Reference measurements must be transmitted in periodic intervals. If no reference measurement is received the monitoring device must have failed in sending the value. Additionally, a hash value of all collected pictures is generated and stored in a specific PCR. All hash values are concatenated to definitely describe all generated values. The current hash value of this PCR is transmitted as security value with each Measurement Record. Having all the hash values stored it can be determined if values have not been transmitted (see Section 4.6.2).

Threats: This requirement addresses Threat T7.

4.3 Definition of Trust

This work is based on the provision of trust for a measuring instrument and its created data. But what actually is meant by the term 'trust' has not been yet specified. Trust in its usage is defined in various ways. Everyone does have his own ideas about trust and defines trust for himself in a specific way. For that reason, a brief unified impression over trust as used in this work is given up next. Whereas, trust and trustworthiness can almost be used exchangeably.

Trust is defined in [60] as

> 'the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party.' [60, page 85]

A new trust model is the definition of transitive trust. This model especially takes effect in the human daily behaviour. People trust people that they do not know in person. But since they know other people that trust this person they believe in the trustworthiness of the unknown person. This principle is also known as web of trust in cryptography. Person 1 trust Person 2. Person 2 trusts Person 3 and so on. Even though Person 1 does not know Person 4, Person 1 trusts in Person 4 to a certain degree (see Figure 8. The degree of trust decreases with every additional person standing between the two parties. Consequently, this chain cannot be arbitrary extended.

Another definition of trustworthiness was given by Schneider [74]. He defined trust in a more technical influenced way. His definition considers the trustworthiness of machines and not human beings in the first place.

> 'Trustworthiness is assurance that a system deserves to be trusted - that it will perform as expected despite environmental disruptions, human and op-

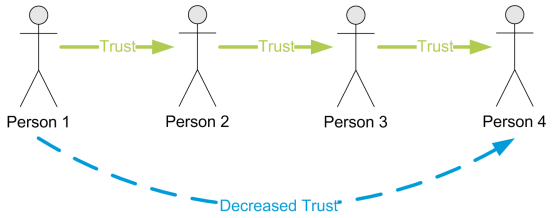

Figure 8: Transitive Trust Model

> erator error, hostile attacks, and design and implementation errors. Trustworthy systems reinforce the belief that they will continue to produce expected behavior and will not be suscepticle to subversion.' [74, page 316]

The TCG defines trust as follows:

> 'Trust is the expectation that a device will behave in a particular manner for a specific purpose.' [89, page 5]

Moreover, they define three principles that must all be true at once to provide trust. The first one is the unambiguous identity which means that an entity to be trusted must unmistakably identifiable. The second principle, unhindered operations, describes that an entity can only be trusted if it acts in a predefined way. This expected behaviour must be warranted at all times. The third principle is the attestation. Attestation means that an entity can be attested to constantly behave in a good manner. For this, the entity must be able to report its system status to a third party. This party then verifies the trusted system state [85].

Trust mentioned in this work is strongly related to Schneiders [74] definition of trustworthiness. The concept allows for the construction of a trusted instrument that carries out executions independently. All results produced by this instrument, the TMS, must be reliable and admissible. Therefore, the system must always behave 'as expected' and must 'not be suscepticle to subversion' [74, page 316]. Likewise, trust as it is used in this work is based on the transitive trust model and the trust model defined by the TCG. The TCG model enlarges the transitive trust model. An entity is trusted if it operates in a predefined way and if its behaviour is not, in any case, compromised. Moreover, trust must be verifiable at all times. Trust in the designed Traffic Monitoring System results out of the realisation of security goals defined in 4.5.

4.4 State of the Art

Regulations for Traffic Monitoring System are among others depicted in various standards. Moreover, the local legislation must be obeyed. This means especially the right of privacy of personal data must be preserved. And since digital evidence just begins to be admitted in legal proceedings the current laws and regulation about the admissibility of this kind of evidence must be observed. This section describes recent standards, laws and regulations considered for building an embedded Traffic Monitoring System.

For this, relevant laws in different countries are observed. Various national and international standards are contemplated. In particular, laws about digital evidence and digital signatures are observed. Digital signatures are one relevant criterion for the authenticity of the evidence and thus admissibility in court.

4.4.1 Relevant Laws for this matter

European Union:

Privacy of data: The German Federal Data Protection Act [29] refers to European directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data (OJ EC no. L281, p. 31 ff.) [15]. Moreover, it defines what kind of data can be collected and how personal data must be kept secret if collected.

Electronic signatures: Article 5 of Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures [17] introduced advanced electronic signatures as admissible in court. Article 5.1 describes advanced electronic signatures. Those are based upon a qualified certificate which is created by a secure signature-creation device. Advanced electronic signature are classified as having the same status for electronic documents as handwritten signatures have for paper based documents. Article 5.2 says that member states shall ensure that electronic signatures shall not be denied legal effect and admissibility as evidence in legal proceedings just because it is in electronic form, not based upon a qualified certificate (issued by an accredited certification service provider) or not created by a secure signature-creation device. For explanation, in [17] is said that the legal recognition of electronic signatures should not only be based on the authorisation of the certification service provider but on objective criteria. Article 3.4, 3.5, 3.6 and 4.2 deal with electronic signature products, secure signature-creation devices and secure signature-verification devices.

Germany: The laws in Germany regarding the privacy of a person are strict. So, the system must not leak any privacy relevant information of a person. All privacy relevant restrictions are defined in the Federal Data Protection Act [29]. Other relevant laws in Germany are the Signaturgesetz [23] and the Signaturverordnung [22]. Both are concerned with the definition and realisation of qualified signatures. First, in §2 of the Signaturgesetz all terms related to electronic signatures are defined. Then in §4 general requirements are depicted. Furthermore, the protection of privacy, qualified certificates and time stamps are classified. In particular, §14 of the Signaturverordnung defines secure time stamping services. In this respect, the used time stamp must conform to this definition. If the time stamp does not comply with the requirements it will be rejected in trial. One of these requirements is, for example, that the time discrepancy of the time stamp-

ing service must be at maximum one minute.

Moreover, the Mautgesetz [24] and several other laws [26, 27, 31, 34] were adopted during the introduction of the toll collection system in Germany. In [24] it is defined what kind of data can be collected and for what purpose it can be applied.

England:

Digital Evidence: In England the Evidence Act 1938 [14] deals with all terms related to evidence. In Cockle's Cases And Statutes On Evidence (10th edn, 1963) [48] real evidence is defined as 'evidence afforded by the production of physical objects for inspection or other examination by the court.' According to Mason et al. [62] this definition was applied in trial. Mason et al. [62] described that Sir Jocelyn Simon P. said in its sentence that the evidence he was observing, respectively the film recording of a radar set of echoes of ships, has nothing to do with hearsay evidence defined in the Evidence Act 1938. But if 'tape recordings are admissible, it seems equally a photograph of radar reception is admissible-as, indeed, any other type of photograph. It would be an absurd distinction that a photograph should be admissible if the camera were operated by a photographer, but not if it were operated by a trip or clock mechanism.' [62, page 176] This sentence opens the legislation for allowing digital evidence automatically produced by machines.

The ACPO guidelines [37] are considered to guarantee good practice when gathering computer based electronic evidence. This guide is a UK document since it points out the differences between the procedures within Scotland and Northern Ireland and England and Wales. These guidelines help investigators with the presentation and storage of digital evidence. A record of all processes applied to computer based electronic evidence should be created. So that an independent third party can verify the results.

> 'It cannot be overemphasised that the rules of evidence apply equally to computer based electronic evidence as much as they do to material obtained from other sources. It is always the responsibility of the case officer to ensure compliance with legislation and, in particular, to be sure that the procedures adopted in the seizure of any property is done in accordance with statute and current case law.' [37, page 5]

The guidelines give a brief explanation of electronic evidence, of investigators, how to seize electronic evidence and the storage after seizure and much more.

Electronic signatures: For electronic signatures England considers the European legislation.

Role of experts: In England and Wales the regulation of expert witnesses is governed by the Civil Procedure Rules (CPR 35) [4] and its associated Practice Direction (PD 35) [11]. Rule 35.3 says that it is the duty of an expert

to help the court within his expertise and to ignore any obligations to the person from whom he has received instructions or by whom he is paid.

Other Countries: Also, in other countries like Australia, Canada, Hong Kong, India, New Zealand, Scotland etc. the considerations about digital evidence, its admissibility and authenticity increased. It exist a variety of laws and regulation concerned with this topic.

The Australian Evidence Act 1995 [28] and the Queensland Evidence Act 1977 [30] deal with evidence in general. In Part 4.3 of [28] evidence produced by processes, machines and other devices and seals and signatures are described. Moreover, it defines the admissibility of evidence in court. Part 4 of [30] explains the role of judicial notice of seals, signatures and legislative enactments. Part 6 Section 92 describes the admissibility of documentary evidence as to facts in issue, Section 93 the admissibility of documentary evidence as to facts in issue in criminal proceedings and Section 95 the admissibility of statements produced by computers [30].

In the Electronic Transaction Ordinance of Hong Kong [20] digital and electronic signatures are defined. Furthermore, it determines that if a law requires a signature of a person or provides for certain consequences if a document is not signed by person a digital signature satisfies the requirements. It must be provided that the digital signature owns a recognized certificate and is generated within its validity.

Mason et al. [62] said that the Indian Evidence Act, 1872 [13] and the Indian Information Technology Act, 2000 [21] define that all electronic evidence falls within the definition of a 'document'. Document in this context means 'any matter expressed or described upon any substance by means of letter, figures or makes, or by more than one of those means, intended to be used, or which may be used, for the purpose of recording that matter.' [21] The IT Act amended the Evidence Act in defining evidence as including all 'documents including electronic records produced for the inspection of the Court; such documents are called documentary evidence.' [21] Section 3 of the IT Act defines how electronic evidence shall be authenticated. It allows only for one method a digital signature.

Noting some Scottish peculiarities, the Requirement of Writing Act 1995 [16] with its subsequent legislation on electronic signatures, the Electronic Communications Act 2000 [18] and the Electronic Signature Regulations 2002 [25] deal with digital evidence and digital signatures. Moreover, the United Nations Commission on International Trade Law (UNCITRAL) adopted on 5 July 2001 the Model Law [92]. This law shall bring legal certainty to the use of electronic signatures and establishes criteria of technical reliability for the equivalence between electronic and hand-written signatures.

In the Sedona Conference [79] the principles for electronic documents were discussed. The Sedona Conference allows leading jurists, lawyers, experts, academics and others to gather and talk about complex litigation in an attempt to advance the law. E.g. in [79] is stated that organisations must properly preserve electronic documents if there is any chance to be relevant to a court. Nevertheless, it would be unreasonable to take every possible step to preserve all potentially relevant data.

4.4.2 Relevant Standards

PTB-A 18.11:

In Germany, the design of any calibrated measurement device needs to conform to the German calibration regulations (Eichordnung EO) and to approved technical regulations. The Physikalisch-Technische Bundesanstalt (PTB) is a national metrology institute which provides scientific and technical services. It publishes different requirements for traffic monitoring devices and other calibrated measurement devices. Those requirements are passed by the PTB, manufacturer and user after reaching a consensus on the needed requirements. One of these calibration conform publications is the PTB-A 18.11 [69]. It explains the necessary requirements for the approval of speed monitoring devices in Germany.

The following paragraphs are summarized excerpts out of the whole document. §3.5.3 - §3.5.6 describe the recording device and the documentation requirements in a speed monitoring device. §3.8 describes the software requirements and the data transmission via interfaces. The transmission of measurement data used for official purposes have to conform to the requirements specified in WELMEC 7.2. §3.10 describes the provisions for the protection against manipulation. Devices shall be secured against easy tampering and destruction. Also, the document refers in this paragraph to WELMEC 7.2.

WELMEC 7.1, WELMEC 7.2 and WELMEC 8.0:

WELMEC is the European cooperation in the field of legal metrology. The aim of WELMEC is to impose a coordinated and consistent approach to European legal metrology. 34 countries are represented on the WELMEC committee and its members are delegated national authorities accountable for legal metrology in the European Union and European Free Trade Association (EFTA) member states [98].

There is a range of requirements for measuring instruments of all kinds in each country but there appears to be great reliance on the WELMEC 7.2 standard [97]. This document describes the application of the Measuring Instruments Directive (MID). Especially, it is concerned with assembling requirements for software-equipped measuring instruments. WELMEC recommends this guide 7.2 for the development of software controlled measuring instruments subject to the MID. As all notified bodies accept this guide as compliant with the requirements contained

in the MID, following the guide results in satisfying all MID requirements [97]. WELMEC 8.0 [96] is another, more general document which gives assistance to all those in charge of the implementation of the MID. WELMEC 7.1 [95] develops software requirements for measuring instruments.

OIML D 11 and OIML R 91:
The aim of the International Organization of Legal Metrology (OIML) is to support the global harmonization of legal metrology. The OIML is an intergovernmental treaty organisation and its members are countries which are involved in technical activities. The OIML provides guidelines for the development of requirements regarding the conception of measuring instruments for legal metrology applications. The OIML provides regulations and recommendations which ensure that products meet the national and international legislation. OIML cooperates with certain institutions, such as ISO an IEC, to prevent the creation of contradictory requirements [9].

OIML D 11 [67] is a general document about the requirements of various measuring instruments. OIML R 91 [66] is especially concerned with radar equipment for the measurement of the speed of vehicles. §2.6 describes the requirements of the recording device and what needs to be documented. §3.4.2 provides requirements related to the electronic failure and especially about the permanent storage of data. §5 describes the protection needed to avoid tampering and §6 describes what is needed to identify a device.

4.4.3 Trusted Computing Group

The Trusted Computing Group develops standards for hardware-enabled trusted computing technologies. The functional integrity, privacy and individual rights should not be harmed implementing TCG computing environments. The main goal is to protect user information like password, keys and data from unauthorised access. The TCG designs and still develops standards regarding TPMs (see Section 3 for comparison). The currently available standard is the v1.2 TPM specification [87–90].

4.5 Concept

As described in Section 4.1 an embedded device is created to assemble Measurement Values (MVs) for traffic analysis. This device shall grant a secure environment for the collection of this measurement data so that it can be used as digital evidence in court. The constructed environment will process and secure data that it meets the legal requirements of a locality. In order to meet legal standards, Trusted Computing is applied, namely the functionalities of a Trusted Platform Module (TPM) are employed.

Digital evidence and its admissibility was already described in Chapter 2. But Mason et al. [62] stated in their book some assumptions matching with the developing concept. They supposed that digital evidence can be sorted in a number of categories.

One of these categories complements our understanding of digital evidence. It supports the idea of creating digital evidence in an entirely automated process with no human interaction.

> 'Records generated by a computer that have not had any input from a human. Examples of such records are data logs, connections made by telephones, and ATM transactions. The main evidentiary issue with such records may be to demonstrate that the computer program that generated the record was functioning properly at the material time.' [62]

Apart from the mentioned category, the challenges digital evidence can be exposed to are also described in [62]. Applicable challenges are the altering, the manipulation, or the damage of records between the time they were created and they appear in court as evidence. In addition, the reliability of the computer program generating the records might be queried.

A safe system environment is required for the accumulation and processing of measurement data. The collected Measurement Values will prove an event so that it is indisputable that it happened like it was documented. The accused shall not be able to deny any validity because of wrongly measured or faulty documented data. Thus, an environment must be created in a manner that detects and reports any alterations done to it and its created evidence. In this purpose, the integrity of the TMS must always be granted. Security of the TMS is achieved through Trusted Computing. To provide a correctly running system from any faulty or malicious alteration the system status must be checked at all time. This must be done during the start up phase and during the runtime. Alterations are detected in comparing trusted pre-defined system states to the currently running environment status. All trusted system states are defined before processing starts. A running system never shall diverge from any of this trusted states. The system status or configuration is defined in [64] as all executed instructions including processes which are currently present in the OS storage as well as processes which have been carried out since the start up of the system.

The secure environment has to offer certain means to report the status of the system. As integrity of the system needs to be proven at all times. In this case especially, the provability of the system configuration is relevant. As it might be necessary to prove the integrity to an external entity such as a reviewer or an expert at court. First, the system status strongly depends on the boot process. A system that is already booted into a malicious state cannot yield adequate trust in a court of law. Therefore, a trusted boot process is required. Second, undetected alterations to the running system status may lead to faulty or malicious running applications. So the whole functionality of the monitoring instrument can be harmed. Certainly, protected means are required to securely store the representation of the system state. If a system can be changed without detecting the change, it cannot be considered as trustworthy but probably will be. This is the worst case and must be prevented at all times. Cryptographic keys can be used to help protect the systems' critical data. Whereas, the creation, usage and storage of cryptographic keys carry some risks. If the keys are compromised the TMS

is not considered trusted. Since the value of evidence strongly depends on the level of protection of these keys. Accordingly, a secure environment must also deliver protected key management to avoid any compromise.

This concept is an approach to create an autonomous device collecting digital evidence without the formerly described risks. The system shall be able to provide authenticity and integrity of data by its own without human interaction in the main processes. Related to the admissibility of digital data the system must be in a trustworthy state. If the device cannot be trusted its generated data cannot be trusted either. Thereto, the system uses a variety of security techniques like digital signatures to provide for trustworthiness. In general, digital signatures are used to determine the identity of a person. Nevertheless, digital signatures are used in this concept to determine the identity of a device. This and the abovementioned security features are feasible through the utilisation of a Trusted Platform Module (TPM). A TPM is embedded in the system environment to process measurement data so that it can be secured by legal standards. The TPM serves as a security anchor for the environment and delivers relevant security mechanisms. Such as, using digital signatures, hash values, trusted boot functionality, time stamping and the secure storage functions of the TPM. A TPM is considered to always be trusted as it complies with all security requirements posed by the Common Criteria to achieve the security level of EAL4+. All functions carried out by the TPM are believed to be tamper resistant. So, the TPM is understood as not vulnerable and qualified as security anchor.

Architecture and Processes

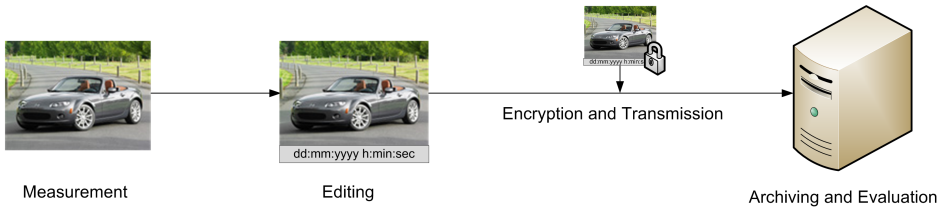

Figure 9: Concept

The concept consists of four main processes: measurement, editing, transmission and archiving. Figure 9 illustrates the processes of this idea. The main purpose of this work is to discuss the editing process since it is of interest to introduce Trusted Computing as solution for securing digital evidence in an automated process. Editing is the main process where the functions of the TPM take effect and measured data is secured. All other processes are also needed but are negligible at this stage. For the sake of comprehensiveness, some considerations about the measurement, transmission and archiving processes need to be briefly explained.

The system is composed of two main entities which are responsible for handling the mentioned processes. The first entity is the Traffic Monitoring System (TMS) itself. And the second entity is the Archiving and Evaluation Unit (AEU). The TMS is responsible for the execution of the measurement and the editing of data. The archiving process, on the other hand, occurs as part of the AEU. The transmission process serves as communication between these two units.

The creation of the Measurement Values is called the measurement process. The TMS consists of one or more sensors which create Measurement Values. The sensors can either be inserted in the TMS or be situated outside as a stand alone device. Since this is a theoretical concept, no more details about the measuring sensors are known. For this reason, it is assumed that all measuring values are generated and transmitted correctly at any time. No further considerations will be taken to prove integrity and authenticity of data at this point. The MVs are the input values for the editing process. The main purpose of this process is to make sure the MVs conform to all legal requirements. This is done in combining the MVs with additional security information. It is general practice to furnish MVs of TMS with additional information [69,80]. That is the reason why this process is called editing.

The processed MVs are then transmitted to an AEU. The transmission process, in brief, presents a general transmission procedure with all its security problems and requirements [97]. At this point, it is optional to provide data encryption to realise privacy with the transmission of data. As at any point, the TPM is utilized for transmission in the remote attestation procedure. With remote attestation the authenticity of the device communicating with the AEU is proven. Hence, this entity can be sure to communicate with a trusted system.

The AEU is mainly concerned with the archiving and evaluation. This component is in charge of the permanent storage of the processed Measurement Values and the detached final evaluation. So, the AEU must provide enough storage capacity to permanently store all produced data for a later purpose [96]. Also, the storage environment must comply with several requirements so that unintentional and intentional damage can be prevented [62,97]. Moreover, it provides the TMS with system updates.

In this system architecture, the AEU is found outside the TMS as self-contained unit. This is done out of several reasons. One of the reasons is trustworthiness. The component responsible for the evaluation must be trusted at all times. If the evaluation of the system status information would be done on the TMS itself then it must contain an evaluation component. If the integrity of the TMS is compromised it is likely that the evaluation part is also compromised. The evaluation component would not detect malicious changes since it would not work properly. This is not acceptable for the design of a trusted system environment. Therefore, the evaluation of the measurement data and the system configuration is located outside the TMS. This adds another security level because not only the TMS but also the remotely located AEU must be compromised (compare to [64, page 74]). Another reason is the size of a traffic monitoring

device. In general, those measuring devices should not be bulky. Consequently, the storage capacity of such devices will be limited as only the most important components are included in the device. Some of these devices may be used at locations difficult to access. It is not convenient to have the measurement data stored in these places since it must be retrieved periodically. Therefore, the data storage is sourced out and the measurements must be transmitted to the remote AEU.

Nevertheless, the tasks of both units cannot be separated. Without measurement data, produced at the TMS, no evaluation of this data could take place. Without evaluation the Measurement Values would be useless. In summary, the architecture of the system consists of two components the TMS and the AEU. Both components are responsible for compilation of processes that later result in a valuable outcome. The TMS embeds the measurement sensors and the TPM. It performs all tasks related to the processing of MVs. The AEU is built for storage capacity and high performance so it is used for storing and analysing the measurements.

Security Goals

Depending on the use case and the requirements, the security goals are associated either with the measuring or the processing of data. As any details about the measuring sensor are known and the correctness of the MVs is assumed the conservation of digital evidence is located parallel or after the creation of data. Figure 10 illustrates the system processes with assigned security goals. Some of these security goals are classified to belong to the protection of data, others to belong to the protection of the device.

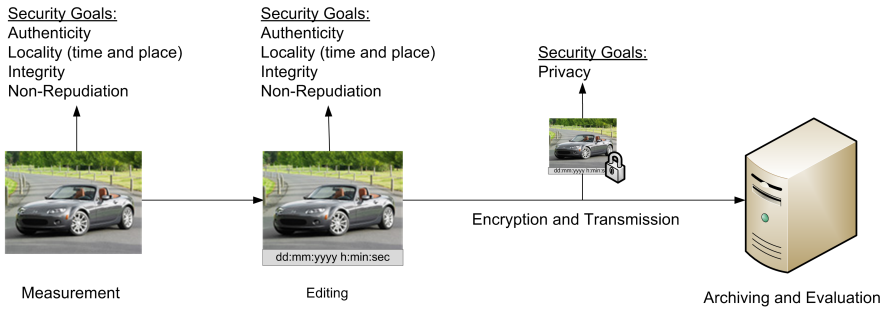

Figure 10: Concept with added Security Goals

For the automatic collecting and presenting of digital evidence in court, security goals have to be warranted to achieve significance. Otherwise, the collected measurement data will be inconclusive as evidence [62]. The measurement data must be collected in such a manner that later its trustworthiness cannot be disputed. Consequently, the overall security goal is to provide ***non-repudiation***. Non-repudiation is not one autonomous security goal but several security goals comprising integrity, authenticity, time and optionally privacy of data. Furthermore, the location, integrity and authenticity of the system must be granted to assure non-repudiation. If the system is not seen

as trustworthy it cannot generate trustworthy data. All of these security goals must be respected to warrant non-repudiation of digital evidence. Regarding the missing security goals - confidentiality, availability and accountability - they are not omitted but not considered as foremost security goals. Nevertheless, depending on the system design, these goals can be included. Non-repudiation serves to build a reliable chain of evidence and makes digital evidence plausible in trial. Below, all security goals are described in more detail.

Integrity is the protection against unauthorised and unnoticed data modification or destruction. In [71] digital integrity is defined as 'the property whereby digital data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source' [71]. Integrity is one of the key goals in securing digital evidence since the data needs to be trustworthy. There should arise no doubt that data has not been tampered with or altered after it was generated. In this design of a TMS the documentation of Measurement Values shall be of integrity. At least, since the documentation is passive and there is no active intrusion prevention, it must be impossible to tamper measuring data undetected. Especially, the requirements R4, R5 and R11 explained in 4.2.2 specify the need for integrity of data. Additionally, the systems' integrity must be guaranteed, no alterations of the system software or hardware shall be possible. If an unknown component is executing the trustworthiness of data cannot be warranted. The system must be protected from unintentional, unauthorized, or unanticipated modification. Requirements R1 to R4 are responsive to the integrity of the system since they define how the system must be constantly checked. Likewise, the requirements R7 to R10 help to protect the integrity of the system.

Authenticity is the attestation of the validity and reliability of an identity. This is another key goal in acquiring digital evidence and presenting it to a court. An object or human being should be identified definitely and correctly. Especially in a court of law, since it must explicitly be proven that an entity took action. In this case, the entity can either be an object, the TMS, which generated data, or a person who committed an action. The TMS shall be able to authenticate itself to another party so that the data it provides can be seen as authentic. Data must be identified to be belonging to one specific TMS. A third party must be able to verify that the data has been generated by a trusted source and has not been modified in transit. The requirements R5 to R9 dwell on the authenticity of the TMS. If no secure authentication techniques, access control mechanisms and transmission techniques are implemented the system or the communication partner cannot be assumed to be authentic. As well as if data or keys are compromised the authenticity of the system and its produced data gets lost.

Time is the reliable time stamping to ensure that an event definitely happened at a specified time and to assure that data has not been altered since the time it was created. In particular, requirement R5 denotes how data must be secured after creation. This requires also time stamps.

Location is the reliable determination of the location of an entity. It must be dis-

tinctively resolved where an event happened. Any change in the location value can lead to inadmissible data. Guo [54] said that 'location accuracy is of utmost importance [...] due to the fact that accuracy of any information on accidents and other traffic conditions is incredibly important for the safety and efficiency of roadways.' [54, page 2] The location as well as time stamp contribute to the protection of the datas security which is concerned in requirement R5. It must be possible to authentically trace back to the measurements. The documentation of the location helps to build up chain of proof.

Privacy is the protection of personal data. No personal data shall be acquired by people who are not allowed to view or collect this data. In every country, personal data is secured differently by law. Germany, for instance, has strict requirements about securing personal data. The Federal Data Protection Act serves to protect the rights of people considering privacy [29]. Depending on the actual realisation of the TMS the privacy of data is essential, e.g., in the case of toll collection (for some related information see [1, 12, 81]). In Germany several new laws have been adopted for the introduction of a toll system defining what exactly can be documented (see e.g. [24,27]). All these points approve that privacy of data in most cases will be a big target. All requirements described can be seen to at least implicitly protect the privacy of data. But above all R5 and R9 care for the privacy of data as both demand for encryption of privacy relevant data.

Confidentiality is the protection against the unauthorised acquisition of information. Losing confidentiality means the unauthorized disclosure of information. This point is more general referred to data on the whole but closely related to the privacy security goal. Again, all requirements can be seen to protect confidentiality at least implicitly since any data can be acquired during any failure of the system.

Availability is the protection against unauthorised impairment of the functionality of a component or system. For example, a system shall be protected against Denial of Service attacks. It is desirable to secure the acquisition of digital evidence as there are no gaps in the documentation process. The system must be available on a timely basis to meet requirements and to avoid considerable losses. In this case, the requirements R1, R2 and R10 specially address the problem of availability of the system. R10 requires for a secure hardware environment to protect the system against outside influences. Whereat, R1 and R2 amplify the inside of the system requiring for documentation of the current system status.

Accountability is the protection against people denying an action like having signed an electronic contract. Guo [54] depicted that 'the system must have the ability to attribute actions to the entity that caused those actions, in case of conflict' [54, page 2] to be accountable. In this case, the difference of non-repudiation and accountability is defined as that accountability is termed to be related to a person. On the contrary, ***non-repudiation*** is a more general term than accountability and also includes the accountability of devices. Accountability, adapted on this case, can only be seen as to

the extent that the produced digital evidence distinctively identifies a person acting in a specified way. Assuming this, accountability as well as non-repudiation is tried to protect through all requirements described in Section 4.2.2.

Non-repudiation of the measured data is achieved by designing a secure environment which meets the defined security goals and requirements and by adding additional security information to the measured data. First, the environment is designed to allow for an appropriate security level. Secondly, during the data editing process, a procedure adds supplementary security information to the measurement data. The procedure is based on the security assumptions given by the secure environment. As additional evidence the following is logged:

- The identity of the device.
- The location of the device.
- The time when a Measurement Value or a set of Measurement Values was recorded.
- And the status of the system so that the integrity of the running system can be verified.

Hereafter, the MVs plus added information are called Measurement Record (MR) in the rest of this document.
In the end, to add another security level the Measurement Records are signed. This adds authenticity to the processed data. Each measurement record is signed by a key created and used by the TPM. Then, the data is ready for transmission to the archiving and evaluation unit (AEU).

Security Assumptions

The design of the Traffic Monitoring Systems (TMS) considers the TPM as security anchor. For this reason, if the TPM is compromised the whole system design would be obsolete. Nevertheless, there are also other attack points on the system. This conceptual model of a TMS does only concentrate on the editing process. That is the reason why some attack points are neglected since they are out of the scope of this concept. Following some security assumptions are summarized:

- A secure environment is offered by the manufacturer. That is, especially, physical attacks and the attacks mentioned in Section 3.6 are not applicable.
- Sensors measure and transmit data correctly i.e. integrity, authenticity and privacy of Measurement Values are guaranteed.
- The storage capacity of the device is considered to be adequate high to store all accruing data.
- The AEU stores transmitted data correctly and in the correct environment. So that, no damage or compromise can occur.

- The AEU enforces all security requirements. It can be assumed that this part is not compromised.
- The transmission of MRs is adequately secured to confirm with legal requirements.
- The TA enforces all security policies required to provide a certified time stamp. No compromise of this unit is assumed.
- The PCA enforces all security policies required to provide a legally relevant certification service. No compromise of this unit is assumed.

These assumptions define the scope of the designed TMS more precisely. Security risks arising to or out of these parts are neglected. With these assumptions the scope is narrowed to the editing process.

Advantages and Restrictions

There are several advantages of this TMS concept. One advantage is that with a TPM an adequate hardware based security can be established which is not vulnerable to environmental conditions. Moreover, the TMS is able to attest its system configuration to a third party. The status of the system is one basic function which is new for this kind of devices. For example, keys can be bound to the system state. Meaning if the source of the MVs is not trusted or another component is not in a trustworthy state signatures are not carried out. This is one main difference to a smartcard which also provides hardware protected signatures but does not check for the authenticity of the signed data. If the authenticity of data is not warranted the system must not sign data as this can be foisted by an attacker. Also, the trustworthiness of the system can be checked some time after the creation of MRs and must not be checked during the creation. This is because the current system configuration is included in the MRs. Another advantage of this concept is the time stamp. As described in the following Section 4.6.3 a certified time stamp is provided and linked with the internal time source of the TPM. This independent time stamp is of important significance since it can be traced back to its source and cannot be broken through its signature. It must be explicitly noted that the designed TMS is restricted to the documentation of its configuration state be it correct or malicious. The TMS is not designed for an active intrusion prevention. Hence, the evaluation of the results must be done at a later point. If any malicious or faulty system configuration is reported by the TMS the AEU must determine it and arrange for changes.

4.6 Solutions

Following in the next sections, some of the previously mentioned solutions, for the conception of a secure Traffic Monitoring Systems producing admissible evidence for legal proceedings, will be presented in detail. As a basis and security anchor will serve the already mentioned TPM. All presented solutions build upon the security functionalities a TPM provides. In Section 4.6.1 a detailed description of a static and dynamic trusted boot process is given and some effective realisation concepts are described. Section

4.6.2 illustrates how additional security information is collected and added to the current Measurement Values forming a Measurement Record. The Measurement Record shall provide all necessary information for the admissibility of the produced Measurement Values at court. The contents of a Measurement Record, the acquisition of the required security information and the signing operation on the Measurement Record are thoroughly described. In Section 4.6.3 the acquisition of a real time value is described. This value combined with an internal TPM time stamp is used to determine the exact time of the MV generation. Section 4.6.4 briefly describes the background of the transmission process. The last Section 4.6.5 collects some TPM functionalities which are also considered to be useful but not yet in a stage of completeness.

4.6.1 Trusted Boot

This section comprises additional information about the trusted boot process (see Section 3.2.3). In this design, the trusted boot procedure offers us the possibility to provide a basis for the trusted measurement of data. It is used to create a foundation for a trusted environment. With this kind of integrity check at the beginning of each reboot, it is trustworthy reported if a malicious or faulty system was booted. If the integrity check of the start-up phase is passed the running systems integrity needs to be measured. Since the measuring device will be used for traffic analysis it is likely to be located outside. As a permanent used TMS it will not be under continuous personal surveillance. This makes it easier for attackers to explore the conditions. On the other hand, TMS can be located above frequented roads out of physical reach. This could pose difficulties for some attackers to physically insert malicious code or even hardware. Thus, an attacker might use an unnoticed remote insertion of fraudulent data. To prevent this, the executed applications must be checked for integrity via a dynamic extension of the chain of trust. The TPM continuously stores the measured integrity values so that they can later be attached as additional information to the Measurement Record. Even thought, an attacker would be able to replace functionalities physically or remotely this would be documented on a system deploying the trusted boot concept. And consequently, steps can be taken against the compromise and to restore trusted conditions. In first instance, the traditional trusted boot concept is briefly explained. Then, the expressions static and dynamic trust are described in more detail. And last, some solutions and approaches are given that try to implement the trusted boot problem.

The traditional trusted boot process as defined in the TCG specification [89] provides means to reliably test the trustworthiness of a starting system. The system shall never be booted from a malicious or faulty basis. In that respect a chain of trust based on the transitive trust concept must be established. The starting point of the measurements is called the Root of Trust for Measurement (RTM). The trusted boot process using a TPM starts with the CRTM. The CRTM represents the root of the chain of trust. This component is always been trusted with no limitations. Then, the CRTM measures the next component and stores the integrity values in a PCR. Afterwards, the control is handed to the measured component. This is subse-

quently done with all components responsible for starting the system. To avoid tampering in any way the integrity values are stored in the PCRs in a concatenated form ($PCR[n] \leftarrow SHA-1(PCR[n] + measureddata)$). So, any alteration can be detected. For comparison, information about the trusted boot process were already given in Section 3.2.3.

It is applicable to divide the trusted boot process in two parts - a static and a dynamic part. This division is also described in [89]. There are the terms Static Root of Trust for Measurement (S-RTM) and Dynamic Root of Trust for Measurement (D-RTM) introduced. It is noted that the S-RTM 'begins measuring from a well-known starting state such as a power on self-test.' [89, page 6] Whereat, the D-RTM 'transitions from an un-trusted state to one that is trusted.' [89, page 6] Müller [64] picked up a similar terminology in his book. He used the terms **'static chain of trust'** and **'dynamic chain of trust'** [64]. Both terms describe a trusted boot process instead of only defining the nature of the root.

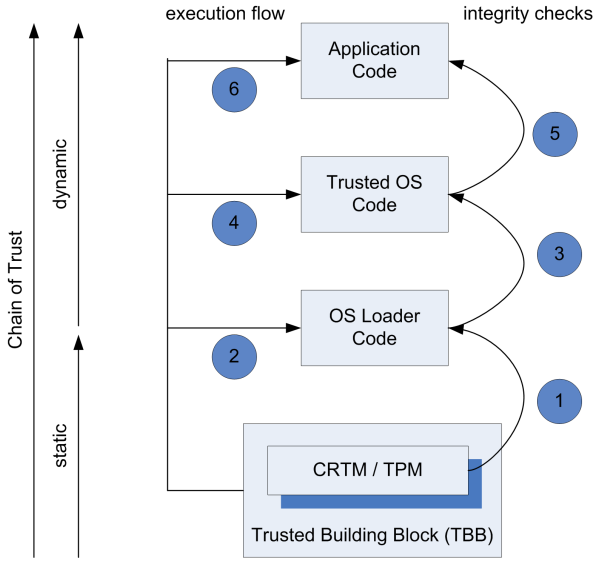

Figure 11: Chain of Trust [64]

The static chain of trust depicts the part of the chain of trust which is realised by the TCP. The realisation is exactly defined through the specifications of the TCG. A TCG compliant platform, i.e. a TPM, must be manufactured in obeying the regulations defined by the TCG. Thus, the static chain of trust is identically realised to the TCG regulations and independent of any operating system. The static chain of trust comprises the integrity check of all components from the CRTM up to the Master Boot Records (MBR) code. That means all components which participate in starting the static part of the system [64]. The allocation of PCRs for the static root of trust for measurement is illustrated in Table 1.

PCR Index	**Alias**	**pcrReset**
0-15	Static RTM	0
0	CRTM, BIOS and Platform Extensions	0
1	Platform Configuration	0
2	Option ROM Code	0
3	Option ROM Configuration and Data	0
4	IPL Code	0
5	IPL Code Configuration and Data	0
6	State Transition and Wake Events	0
7	Reserved for future usage	0
8-15	unspecified	0

Table 1: PCR usage of static root of trust for measurement as defined through the TCG [76]

PCR Index	Alias	pcrReset
16-23	Dynamic RTM	1
16	Debug	1
17	Locality 4	1
18	Locality 3	1
19	Locality 2	1
20	Locality 1	1
21	TOS Controlled	1
22	TOS Controlled	1
23	Application Specific	1

Table 2: PCR usage of dynamic root of trust for measurement as defined through the TCG [76]

In contrast, the dynamic chain of trust is realised through a so-called trusted OS or a comparable entity. The last component of the static chain of trust forms the security anchor for the dynamic chain of trust. In general, this will be the executable part of the MBR. The dynamic chain of trust implies everything from the MBR up to the application code. The PCR usage for the dynamic chain of trust is depicted in Table 2. Since especially checking the application code for its trustworthiness is difficult the dynamic chain of trust concept grows complex. Shall the application code be checked before runtime or in discrete intervals? Or is it better, since malicious code can also be loaded during runtime, to constantly check the running code? Constant checks at runtime probably would be the best answer to be positively convinced that the system is running correctly. But measuring all applications during runtime is performance intensive. Thus, there exists no consensus on what exactly needs to be reported to provide a reliable and trustable dynamic chain of trust. Even, the TCG has not yet defined any exact concepts or approaches for this part of the chain of trust. To continue the chain of trust in the sense of Trusted Computing, all steps carried out since the control was given to the MBR must be checked. This means, each step must be enhanced for integrity checks in the form of hash values. Therefore, already the MBR must be measured before control is given to it. Then, the MBR must measure the OS Loader before control is given to it, etc. When the OS Kernel was measured and passed control, applications and services are started that must be measured. This presents a challenge as the reporting must be continued during runtime to reliably document the system status. Today, operating systems are capable of multi-tasking meaning that multiple applications can be executed coincidentally. Up to now, the integrity checks were done before the code was executed. But measuring the code of multiple applications before runtime is slightly effective. This is because measuring multiple applications takes too long and it cannot be guaranteed that they were not altered during runtime [64].

Possible realisation concepts

Grand Unified Bootloader (GRUB):
GRUB provides a flexible architecture with multi boot standard. That is it is able to boot different operating systems like Linux or Windows. GRUB is composed of multiple parts. Each part is executed consecutively. The first part, also known as stage1 is located in the boot sector and is responsible for configuring important settings. The second part, or stage2, is the major part of GRUB. stage2 accomplishes the tasks of initialising the system and loading the operating system. GRUB owns a configuration file menu.lst. This contains all parameters essential for booting an OS. The configuration file provides the basis for a boot menu. The boot menu performs following functions [75]:

- The user can choose configuration settings for the boot process.
- The user can boot different operating systems.

TrustedGRUB:
TrustedGRUB is an enlargement of the previously described GRUB. Its current version is 1.1.3. It implements the security mechanisms provided by a TPM. The implementation of TrustedGRUB is compliant with the TCG specification [75]. It is the best known and the most sophisticated realisation of the continuation of the chain of trust from TCP up to the entire start of the operating system [64]. TrustedGRUB was developed to offer a multi boot capable boot loader for the Turaya security framework [75]. Nevertheless, it provides the ability to boot all OS that were formerly supported by GRUB. Aside from this, the owner gets the opportunity to define arbitrary files which then will be checked for integrity. For the realisation of the trusted boot concept TrustedGRUB measures different components. These components include relevant parts of its configuration file, an optional list of files and the multi boot modules which need to be loaded. TrustedGRUB uses a Static Root of Trust for Measurement. It establishes a chain of trust which is started by the Core Root of Trust for Measurement (CRTM) and the TCG enlarged BIOS. As GRUB TrustedGRUB has two parts, stage1 and stage2. stage1 is enhanced so that it measures the first sector of stage2. For the realisation of the trusted boot all of stage2 required inputs must be measured. This comprises e.g. the OS kernel and multi boot modules. All measurements of modules to be loaded are stored in PCRs 4, 8, 9, 12, 13 and 14 (Compare to Table 3.). In addition to the functionalities provided by GRUB, TrustedGRUB offers password checks, checkfile and SHA-1 calculation [75].

Integrity Measurement Architecture (IMA):
IMA is an IBM Research software architecture and implementation developed for Linux. The system does not rely anymore on the trustworthiness of the software. IMA builds on integrity protection using a TPM hardware extension. It uses the TrustedGRUB boot manager to load the Linux kernel [64]. So, it provides verifiable evidence about a measured system. A verifier establishes trust into the measurements. Through the measurements trust is established into runtime properties of the measured system. A measurement is performed in computing a SHA-1 hash value over a file. This value is

PCR Index	Usage
4	MBR information and stage1
8	Stage2 Part 1
9	Stage2 Part 2
12	Command line arguments
13	Checkfile measurements
14	Kernel and modules

Table 3: PCR usage of TrustedGRUB [76]

added to a measurement list. IMA maintains this measurement list which covers all executable content loaded since the boot of the system. And then, a hash value of the list itself is inserted into a PCR of a TPM. So, it is possible to link each component with a separate checksum. The limited PCR storage capacity can be circumvented and additional information like the file name or a time stamp can be documented [64]. First, IMA documents the measurements taken during the boot process. And then, it continuously measures itself executable content that is loaded during runtime and sensitive files. The measurement list is reset during a reboot. Since all PCRs are reset automatically. Summarized, it is to be noted that illegal behaviour of the system is documented but the compromise might not be prevented. However, the contents of the measurement list are protected and cannot be changed unnoticeably [6].

GRUB-IMA:
GRUB-IMA is a Trusted Computing enlargement of GRUB. The implementation is totally compliant with the TCG specifications. It was developed by IBM for its Integrity Measurement Architecture (IMA) described in the previous paragraphs. GRUB-IMA like TrustedGRUB measures all components of GRUB and all operating system components which need to be loaded. The hash values measured are stored into the PCRs of a TPM and additionally, to the ACPI measurement log. There different event types are utilized (see Table 5). The measurements taken are saved to the PCRs 4, 5 and 8 (see Table 4). GRUB-IMA provides some new commandos and functions for the GRUB command line and the GRUB configuration file. These commands are the 'tpm' and the 'measure' command. GRUB-IMA uses a static Root of Trust for Measurement.

BIND – Binding Instructions aNd Data:
BIND as noted by Shi, Perrig, and Van Doorn [77] is a 'fine-grained attestation service for securing distributed systems.' [77, page 1] BIND was developed to advance attestation of running software. However, at the moment it is still only a theoretical approach [64]. This exactly addresses the dynamic chain of trust problem. Since there is such a variety of software the integrity checks with hash values is complicated. Additionally, Shi et al. [77] depict another problem. They say that the time-of-use and the time-of-attestation discrepancy are too high to be assured that the code was not

PCR Index	Usage
4	MBR information, stage1 and stage2.
5	ACTIONs (here: EventLog).
8	OS components (kernel, initrd, module, measure-command).

Table 4: PCR usage of GRUB-IMA [75]

Event-type	Discription
0x1005	SHA1 digest of GRUB ACTION message.
0x1105	SHA1 digest of kernel commandline strings.
0x1205	SHA1 digest of kernel image.
0x1305	SHA1 digest of initrd image.
0x1305	SHA1 digest of module image.

Table 5: Event types of GRUB-IMA [75]

compromised. During attestation the code might have been correct. Nevertheless, it is possible that compromise took place before or during usage.
BIND tries to realise following three principles:

1. Fine-grained attestation meaning only critical code sections are attested.

2. Narrowing the time-of-use and the time-of-attestation discrepancy meaning attesting the code immediately before its execution.

3. Binding the code attestation to the data produced meaning that it can be exactly determined what code produced what data.

Fine-grained attestation on the contrary to coarse-grained attestation considers only this piece of code which is relevant for the current execution. The critical code section is marked with some starting and completion flags (ATTESTATION_INIT and ATTESTATION_COMPLETE). Then an 'attestation service' [77, page 2] is invoked which attests the integrity of the designated code. In coarse-grain attestation all running applications are examined. This leads to the problem that already the tiniest difference in the executed code results in a different hash value. For realising the second principle, to reduce the discrepancy, Shi et al. [77] attest the code immediately before execution and implement a sand-boxing mechanism. The sand-boxing mechanism secures the execution of code. This should make it possible to even execute a trusted piece of code on an untrusted compromised system. Binding the code to the produced data is realised through cryptographically attaching information to the produced data. This information is somewhat integrity proof of the code. So, it can be

determined exactly which code was executed to generate the data. Furthermore, the integrity information of the input data is attached to the integrity statement of the code. Thus, a transitive trust chain is established. And it is only necessary to prove the integrity of one piece to prove the integrity of the whole chain. [77]

For the whole concept to be working it is essential that the attestation service is not compromised and that the sand-boxing mechanism is reliably implemented. For this, BIND assumes some properties like a secure kernel, a hardware supported secure environment for the realisation of the sand-boxing mechanism and a TPM with good performance for calculating and storing hash values. BIND relies on the integrity of these components. Consequently, as Müller [64] described, BIND is to be comprehended as attachment to IMA or TrustedGRUB. As Shi et al. [77] discussed BIND still does not present a fundamental solution. It does not deal with attacks on the code inside the labelling. But Shi et al. [77] assume an adequate solution because even a sophisticated attacker has a restricted attack interface.

4.6.2 Measurement Records

From a legal perspective, with any mechanism implemented, the operator of a measuring instrument must ensure that any data must be admissible if used as evidence in a court of law. Since the evidential integrity of digital data must be ensured the collected data must be secured against any tampering. In a traditional way, this is done with so-called measurement logs or records that save details about the measuring. A measurement log must contain different values like the device identification, beginning function checks, half hourly function checks, beginning and end measurements, the name of the person carrying out the measurement, and the current time. To mention some requirements for measurement logs, [66] defines for radar equipment that the log must contain 'the date and time of the measurement, the measured speed and the vehicle's direction of travel.' [66, page 4] If using a camera the log must enclose the correct association between the direction of radiation and that of the optical axis of the camera. Moreover, all device checks must be included in the log [66]. [69] describes that in every picture or sequence of pictures the date and the time in a resolution of seconds must be superimposed. The traffic situation must be documented so that no faulty association occurs. [69] depicts a variety of documentation methods and its requirements as well as the requirements on the device in general. What exactly is needed is subject to the local legislation and the actual purpose of the device.

Considering newer documenting techniques, especially digital pictures, they must be secured separately to provide for authenticity. Cottingham, Beresford, and Harle [50] noted that if digital pictures are taken they require digital signatures at the point they are taken. Transport for London [82] said in their report that especially, the evidential integrity of digital pictures must be provided. Digital pictures are easier to alter than their analogous part. Consequently, any digital techniques used must satisfy the legal requirements of a court of law. It must be proven that digital data has not been corrupted deliberately or not. [82]

To accomplish this purpose another processing step is added to the creation of measurement data. This process based on our security assumptions adds additional evidence to the Measurement Values created by the system to log. The created documentation structure containing the raw values of the measurements plus additional information is called Measurement Record (MR). The generic contents of a MR are listed below:

- Identity of the device
- Identification number of the MR
- Location (might be defined through MVs or the device identity but can additionally be given)
- List of raw Measurement Values (MVs) (to describe the contents of this MR)
  - Related hash values (to secure integrity of MVs and to bind the actual MVs to the MR)
  - Signature of the hash value (to provide authenticity)
- Time (related timing information to verify at a later point of time when the event happened and how the time stamp was created)
- Status of the system (to prove the integrity of the running software)
- Completeness value (to prove the completeness of transmitted data)

The ***identity*** of a Traffic Monitoring System or device ID is used to validate the identity of an instrument. Only data measured by trusted a TMS is accepted. The ***identification number*** of the Measurement Record is applied to properly sort the incoming MRs and control the flow. If a MR is lost this might be determined through the identification number.

The ***location*** needs to prove that the TMS was used at the correct position. Especially, for speed cameras restrictions on the positioning of the device are given. For example, it is important to set up the TMS so that both sides of the road can be seen in the pictures and that the driving direction can be determined. For this, it is best practice to create a reference measurement when the device is installed and repeat them in periodical time intervals [66]. The location can implicitly or explicitly be given. Details are described with the acquisition.

Raw Measurement Values are the data measured by the TMS. MVs can be data in any format whether it is a speed representation, a number, a picture or a video. The MVs are the actual evidence that is used to prove that an event happened and in which extend. This evidence needs to be secured through other collected security values. Depending on the local legislation, MV must be secured individually or can be secured in a set. Securing more than one MV would drastically reduce the performance of the TMS. Anyhow, a ***list of the MVs*** is stored in the MR to identify which MVs

the MR comprises, be it just one MV or more. The hash value of the MV set ensures the integrity of the MVs. Besides, the signature over the hash value guarantees that the hashed data originates from the TMS holding the private part of the signing key. The hash value and the signature ensure the integrity and the authenticity of the MVs. Consequently, the actual MVs can be transmitted separately to the AEU. If they get corrupted this can be determined through the hash value and the signature enclosed in the MR.

The ***time stamp*** is specifically added to each MVs at the time of their creation. The time stamp is needed for proving the time of the happened event. The v1.2 TPM Specification newly defines a Tick Counter [84]. The time stamp created by the TPM is no real time value but only a Tick Counter. Since the time stamp is no real time value additional information is necessary for the association of tick count and real time value. The detailed description for the time stamping process is found in section 4.6.3.

The ***system status*** must be documented to provide the MVs with a proof that the system was running correctly and no faults occurred during the creation. This can be done in three different ways:

1. Explicitly - Adding PCR values and the system measurement log.

2. Implicitly - Using sealed signing and so binding the signing key to a trusted system configuration.

3. Both - Combining the explicit and implicit way.

The explicit documentation of the system state provides for the inclusion of the measurement log and the current PCR values. Both are continuously created by the TPM during runtime and represent the system status. The special creation technique of PCR values and the documentation in the measurement log are considered to unchangeably document the system status with the help of the TPM (compare 3.2.2 and 3.3). The current PCR values are compared to known pre-defined system values. If the PCRs equal trusted pre-defined values the integrity of a system is proven. Particularly, this proves at a later state that a system was running in a pre-defined state. It is believed that if the system is of integrity the conducted operations must have been correct.

The implicit way gives the opportunity to link the MVs to a special system configuration by binding a signing key to a trusted system state (see Sealed-Signing in 3.3.3 for comparison). Given that a signing key needs to be loaded for several purposes (e.g. during the TPM_Quote, TPM_TickStampBlob, TPM_Sign operations) adding another security mechanism of the TPM to it is convenient. The signing key is bound to PCR values during the creation of the key. This is done in setting the PCRInfo field to specified PCR values during the TPM_LoadKey execution. The key can only be loaded by the TPM if the current PCR values equal the values the key was bound to. So, if alterations to the system state have been carried out the key cannot be

loaded and no signing operation can take place. No signature means that the authenticity and integrity of the system cannot be granted. Then, all measured values are considered as untrusted and invalid. At this point, it is the operators' decision whether to keep the measured values or to discard them instantaneously. In contrast to the explicit solution where the system status can only be determined during the evaluation process, this solution would diminish the data overhead.

The third idea combines the implicit and explicit approach. In combining both approaches the data overhead is reduced but still the current system configuration is documented. The explicit documentation of the system state is convenient especially when it must be proven to a third party. Having a written documentation of the system status might be better deducible by non-experts of Trusted Computing. And consequently, helps in presenting and proving that the TMS was working correctly in court.

The ***completeness value*** is added to the MR so that it can be determined that all measured values have been transmitted. As mentioned in threat T7 it might be possible that not all values are transmitted. The completeness value consist of the PCR 11 value and the related SML (10 or 15 would also be possible). This PCR is used because it is unspecified in its usage and not already occupied through TrustedGrub or GRUB-IMA (cf. Section 4.6.1). We define this PCR as to store all hash values of the measured data. In doing so, it is later possible for the operator of the TMS to determine if the whole set of data was transmitted. With the measurement records the operator owns all needed hash values to re-compute if the current hash value is matching with the one he computed. If the value matches the operator is sure that the whole set of data was transmitted. And in the next instance, he must only remember this hash value for the next computations. If the values do not match a problem occurred transmitting all measured data. This principle is based on the explicit system status recognition. If this explicit method is implemented the list of PCRs to be reported must just be enhanced with the PCR value 11. Then again, if the implicit system status method is put into practice the values must be retrieved and documented separately.

Acquisition of MR information

Measurement Values are created by measuring sensors. These sensors are located either inside the TMS or remotely. MVs are delivered to the system including the TPM and their processing starts. It is assumed that the transmission from the sensors to the actual editing component is authentic and of integrity in any cases. Provisions must be made by the manufacturer to warrant the trustworthiness of the measuring sensor and its captured data. The MVs are not processed in the original format but a SHA-1 hash value is generated and all further processing is done on this hash value. The hash value represents the original data and serves to protect the integrity of it. Operating on the hash significantly decreases the processing time. Since the current available TPMs are not built for performance the processed amount of data must be minimized anyway. To further reduce the data volume the hash value can be calculated over a set of MVs.

As previously stated, for this reason, only the hash value is stored in the Measurement Record. The hash can be calculated using different TPM commands depending on the actual implementation (e.g. TPM_SHA1Start, TPM_SHA1Update, etc.) [88]. The signature on the hash value is created with the TPM_Sign command [88]. Details about the signing process are described later in this section.

The identity of the device is imprinted in the hardware. Best would be invisible from the outside to prevent any corruption. Besides, the identity of the device can be determined through the unique identity of the TPM. One distinct identification number is given to the first Measurement Record. Then, every time a new Measurement Record is created the number is incremented.

The location is given implicitly or explicitly. Implicitly it is given through reference measurements exactly defining the position like pictures or videos. Through the analysis of picture or video information it is possible to determine the position of the TMS. Pictures and videos may show distinct location characteristics used to identify the position. GPS localisation is not employed because at the moment the supplied information is inaccurate. Digital maps currently do not have sufficient accuracy to warrant correct localisation even with perfect GPS reading. Depending on road type, GPS systems are supposed to incorrectly identify the road a vehicle is on in 13-27% of the time [46]. Consequently, this kind of information is considered as not valid for our purpose. Alternatively, it is implicitly defined through the device ID considering that it is unique and the TMS is fixed at a specific location. If not implicitly given the location must explicitly be defined through any term the operator stipulates that distinctively describes the position. For example, it can be documented separately by a person during set up.

The real time must be associated with related timing Information. To accomplish this association, the current tick count structure is time stamped with a real time value. Where, the current tick count structure, namely the current tick count, the associated session nonce and the tick rate, are provided though TPM_TickStampBlob function. The real time value is obtained through a remote entity, the Timing Authority. Since this is a complex process all details are described in section 4.6.3.

The status of the system can be documented in different ways. The implicit solution binds a trusted system configuration with a signing key to the Measurement Values. The key is bound via the PCRs to the system state during the key creation. The TPM_LoadKey command always checks before loading a key if the key was bound to a specific configuration. The parentPCRStatus and the PCRInfo field indicate if a key in the storage hierarchy or the currently to be loaded key was bound to a system state. The signing key cannot be loaded if the PCR values do not match. For the explicit and combined solution the PCR values and the Stored Measurement Log (SML) must be obtained. As mentioned before, the combined solution additionally uses the sealed-signing capacity of the TPM. The PCR values and the SML, both, are protected and it is impossible to delete or change the contents. As described in sections 3.2.2 and 3.3

the PCRs store hash values of all executed events in the following form:

$$PCR[n] \leftarrow SHA-1(PCR[n] + measureddata) \quad (1)$$

In this manner, the order of the events and its current hash value influence the next hash. Thus, every discrepancy can be seen. The SML, on the other side, exactly reflects the executed events and their order. To verify, if the system configuration is trusted, the SML in conjunction with pre-defined hash values for trusted events are used. The trusted hash values can be compared to the measured process hash values in the SML to determine malicious events. Using the order of events stored in the SML combined with some pre-defined hash values a trusted hash values can be calculated using (1). The trusted hash value must equal the hash value in the PCR. Otherwise, the system configuration fails to pass the integrity check.

PCR values are obtained through the TPM_Quote command. TPM_Quote offers certain means for cryptographically reporting the current values of a chosen PCR and some external data. Among others, the external data value is used for anti-replay purposes. The TPM_QUOTE_INFO structure defines the output format of the TPM_Quote command. Figure 12 shows the structure of the output as described in [87].

```
typedef struct tdTPM_QUOTE_INFO{
  TPM_STRUCT_VER version;
  BYTE fixed[4];
  TPM_COMPOSITE_HASH digestValue;
  TPM_NONCE externalData;
} TPM_QUOTE_INFO;
```

Figure 12: TPM_QUOTE_INFO structure [87, page 99]

As noted in Table 6 the digestValue is a hash value of the chosen PCR and the externalData is typically a nonce used against replay attacks. The TPM_Quote operation signs the hash of a TPM_QUOTE_INFO object with an especially loaded signature key. The signature then is returned [88].

As Mitchell [63] said the SML stores details of all events measured and recorded in PCRs. Furthermore, it is described as maintaining an ordered database for integrity changing events. The SML is stored outside the TPM in a non-shielded location since it is likely to grow large. This is because the complete history of all 16 PCR values needs to be saved. Thus, the SML is located outside the TPM but within the TSS. According to Mitchell [63], the SML is not a log but an array of events. All events are logged in form of the TSS_PCR_EVENT structure. The TCS (TCG Core Service) Event Log Services maintain the SML. [63]

The TCG Software Stack (TSS) Specification Version 1.2 defines the TSS_PCR_EVENT structure as denoted in Figure 13 [86]. Details about the used fields are

Type	Name	Description
TPM_STRUCT_VER	version	This MUST be 1.1.0.0.
BYTE	fixed	This SHALL always be the string 'QUOT'.
TPM_COMPOSITE-_HASH	digestValue	This SHALL be the result of the composite hash algorithm using the current values of the requested PCR indices.
TPM_NONCE	externalData	160 bits of externally supplied data.

Table 6: TPM_QUOTE_INFO Data Structure [87, page 99]

```
typedef struct tdTSS_PCR_EVENT
{
TSS_VERSION versionInfo;
UINT32 ulPcrIndex;
TSS_EVENTTYPE eventType;
UINT32 ulPcrValueLength;
BYTE* rgbPcrValue;
UINT32 ulEventLength;
BYTE* rgbEvent;
} TSS_PCR_EVENT;
```

Figure 13: TSS_PCR_EVENT structure [86, page 98]

described in Table 7. All entries are stored as an unstructured array within the ACPI (Advanced Configuration and Power Interface) table. This is practicable, as there are already mechanisms available that allow for the transfer of this information to the post boot state. Once the post boot state governs the platform, the post boot OS is believed to read this data and transfers it to its own event log [83]. The SML must be retrieved from the ACPI table.

The ACPI table is used to identify the Trusted Computing capabilities to the post boot environment. As long as the BIOS has not performed the transition from pre boot to post boot state this table is considered to be invalid. [83]

The completeness of the transmitted data is determined through the completeness value. This value is the current hash value of PCR 11 and the related SML. PCR value and SML can be obtained as it is described previously. The targetPCR parameter of TPM_Quote is set to 11 so that the index of the PCR that is to be reported is delivered.

Name	Description
versionInfo	Version data set by the TSP.
ulPcrIndex	Index of the PCR this event belongs to set by the TSP.
eventType	Flag indicating the type of the event (see section 2.3.2.21 for definition).
ulPcrValueLength	The length (in bytes) of the rgbPcrValue parameter set by the TSP rgbPcrValue pointer to memory containing the value extended into the TPM by Tspi_TPM_PcrExtend. This SHALL be the result of the calculation of SHA-1(ulPcrIndex || pbPcrData || eventType || rgbEvent), where ulPcrIndex and pbPcrData are passed as parameters to the Tspi_TPM_PcrExtend command. Note that 1.1 TSPs may calculate this parameter as SHA-1(ulEventLength || ulPcrIndex || rgbEvent || eventType), where ulPcrIndex is passed as a parameter to the Tspi_TPM_PcrExtend command.
ulEventLength	The length (in bytes) of the rgbEvent parameter.
RgbEvent	Pointer to the event information data.

Table 7: TSS_PCR_EVENT Data Structure [86, page 98]

Signing of Data

When all required information is assembled in one XML structure namely the measurement record the whole object is digitally signed. This secures the authenticity of the MR that it can reliably be believed to be created by a trusted TMS. The signing can be done in two different ways the general signing operation with TPM_Sign command or the implicit signing operation with the TPM_TickStampBlob command.

At large, the TPM_TickStampBlob command carries out a signature on externally supplied data with a specially created signing key. The externally provided data, in this case, could be a hash value of the already collected MR data. At this point, the timing information would not be included in the MR as this comes with the TPM_TickStampBlob operation. The result is a SHA-1 signature and a parameter of the TPM_CURRENT_TICKS structure. The creation of the signing key is described together with the general signing operation. The usage of the TPM_TickStampBlob command is described in more detail in Section 4.6.3.

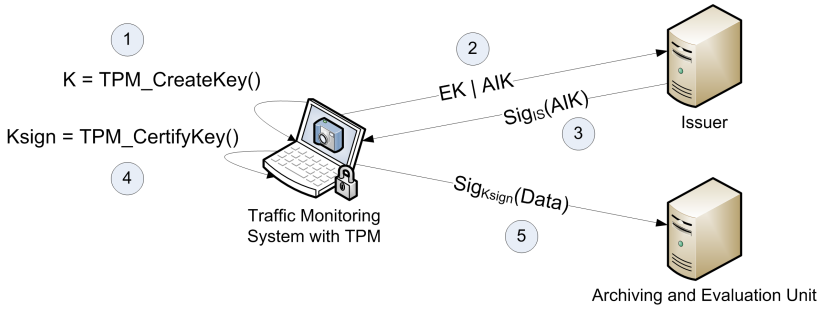

Figure 14: Signing Process

Figure 14 illustrates the particular steps of a general signature creation. In the first step, the TPM must generate a non-migratable key K_{sign} with the TPM_CreateWrapKey operation. The key usage is set to SIGN to restrict the designated usage. This key is needed as signature key that never leaves the inside of the TPM. So it can be warranted that no compromise took place and that only this specific TPM signed the MR. Additionally, K_{sign} can be bound via the TPM_LoadKey restriction during the creation to the instantaneous system status of the TMS. The PCRInfo field is initialised with a manufacturer or operator given combination of PCR values. K_{sign} was securely created but since a third party, e.g. the AEU, must verify the trustworthiness the key must be certified. Steps two to four in Figure 14 show the certification process. K_{sign} must be signed by an AIK to prove its qualities Qualities here mean that the key is non-migratable and bound to the system configuration. At this point latest, the AIK must be generated by the TPM and an AIK credential must be obtained from a PCA. The TPM_MakeIdentity command is executed for the creation of the AIK. To receive an AIK credential from the PCA, the generated AIK and the private portion of the

EK must be transmitted to the PCA. The PCA then verifies if the querying TMS is a trusted TMS with the help of the EK. If the TMS passes the integrity check the PCA issues the AIK credential and sends it back to the TMS. This is demonstrated in step two and three in Figure 14. These two steps also can be separated from the actual signing process with key K_{sign}. Since creation of an AIK and issuing of an AIK credential are independent of the K_{sign} creation this process easily can be outsourced. It is advantageous to acquire the AIK and the related credential in an early processing phase to avoid a time delay until the next signing key certification step can be carried out. Now, K_{sign} can be certified using TPM_CertifyKey. This command creates a credential using the formerly generated AIK. K_{sign} is now prepared for signing MRs. In the last step the signature will be carried out using TPM_Sign. This command carries out a SHA-1 digital signature on the provided data. Here, the data to sign is the hash value of a MR. The hash value represents the actual data and can be signed instead. This will save processing power and time since only a small hash value must be signed and no large images.

Detailed operation description

TPM_CreateWrapKey:

With the TPM_CreateWrapKey command it is possible to create any kind of keys. The key created can be bound to PCR values [88]. It returns a TPM_KEY structure which holds the newly created key. The TPM_CreateWrapKey command always needs authorisation even if TPM_AUTH_NEVER is set [87, page 33].

```
typedef struct tdTPM_KEY{
  TPM_STRUCT_VER ver;
  TPM_KEY_USAGE keyUsage;
  TPM_KEY_FLAGS keyFlags;
  TPM_AUTH_DATA_USAGE authDataUsage;
  TPM_KEY_PARMS algorithmParms;
  UINT32 PCRInfoSize;
  BYTE* PCRInfo;
  TPM_STORE_PUBKEY pubKey;
  UINT32 encDataSize;
  [size_is(encDataSize)] BYTE* encData;
} TPM_KEY;
```

Figure 15: TPM_KEY structure [87, page 88]

TPM_CertifyKey:

The TPM_CertifyKey command certifies with one key the public portion of another key [88]. The TPM_CertifyKey command distinguishes between the keys which are used to issue a certificate. Signing and legacy keys may be applied for the certification of both migratable and non-migratable keys. Consequently, the value of a certificate

Type	Name	Description
TPM-_STRUCT-_VER	ver	This MUST be 1.1.0.0
TPM-_KEY-_USAGE	keyUsage	This SHALL be the TPM key usage that determines the operations permitted with this key
TPM-_KEY-_FLAGS	keyFlags	This SHALL be the indication of migration, redirection etc.
TPM-_AUTH-_DATA-_USAGE	authDataUsage	This SHALL Indicate the conditions where it is required that authorization be presented.
TPM-_KEY-_PARMS	algorithmParms	This SHALL be the information regarding the algorithm for this key
UINT32	PCRInfoSize	This SHALL be the length of the pcrInfo parameter. If the key is not bound to a PCR this value SHOULD be 0.
BYTE*	PCRInfo	This SHALL be a structure of type TPM_PCR_INFO, or an empty array if the key is not bound to PCRs.
TPM-_STORE-_PUBKEY	pubKey	This SHALL be the public portion of the key
UINT32	encDataSize	This SHALL be the size of the encData parameter.
BYTE*	encData	This SHALL be an encrypted TPM_STORE_ASYMKEY structure or a TPM_MIGRATE-_ASYMKEY structure

Table 8: TPM_KEY Data Structure [87, page 88]

depends on the trust in the certifying key. If the verifier is not convinced of the trustworthiness of for example a migratable key, he will not trust in the certificate [88]. On the other hand, TPM identity keys can only be used to certify non-migratable keys. The certified keys are always stored in a protected environment and never leave it. In consequence, the verifier must be ensured that the entity that issued the certificate implements its policies correctly if he trusts the TPM. If trusting the TPM the verifier must put trust in this entity. Additionally, the verifier must trust the TPM manufacturer and his maintenance policy. Only then, the verifier will totally trust the certificate created. The result is handed back in a TPM_CERTIFY_INFO structure if no locality restrictions are made on the certified key [88]. Since an AIK is used to certify a non-migratable signing key. AIKs belong to identity keys, so if the AEU trusts the TPM manufacturer and the PCA who issued the AIK credential then it trusts the signing key. The TPM_CERTIFY_INFO structure comprises the information that describe the key to be certified. Figure 16 and Table 9 show the contents of this structure.

```
typedef struct tdTPM_CERTIFY_INFO{
  TPM_STRUCT_VER version;
  TPM_KEY_USAGE keyUsage;
  TPM_KEY_FLAGS keyFlags;
  TPM_AUTH_DATA_USAGE authDataUsage;
  TPM_KEY_PARMS algorithmParms;
  TPM_DIGEST pubkeyDigest;
  TPM_NONCE data;
  BOOL parentPCRStatus;
  UINT32 PCRInfoSize;
  [size_is(pcrInfoSize)] BYTE* PCRInfo;
} TPM_CERTIFY_INFO;
```

Figure 16: TPM_CERTIFY_INFO structure [87, page 96]

TPM_LoadKey:

The TPM_LoadKey command loads the key for further usage into the TPM. For further operation like wrap, unwrap, bind, unbind, seal, etc. keys need to be present in the TPM. Furthermore, TPM_LoadKey enforces restrictions on the key usage. PCR values are controlled if the key was bound to a specific configuration state. The parentPCRStatus value is used to determine whether any previous key in the key hierarchy was bound to a PCR. Any restrictions like the size of keys are checked on every key loaded. [88] In this case, especially the use of keys restricted to a specially pre-defined system status is enforced.

TPM_MakeIdentity:

A new AIK is created using this command. The user is advised to set the labelPrivCADigest. This label identifies the PCA that must be concerned with the certification

Type	Name	Description
TPM-_STRUCT-_VER	version	This MUST be 1.1.0.0.
TPM-_KEY-_USAGE	keyUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified. The upper byte MUST be zero.
TPM-_KEY-_FLAGS	keyFlags	This SHALL be set to the same value as the corresponding parameter in the TPM_KEY structure that describes the public key that is being certified. The upper byte MUST be zero.
TPM-_AUTH-_DATA-_USAGE	authDataUsage	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified.
TPM-_KEY-_PARMS	algorithmParms	This SHALL be the same value that would be set in a TPM_KEY representation of the key to be certified.
TPM-_DIGEST	pubKeyDigest	This SHALL be a digest of the value TPM_KEY → pubKey → key in a TPM_KEY representation of the key to be certified.
TPM-_NONCE	data	This SHALL be externally provided data.
BOOL	parentPCRStatus	This SHALL indicate if any parent key was wrapped to a PCR.
UINT32	PCRInfoSize	This SHALL be the size of the pcrInfo parameter. A value of zero indicates that the key is not wrapped to a PCR.
BYTE*	PCRInfo	This SHALL be the TPM_PCR_INFO structure.

Table 9: TPM_CERTIFY_INFO Data Structure [87, page 96]

of this AIK. It was introduced because the TSS cannot be trusted to always correctly contact the right PCA. [88]

TPM_Sign:

This command signs the given data and returns the digital signature. The signature is computed by the TPM using the key referenced by keyHandle. [88]

4.6.3 Time Stamping

As described in the previous sections a time stamp is added to the measurement data. Now, we want to have a closer look at the creation of time stamps. The utilized time stamp actually is a timing value associated with a signature. A time stamp is used out of two reasons. First, with time stamps the point of time an event happened is assured. This is especially necessary to prove the presence of a person and to reconstruct the order of events. Second, it must be proven that the measured data has not been altered since it was seized. With the combination of time stamps and signatures one distinct moment can be determined when data was known to be correct and unaltered. In particular, when initiating legal action, it is necessary to prove the integrity of measurement data many years after the evidence was assembled. Time stamps can bind the measured data needed to secure, the signers' identity and the time of an event. Once bound, the integrity of data can be determined years after the data actually was collected. It just needs to be warranted that the utilized time stamps are tamper resistant and auditable. In his paper, Hosmer [47] said that secure and auditable time stamps improve the integrity of digital evidence as well as they provide higher assurance required for the digital chain of custody. Later, Hosmer defined attributes for 'secure, auditable digital date/time stamps' [47, page 4]:

- 'Accuracy. The time presented is from an authoritative source and is accurate to the precision required by the transaction, whether day, hour, or millisecond.
- Authentication. The source of time is authenticated to a National Measurement Institute (NMI) timing lab so that a third party can verify the precision and accuracy of the time.
- Integrity. The time should be secure and not be subject to corruption during normal "handling." If it is corrupted, either inadvertently or deliberately, the corruption should be apparent to a third party.
- Non-repudiation. An event or document should be bound to its time so that the association between event or document and the time cannot be later denied.
- Accountability. The process of acquiring time, adding authentication and integrity, and binding it to the subject event should be accountable, so that a third party can determine that due process was applied, and that no corruption transpired.' [47, page 4]

By warranting these attributes a manipulation resistant time stamp can be created. They are considered to be retraceable and correct. For this purpose it is necessary to bind the manipulation resistant time stamp to the measured data. In doing so it can be guaranteed that at a later stage a third party, e.g. the courts, can verify the legal effect of the evidence. It must be kept in mind that secure time stamps are not only responsible for guaranteeing non-repudiation of measurement data. The security of data needs to be provided at all times. Within this concept this is tried to realise with a variety of Trusted Computing techniques. In this section we concentrate on how a manipulation resistant time stamps can be created with the help of the TPM and a Timing Authority (TA). Further provisions made beyond time stamping are described in the preceding sections 4.6.1 and 4.6.2.

TPM and Time Stamps

The utilization of a time stamp with the help of the TPM is not as simple as it seems to be at the first glance. A time stamp for the hashed Measurement Values is needed in form of a certified real time information. However, the TPM does not own an internal source of time. It merely offers a number of tick counts which have occurred since the start of a timing session. The in v1.2 TPM Specification newly introduced Tick Counter is used for this purpose [84]. In this context defined timing sessions are platform dependent events that potentially will or will not correspond with TPM_Init and TPM_Startup sessions [90]. The TPM time measurement service TPM_TickStampBlob - relates a time stamp to various blobs. The provided time stamp is not a real universal time clock (UTC) value but a number of timing ticks. We assume that it is best to use the UTC time zone when creating timestamps. On different machines may be involved different time zones other than the time zone used on the AEU, so using a common reference point will make tracing much easier. It is still possible to change the UTC presentation to other more suitable notations. The timing ticks are always available for use although, the association of ticks and actual and local real time need to be done partly outside of the TPM. An online clock synchronization protocol must be established using a mechanism that employs TPM_TickStampBlob.

A separate clock circuit may serve as time source for the Tick Counter. As stated in the TPM specification [90, page 102], the availability of the ticks and the tick resolution is an area of differentiation available to TPM manufacturers and platform providers. The inconsistent availability of power to the TPM is the reason for this difference. In contrast to desktop PCs, mobile PCs are not always connected to the wall socket and its power supply is reliant on the internal battery. This can cause power variations which decrease the ability to maintain the Tick Counter. Consequently, it is not always granted that a TPM can continue incrementing the Tick Counter across power cycles. It is the platform designer's decision as to how they supply power to the TPM to maintain the tick count. The specification [90, page 102] does not bring up any requirements on the ability for the TPM to maintain the incrementing of the Tick Counter across power cycles or in different power modes causing problems. But since a stable and reliable timing source is needed the variances in counting the ticks must be eliminated.

Certain precautions need to be considered to provide a secure tick count and with it a secure real time value. When tick count critical events occur a new timing session must be started to avoid tick incrementing problems. For this, the tick count value is reset when power cycles occur, and when timing protocol procedures finish incorrectly. Also, the Tick Counter is reset in discrete intervals before the internal deviation of the clock becomes too large.

The tick value is reset during each power cycle because if no power supply can be granted the tick count cannot be reliable incremented. Errors may occur that are avoided with resetting the tick value. As well, the tick count is reset when the timing protocol (4.6.3) does not finish properly. This means, for instance, that the expected answer (the signature of the time stamp) has not arrived after exceeding one minute or that an error message was received. One minute is the assumed upper boundary in receiving a valid time stamp. Since the maximum time discrepancy of one minute to the actual real time value is defined in [22]. Using this value, time stamps will in any case comply with the legal timing restrictions. Furthermore, the internal deviation of the clock of a TPM is crucial for the tick count. The clock deviation can be different from manufacturer to manufacturer. To circumvent problems the tick value is reset in discrete intervals. The TPM maintains for each timing session three values [90, page 102]:

- Tick Count Value (TCV) the current tick count for the session.
- Tick Increment Rate (TIR) the rate at which the TCV is increased. There is an association between TIR and seconds, the relationship is set during manufacturing of the TPM and platform. This is the TPM_CURRENT_TICKS tickRate parameter.
- Tick Session Nonce (TSN) nonce renewed at each start of a tick session.

These three values comply with the parameters given in the TPM_CURRENT_TICKS structure described in [87](see Figure 17). A detailed description of the TPM_CURRENT_TICKS data structures can be found in Table 10. There it is denoted that the minimum resolution of the tick rate is one microsecond.

```
typedef struct tdTPM_CURRENT_TICKS {
  TPM_STRUCTURE_TAG tag;
  UINT64 currentTicks;
  UINT16 tickRate;
  TPM_NONCE tickNonce;
}TPM_CURRENT_TICKS;
```

Figure 17: TPM_CURRENT_TICKS structure [87, page 117]

In consequence, resetting the tick count value starts a new tick session and is always accompanied with the reallocation of a new session nonce. Additionally, a new session means to newly acquire a UTC value. Both have to be done to sustain the relation between the current tick count and the obtained real time value.

Type	**Name**	**Description**
TPM- _STRUCTURE- _TAG	tag	TPM_TAG_CURRENT- _TICKS.
UINT64	currentTicks	The number of ticks since the start of this tick session.
UINT16	tickRate	The number of microseconds per tick. The maximum resolution of the TPM tick counter is thus 1 microsecond. The minimum resolution SHOULD be 1 millisecond.
TPM_NONCE	tickNonce	The nonce created by the TPM when resetting the currentTicks to 0. This indicates the beginning of a time session. This value MUST be valid before the first use of TPM_CURRENT- _TICKS. The value can be set at TPM_Startup or just prior to first use.

Table 10: TPM_CURRENT_TICKS Data Structure [87, page 117]

Terms

To avoid confusion, some terms describing the time are defined:

- **Real time** UTC time obtained from the Timing Authority. Ideally, this timing value is consistent with the event time value. But in general, will deviate from the event time value for some millisecond.
- **Event time** time at which the event was measured and the measurement data created. Time that needs to be calculated out of the real time value and the TPM_CURRENT_TICKS values.
- **Elapsed time** delta time representing the duration of the time span passed since the real time was obtained and the measurement data was created.

Time Stamp Concept

The general process of time stamping is done via a remote TA (compare Figure 18). The process starts with the creation of the value that needs to be time stamped. In this case, it is not relevant what is signed as this process only aims to establish a relation between the real time and the ticks. So, it is made use of a nonce as input value. The nonce is hashed and sent to a Timing Authority (TA). The TA generates an exact date at which it received the hashed nonce. Then it signs the hashed nonce with the concatenated date with its private key. This is the time stamp. Later, the signature warrants that a third party is able to verify the timestamp. The time stamp is sent back to the TMS which adds the time stamp to its created nonce. For verification both, the nonce and the related time stamp are sent to a third party, in this case, the AEU. The AEU retrieves the public key of the TA and verifies the signature of the time stamped nonce. It is of concern that the TA is trusted and able to issue legally relevant time stamps. Or else, the time stamp would loose its significance.

Time stamping with the TPM is done in three processes including the previously described creation of time stamps with a remote TA. The first process establishes a relationship between Tick Counter and real time. The second process gathers the values needed for the third process. These values cover the session values obtained in process one and the present TPM_CURRENT_TICKS values for the effective measurement data. And last, the third process calculates the real time value out of the given information. Particularly, the first process must be repeated every time a new timing session starts. Process two anyway is constantly executed during the runtime and automatically gets assigned a new timing session. Process two should not be carried out if process one is already running because a transition between two timing sessions is executed. Process three does not obey timing sessions. The third process is treated isolated from the first two processes. Since all required values are saved together with the measurement data. And thus, will be always available for calculation.

Subsequently, the first process of time stamping with the help of the TPM is described elaborately. As mentioned in the preceding paragraphs, for establishing reliable

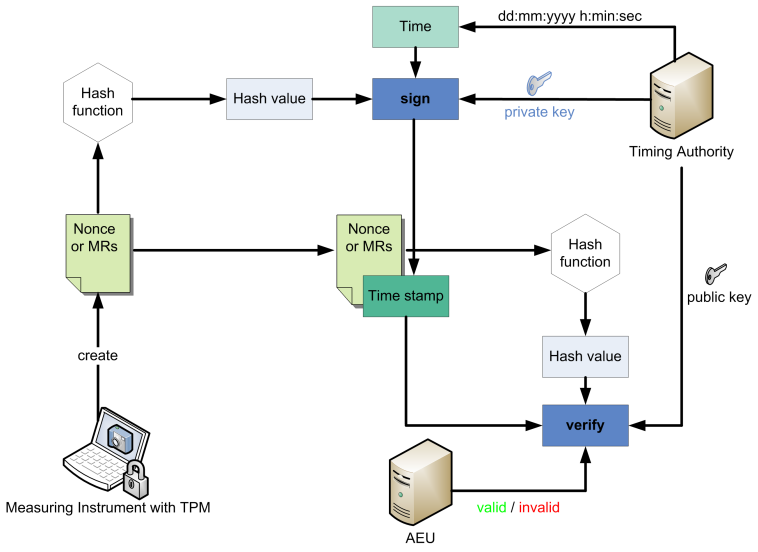

Figure 18: General Time Stamping Procedure

timing information it is required to prove a relationship between real time and Tick Counter. The Tick Counter starts with the activation of the TPM. To determine the actual real time value of an event or better to document the relation between Tick Counter and the real time following three steps have to be completed:

- The method TPM_TickStampBlob is executed (T1).
- The result T1 is transmitted to the Timing Authority (in the net) and T1 is time stamped (T2).
- T2 is countermarked through the TPM_TickStampBlob function (T3).

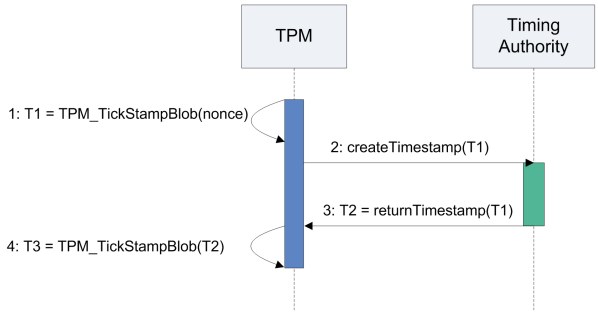

Figure 19: TPM Time Stamping

By this means, the relation between real time and tick count is established. One would argue that the third step does not need to be taken. But only, with its completion the elapsed time from the start of the process up to the receiving of the time

stamp can be included in the timing calculations. The time of an event cannot be fixed to an exact real time value but is designated to a time interval. The elapsed time might be indefinitely high what must be prevented at all events. As in [22] defined the interval should not exceed one minute (cf. Section 4.6.3). The obtained real time value is protected through the signatures of the TPM_TickStampBlob command and the signature provided by the TA. Hence, it is protected from tampering or foisting. Figure 19 illustrates these steps.

The second process is to obtain the current tick count (CTx), the related session nonce (Nx) and the current tick rate (TRx). So the acquired real time value can be linked to the current tick count. The current tick count reflects the event time. The mentioned values are found by executing TPM_TickStampBlob on the hash value of the recent MR. The TPM_CURRENT_TICKS object is returned. The signature key K_{TS} is used for the signature creation in the TPM_TickStampBlob command. This key can be bound to specified system state. So, the integrity of the operation and the authenticity of the signed MR is warranted. Finally, with the values collected during process one and two, it is possible to actually calculate the concrete event time value in process three.

The first and the second process especially rely on the usage of the TMS TPM_TickStampBlob command. TPM_TickStampBlob is a TPM command that creates a time stamp on a passed blob: $TS \leftarrow Sign_{K_{TS}}(blob||t||TSN)$ Whereas, K_{TS} is the signature key, blob is the digest to stamp, t is the current time and TSN is a nonce determined by the TPM. The blob must be present in the TPM at the time t indicated. The command performs a digital signature with a given signature key K_{TS}. The signature is carried out on the hash of the blob and the current tick count [72,88]. The blob can contain any data. In the first process during the acquisition of the real time the blob is filled with a nonce. At this point, it is not necessary what data is signed but only the determination of the relation between tick count and real time is important. In the second process, the hash value of the currently created MR is given as input for the blob. So, it is implicitly achieved to authenticate the MR with the granted signature. TPM_TickStampBlob returns the current tick value in form of the TPM_CURRENT_TICKS structure. From now on, the returned object will be referred to as *TickStampBlob*. To further protect the usage of TPM_TickStampBlob the usage of the signature key and thus, the whole completion of the command, can be bound to the systems' state. It will only be possible to carry out the signature if the Traffic Monitoring System is in a trusted system state. Therefore, to bind TickStampBlobs to the platform configuration state a signing key K_{TS} is created which is deliberately bound to a trusted PCR configuration. During the initialisation or preliminary phase before processing starts the non-migratable key K_{TS} is generated using the TPM_CreateWrapKey command. K_{TS} is bound to the instantaneous platform configuration and thus, is able to attest a configuration state. K_{TS} is bound through the PCRInfo field to the current combination of PCR values. But, a third party needs to reproduce this binding. Thus, as second step, a certificate is created that approves the limitations of the key K_{TS} and attests that K_{TS} is bound to the system state. For this, TPM_CertifyKey with a specifically created and certified

AIK is executed. To warrant a steady time source, TickStampBlobs are generated with K_{TS} in periodic intervals. Since the used key is bound to the system state the creation of the TickStampBlob is only possible if the system configuration did not change since the creation of the key. The system configuration still must confirm to the configuration denoted in the key certificate. If the platform configuration changes to an untrusted state the configuration must be regenerated to be valid again. If it changes to a trusted state a new signing key is needed as the old one does not comply with the system configuration. The steps for the creation of a restricted key and its certification need to be reiterated. Otherwise, no further time stamping will be possible. In consequence, the obtained real time value is also bound to the system state as all values are signed with the key through TPM_CreateTickStampBlob.

In the third process, as all required values for the calculation of the event time are acquired, it is possible to perform the rest of the calculations. The association of the current tick value (CTx) to the real time value is shown in the subsequent paragraph (see 4.6.3). After the execution of the first process the real time value is known. To determine the event time, first, the time span which elapsed since the real time value was obtained needs to be calculated. This is done by first comparing the session nonces. If they comply no new session was started and the real time value will conform to the CTx obtained in process two. Then, the calculation of the elapsed ticks can be performed. This is, CTx minus the current tick value of the real time (CTr). Stated mathematically:

$$elapsedTicks = CTxCTr$$

Now, the conversion of ticks to seconds must be established regarding the possible errors per tick. k1 is the relation between ticks and seconds and is obtained through the TPM_GetCapability command. k2 is the possible error per tick. Mathematically the elapsed time is calculated as:

$$elapsedTime = (k1 * elapsedTicks) + (k2 * elapsedTicks)$$

This is the delta time value of the elapsed time between the acquisition of the real time and the measurement of data. This value must be added to the real time value to determine the demanded event time. Stated mathematically:

$$eventTime = realTime + elapsedTime$$

With this value the indispensable event time is obtained. Since all values are obtained through trusted operations or entities this time stamp will bear up against any questions in legal actions.

Association of current tick value to real time value

As mentioned before, the event time cannot be precisely determined to be at one exact point of time. Precisely, the event time is settled in a timing interval since the acquisition of the real time value consumes some time. It takes some time between when a request goes to a Timing Authority and when the TA responds. The greater the time

difference the larger the delta value will be. Thus, the uncertainty of the relation of the event time value to real time value will be large. But, given that we restrict the time interval to maximum one minute the uncertainty is located in an appropriate scope. Another consideration would be to create a TickStampBlob and sending it to the TA each time a measurement value is created. However, this is not considered valuable because Measurement Values can be created in non-deterministic time intervals. So, it might be possible that shortly one measurement after another is done and that the TA is not able to process these measurements in an adequate time. This would not only cost a lot of calculation time and power but also money. The acquisition of a certified time stamp from a legal TA is pretty expensive when it comes to a regular issuance of time stamps. For instance, D-TRUST makes a time stamp service available which issues qualified time stamps after RFC 3161. Continuous time stamps will cost the inquirer between 0.10 and 0.14 per time stamp. If an value of 0.10 for 30,000 time stamps is assumed this will already cost 3.000 (zero-rated for VAT) [5]. As a result the purchase of time stamps sums up quickly and gives rise to costs. Consequently, first a real time value is obtained and then the time difference to this value is calculated to determine the event time.

The mathematical association of the real time value with the current tick structure, which is maintained inside the TPM, is shown subsequently. The Traffic Monitoring System first wants to obtain a real time value. For this, the TMS wants to have a nonce time stamped. The TMS performs TPM_TickStampBlob on this nonce and receives T1. T1 records the current tick value (CT1) and the associated nonce (N1). Now, T1 needs to be linked to a UTC time value. The monitoring device sends the result T1 to a TA. The TA time stamps T1. The result of this time stamping operation is T2. From this point on, T1 is related to an UTC1 time. The TA sends T2 back to the TMS which then performs another TPM_TickStampBlob operation on T2. The result T3 documents the current tick value (CT3) and the corresponding nonce (N3). So far, nothing new was explained. Now, the associations have to be examined. For comparison see [90, page 103-105]. The TMS now owns two time stamp results (T1 and T3) which need to be associated such as the connection of the real time and the event time becomes clear. Starting with the first obtained values, we know that CT1 is less than UTC1. This is true because the TA signs result T1 and T1 must be created before it can be signed. Mathematically:

$$CT1 < UTC1$$

On the other hand, CT3 must be greater than the UTC1. This is true since the signature of T2 cannot be executed before T2 was created. Stated mathematically:

$$CT3 > UTC1$$

It is certain that $CT1 < UTC1 < CT3$. This association is true if N1 matches N2. That means, no new timing session was started. If N1 does not match N2 the whole process must be started from the beginning. Still, it is undefined when UTC1 occurs in the timing interval with the boundaries CT1 and CT3. Indeed, the delta time defined by CT3 minus CT1 is the uncertainty of the current tick value in association to UTC1.

Stated mathematically:

$$\Delta CT = CT3 - CT1 iff N1 = N3$$

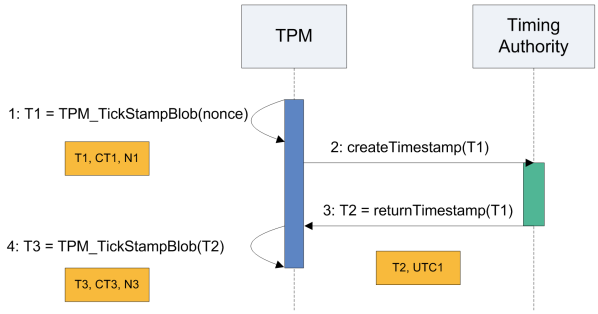

Figure 20: Association of Real Time to TPM Time Stamping

Now, the conversion of ticks to seconds must be established. This is done with integrating an error factor. Using the TPM_GetCapability command the relationship k1 between ticks and seconds can be obtained. Furthermore, the possible error per tick k2 must be obtained. Mathematically the delta time is calculated as:

$$\Delta time = (k1 * \Delta CT) + (k2 * \Delta CT)$$

$$\Delta time = (k1 * \Delta CT) + drift = timeChange + drift$$

Where $ABSOLUTEVALUE(drift) < k2 * \Delta CT$.

For the association of $\Delta time$, UTC and the event time following needs to be considered:

1. $CT1 < UTC1 < CT3$
   – True as seen above.

2. $CT1 < UTC1 < CT1 + CT3 - CT1 \leq CT1 + \Delta time (= CT1 + timeChange + drift)$
   – True because CT1 and CT3 are in the same timing session. timeChange is positive.

3. $0 < UTC1 - CT1 < \Delta time$
   – Subtract CT1 from all sides.

4. $0 > CT1 - UTC1 > -\Delta time = -timeChangedrift$
   – Multiply with -1.

5. $timeChange/2 > [CT1 - (UTC1 - timeChange/2)] > -timeChange/2drift$
   – Add timeChange/2 to all sides.

6. $timeChange/2 + ABSOLUTEVALUE(drift) > [CT1 - (UTC1 - timeChange/2)] > -timeChange/2 - ABSOLUTEVALUE(drift)$
   — Make the large side of an equality bigger, and make the small side smaller.

7. $ABSOLUTEVALUE[CT1-(UTC1-timeChange/2)] < timeChange/2+ABSOLUTEVALUE(drift)$
   — Definition of absolute value and timeChange is positive. So, $CT1 - (UTC1 - timeChange/2)$ must be positive.

As timeChange and absolute value are positive the bigger side must be positive. From which we see that CT1 is approximately $UTC1-timeChange/2$ with a symmetric possible error of $timeChange/2+ABSOLUTEVALUE(drift)$. Whereat, this error is less than $k1*\Delta CT/2+k2*\Delta CT/2$. That means, that the realTime can approximately be seen as the UTC1 value.

Protocol:

The protocol used for the first time stamping process can be designed as follows. This protocol detailed describes all steps of the acquisition of the real time.

1.) $TMS(TPM) : K_{\mathrm{TS}} = CreateWrapKey(PCRInfo \Leftarrow CurrentPCRs)$
2.) $TMS(TPM) : cert(K_{\mathrm{TS}}) = CertifyKey(K_{\mathrm{TS}})AIK_{\mathrm{priv}}$
3.) $TMS(TPM) : TickStampBlob0 = sig\{currentTicks\}K_{\mathrm{TS_{priv}}}, ticks0 = currentTicks$
4.) $TMS \rightarrow TA : TickStampBlob0, currentTicks, cert(K_{\mathrm{TS}}), SML, Cert(AIK)$
5.) $TA : validatecert(AIK_{\mathrm{pub}})$
6.) $TA : validatecert(K_{\mathrm{TS}}) \rightarrow PCRInfo$
7.) $TA : validateTickStampBlob0$
8.) $TA : timeValue = sig\{TickStampBlob0, realTimevalue\}K_{\mathrm{TA_{priv}}}$
9.) $TA \rightarrow S : timeValue$
10.) $TMS(TPM) : TickStampBlob1 = sig\{timeValue, currentTicks\}K_{\mathrm{TS_{priv}}}$

Timing Authority (TA):

Generally spoken, a Timing Authority (TA) is a trusted third party that provides a certified time stamping service. With a trusted time stamp the existence of certain data before a certain point can be proven without the possibility that the party obtaining the time stamp can backdate the timestamps [35]. In this design it is also possible that a PCA or an operator adopt the role of the TA. The only requirement is that the party taking over the role of the TA fulfils its role conscientiously. The entity must produce legally conformant time stamps and must not forge certificates or time stamps for its purposes.

As mentioned, it is possible to acquire time stamps from an external maintained entity (TA or PCA) or an entity managed by the operator of the TMS. External entities can be national institutions or companies which provide legally conclusive time

stamps. Those TAs are controlled in distinct time intervals if they comply with all legal requirements. As an external entity, e.g., D-TRUST can be mentioned as this company provides a time stamping service issuing qualified time stamps after RFC 3161. (For more information about RFC 3161 time stamps see [35]) Among external entities, the operator is able to provide a time stamping service as long as all legal requirements are fulfilled and can be testified. Therefore, the operator can obtain his own time stamping server. Time stamping sever which create legally conformant time stamps are offered by multiple companies like AuthentiDate or nCipher.

AuthentiDate, for example, provides the TimeStamp Server SP250 and other time stamp solutions. The TimeStamp Server SP250 is a combined software and hardware solution for the creation of at least 250 time stamps per second. AuthentiDate say that the TimeStamp Server SP250 is suited to protect critical processes and high volume documentations. Furthermore, they claim that it is suitable to build own time stamp services and Trust Centre [2].

Another company offering time stamp servers is nCipher. nCipher is part of Thales which is one of the world leaders in granting of Information and Communication System Security solutions. Governments, defence, critical infrastructure operators, the finance industry and other enterprises rely on their state-of-the-art security solutions protecting critical data and meeting most sophisticated regulatory requirements. Regarding time stamps, nCipher offers the nCipher Time Stamp Server. nCipher Time Stamp Server is a digital signing and time stamping server. It can be combined with nCipher Time Source Master Clock for protected and scalable time stamping. nCipher states to add an auditable, legally valid proof of time to digital documents without requiring for notary services for paper-based documents. Moreover, this time stamp server shall provide auditable evidence through adding digital signatures to transactions and reports. For instance, nCipher Time Stamp Server is certified by the National Institute for Standards and Technology (NIST) FIPS 140-2 Level 3 certification [8].

A TA accomplishes important tasks for a satisfying realisation of a trusted Traffic Monitoring System. First, the TA is needed to provide a real time value to the TPM. Second, the TA is needed to provide a time value that is admissible in a court of law. The Traffic Monitoring System obtains a real time value from a TA which is on the contrary to the system time a more reliable and admissible timing value. The system time is not applicable for legal enforcement because it is not in a unified format (like UTC) and varies from system to system. With the TA a time value can be used that is unified and retraceable. And third, the TA can support the validness of a signature in delivering a legally conformant certified time stamp and combining the signature with it. Ansper, Buldas, Roos, and Willemson [36] stated that signature schemes and their validation techniques are designed for a short-lived use and not for long-term validation. Therefore, they introduced a long-term validation protocol so that signatures do not loose their validity after their signing keys were revoked or their security was broken. The Traffic Monitoring System needs long-term provable authenticity of its signatures since data might be stored a long time before actually using it. This might be out

of the reason that it takes a long time to bring a case and thus the evidence in front of a jury. Or a traffic study and it evaluation might take longer than a few months. There are surely other purposes that need to prove that the collected electronic data maintain authentic long after their creation. On all accounts, the collected Measurement Values must be suitable. Ansper et al. [36] said that the 'evidentiary function is one of the essential properties of handwritten signatures and thereby it may seem natural to assume that digital signatures have this property as well.' [36, page 402]. For this, the revocation of a certificate should not restrict the authenticity of a signature. In their solution to this problem they linked time stamps to the signature.

4.6.4 Transmission

As last step the transmission of the Measurement Records need to be secured. It is necessary to establish a secure connection in between the communication partners, e.g. the AEU and the TMS. First, the authentication of the AEU towards the TMS must be ascertained. Whether to securely transport the Measurement Records to the right destination (the AEU) or to be ensured that updates are received from the right source (the AEU). Second, the AEU shall be ensured that it is communicating with a trusted Traffic Monitoring System and that this TMS is in a trusted system configuration. The authentication of the TMS towards the AEU must be ensured. And third, the transmitted MRs shall be secured against spoofing or other attacks. No attacker shall be able to read or obtain information contained in the MRs.

The AEU side authentication can be solved by implementing authentication protocols using certificates. Additionally, the AEU signs its software updates so that the TMS is sure having received an update from a trusted party.

To assure the AEU that it is communicating with the right Traffic Monitoring System a Trusted Computing concept is applied. The concept of remote attestation with privacy CA (see 3.4.3). This concept allows for authentication towards a third party and at the same time for approval of the integrity of the system. The TPM uses a certified key pair (AIK) that classifies the platform as a genuine TCG platform, to sign the current PCR values. In this design, the monitoring system is the TPM superior system since the TPM is embedded in the TMS. Thus, the TMS is actually communicating with the AEU using the underlying TPM functions. Depending on the system design, the AIK could be created during the manufacturing process and be equipped with an AIK credential. In this case, the manufacturer or provider would adopt the role of the PCA to certify the trustworthiness of the AIK. Or the AIK is created at a later stage and the certificate is remotely issued by a PCA or another trusted entity. Before the delivering of measurement data can take place the AEU can challenge a TMS to attest on its current configuration. The TMS sends the signed PCRs together with the measurement log containing each of the individual measurements that have been added to the PCR hash chains. Also, the certificate attesting the trustworthiness of the AIK is sent. The AEU reviews both to verify the correctness of the certificate and system state. If the verification turns out to be positive the measurement records

can be delivered. For comparison refer to 3.4.3. Figure 21 illustrates the attestation procedure of this design.

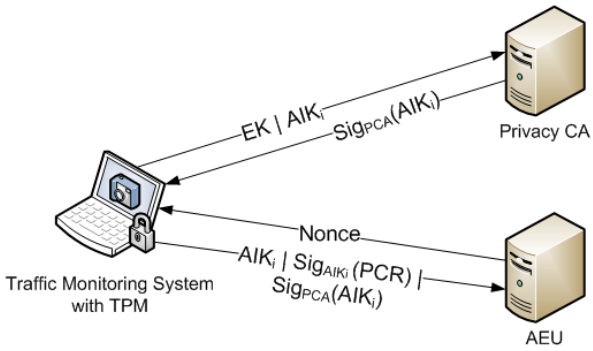

Figure 21: Remote Attestation with PCA

The anonymity problem which is often stated while speaking of remote attestation with PCA does not concern this model. It is actually desired to be able to reconstruct the identity of a monitoring device. For example, if errors in the system state of a Traffic Monitoring System occur it is necessary to definitely know which device failed. Especially, with a growing number of measuring devices remotely connected to one central AEU the distinction of instruments is important. This can diminish the downtime of the instrument and thus, the cost incurred due to failure. Furthermore, the concept does not suspend the possibility that the role of the PCA is carried over by the manufacturer or the provider 4.1. Then, the certificates are only valuable for finding the correct communication partner but not for obfuscating the identity of a TMS. Therefore, Direct Anonymous Attestation is not considered as reasonable solution.

To guarantee the privacy of transmitted Measurement Records the whole content can be encrypted. For this, a hybrid encryption scheme can be used. Initially, a migratable symmetric key must be created by the TPM. The TMS signs the Measurement Records with this symmetric key. And then, the symmetric key is encrypted with an asymmetric key pair. Both, the encrypted MRs and the encrypted key, is transmitted to the AEU. Alternatively, a symmetric key can be inserted in the TPM during the manufacturing process or before the establishment of the system. The key is protected through the TPM so that the AEU can be sure that the key was not compromised. A copy of the key is stored at the AEU. For protection at this side the user itself is responsible. The MRs are encrypted with the key and transmitted to the AEU. Both techniques warrant the privacy of the encrypted data but not the loss or the broaching of data. For this, secure and authenticated transmission channels need to be established.

To provide authenticity of the transmitted data signatures can be utilised. When we sign a statement we want to know what was signed not that later a totally different content can be claimed. That is the reason why in this case the encryption of data must be carried out after adding a signature. This is important since the authenticity

and privacy of data must be provided. Authenticity can only be achieved through first signing and then encrypting data. Data must be in a readable format so that the operator later can be sure having signed the correct content. The operator cannot be hold liable for the signature of a foisted content. If the signature would have been carried out over a cipher text it would be as if the operator signed a contract he has never read. First signing and then encrypting offers the possibility for everybody to verify the signature. With encrypting first, only the person who owns the decryption key can verify the signature since only the key owner knows what was signed. If data would be encrypted first, the content could not be reproduced without the correct decryption key. Hence, obfuscated data would be signed of which only the decryption key owner exactly knows if it consists of the right content. It cannot be guaranteed which content is signed. The signature over encrypted content just provides the assurance that the cipher text was authentic.

4.6.5 Miscellaneous

The v1.2 TPM Specification defines new features of the TPM. So far, only the Tick Counter was considered as a key feature used to establish non-repudiation of the Measurement Records. In this section, some more features are contemplated. It needs to be awaited if and how these concepts are realised in future.

Locality is one of these new features. Given that there exist hardware and running software that creates measurement data outside the TPM these entities cannot be trusted at all events. Now, it might be convenient to assign to different software and hardware parts different levels of trust. E.g. the internal hardware sensors might be trusted far more reaching than external sensors transmitting their data wireless. Moreover, different software applications might not be allowed to access special parameters. This can be controlled by security levels. For this reason, it might be applicable to assign different components a different security or trust levels. Locality exactly addresses this idea. Locality specific PCRs are introduced to save external process characteristics.

With the delegation concept an operator of the Traffic Monitoring System has the opportunity to give a software module the ability to use only a subset of owner authorised commands, without granting permission to any other TPM commands. This is especially in the context of trusted running applications important. Using the dynamic boot process, the integrity of the system is checked permanently. If applications are restricted in their usage of commands, particularly owner authorisation commands, this distinctively decreases the amount of information to be checked for its integrity. Only those applications which are allowed to make changes on the measurement data can be granted access through a subset of owner authorisation commands. These are exactly defined for their own purpose. And do not grant any more permissions.

The necessity of securing the data paths is obvious. Measured data shall not be changed on any path from its place of origin to its terminal. The v1.2 TPM Specification includes some general purpose I/O function. The I/O functions serve to protect

the communication between the TPM and other platform components. Particularly, this protects the key usage what is strongly desired. Since the trustworthiness of MRs strongly depends on the security level of the used keys.

With delegation and locality more flexible authentication methods are imaginable. Authentication is combined with both features. Surely, a more restricted authentication procedure would increase the trustworthiness of the system. It must be seen if these features are realisable in a distinctive system architecture.

5 Implementation and Security Analysis

A demonstrator application is implemented for proving the feasibility of the presented concept. The demonstrator is not a fully implementation but shows the achievability of the main features. The Java programming language was used to implement the demonstrator application. As programming environment Eclipse with a JDK 1.6 Compiler was used. Testing was done with the help of a VM running on a separate server.

Figure 22 illustrates the main implemented components that realise the major TMS functionalities. The `apps.MeasuringInstrument` class is the master class containing the TPM and several other classes to implement functions like key creation, time stamping and Measurement Record creation. For the key creation the newly implemented `tools.MyCertifyKey` and `tools.MyCreateKey` classes in addition to the ethemba `PCAclient` class are available for the MeasuringInstrument. The `tools.TickStampUnit` class is accessible for the MeasuringInstrument for all time stamping processes. And the `xml.MyXMLBuilder` class was implemented to create a XML-file representing the desired Measurement Record. Additionally, the whole implementation consists of further classes helping to carry out subprocesses. In the subsequent sections, first the necessary implementation framework is described. Then, some selected implemented components classes are described.

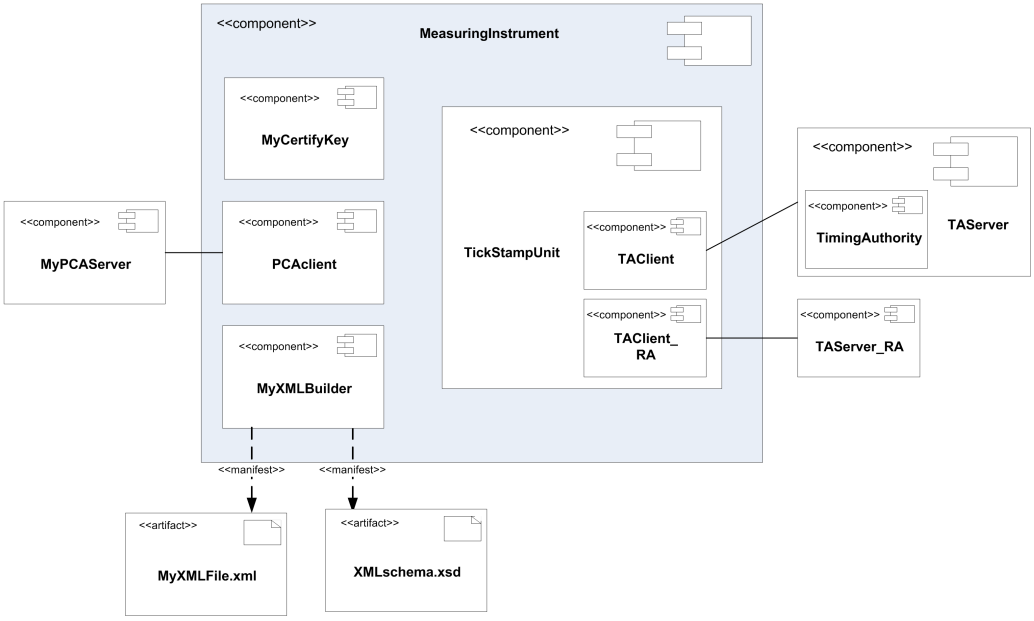

Figure 22: Implemented Components

5.1 Implementation

5.1.1 Used Components

TCG Software Stack and TrouSerS:

Trusted Computing comprises multiple layers of hardware and software. Whereat, the TPM and the trusted building blocks form the hardware part. The TPM hardware driver and a Trusted Software Stack form the software part. A ***TCG Software Stack (TSS)*** is one of the main software building blocks supporting the platform's TPM. The TPMs capabilities are restricted due to performance and security reasons. In separating the functions requiring protected TPM storage from those that do not and outsourcing the latter functions to the platform's main processor and memory space the restricted capabilities of a TPM can be handled. The TSS includes all components supporting these functionalities. [75, 86]
A TSS is divided in four layers (a top down listing) 23:

- TCG Service Provider (TSP) and TCG Service Provider Interface (TSPI)
- TSS Core Services (TCS) and TSS Core Services Interface (TCSI)
- TCG Device Driver Library (TDDL) and TCG Device Driver Library Interface (TDDLi)
- TPM Device Driver (TDD) and TPM Device Driver Interface (TDDi)

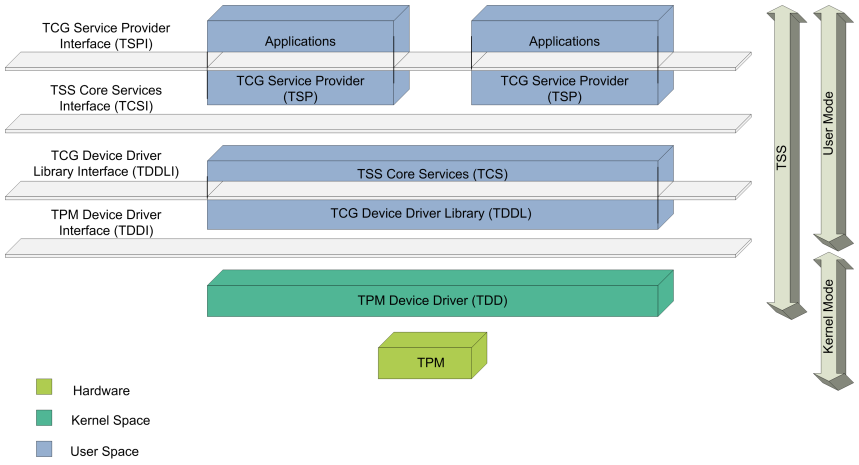

Figure 23: TSS Layers

TSP and TSPI:
Each application that wants to use TPM functionalities needs to use its TSP over the TSP interface (C interface). Every application has its own TSP that is executed inside the application process. Realised as library, the TSP abstracts high level TCG functions used for hashing or signature validation. Two parts of the TSP are the TSP Context

Manager (TSPCM) and the TSP Cryptographic Functions (TSPCF). The TSPCM provides dynamic handles for the efficient usage of the TSP and application resources. The TSPCF offers protected functions of the TPM.

TCS and TCSI:
The TCS serves all platform TSPs and realises multiple access situations to a TPM. Furthermore, it makes functions for storing, administering and protecting keys and secrets available. The TCS is implemented as user space daemon.

TDDL and TDDLi:
The TDDL with its interface provides a uniform interface to different TPM driver implementations. It is an interface between kernel and user mode. The TDDL provides functions to administer the underlying connection to the TDD. For this it supplies functions to read and set attributes of the TPM, TDD and TDDL. Also, it supplies functions for transmitting and cancelling TPM commands.

TDD and TDDI:
The TDD receives messages regarding the behaviour of the TPM. It is the only module that directly communicates with the TPM. It receives byte streams of the TDDL and sends them to the TPM. Then, the TPM answers directly to the TDDL. TDDI is defined as interface in the TCG specification but it is not yet realised. One difference between TDD and TDDL is that the TDDL is implemented in user space. While, the TDD is implemented in kernel space.

TrouSerS is an open source TCG Software Stack (TSS) implementation. It provides an Application Programming Interface (API) for the OS applications provided by the TSS so that the applications are capable of using the TPM functionalities. As long as applications or different OS running on a Virtual Machine Monitor(VMM) comply with the TSS specification TrouSerS allows the use of TPM functionalities. The main purpose of TrouSerS is to control the various admissions to the TPM. Admissions must be managed since the TPM capacity is restricted and the TPM is only able to communicate with one client at a time. The current version is TSS version 1.2 which is compliant with all v1.2 TPMs. [75]

ethemba, jTSS and jTSS Wrapper

For the implementation of the demonstrator application the ***ethemba*** framework [40] was used. This framework provides for a TPM simulation so that a variety of TPM features can be used without actually possessing a real hardware TPM.

As underlying implementation of the TCG Software Stack for the Java programming language ethemba uses jTSS. ***jTSS*** implements all the TSS layers in Java to provide a TCG Software Stack(TSS). On the contrary, the jTSS Wrapper wraps a C stack like TrouSerS. ***jTSS Wrapper*** provides language binding for Java since a TSS usually is developed in the C programming language. Accordingly, the TSS cannot be directly

used from other languages as Java. But jTSS Wrapper aims to make the Trusted Service Provider Interface (TSPI) layer of the TSS available to Java. jTSS and jTSS Wrapper are supported by the European Commission as part of the OpenTC project. The OpenTC project aspire to reduce system-related threats, errors and malfunctions in traditional computer platforms as well as in embedded systems (find more about OpenTC at [10]). Both are developed and retained at the Institute for Applied Information Processing and Communication (IAIK) at Graz University of Technology. [7,100]

Ethemba uses a later revision of jTSS than the one which is needed to implement this demonstrator. Version 0.3 of jTSS used in ethemba does not comprise the tickStampBlob-method in the class iaik.tc.tss.impl.java.tsp.TcHash which is essential for the implementation of the concepts demonstrator. Since the concept uses time stamping as one of the main features to provide security of data this must be proven functioning. jTss 0.4 for the first time implements the tickStampBlob-method used for time stamping and signing a hash value. Consequently, the latest version must be combined with the ethemba framework.

5.1.2 Classes

This section describes associated classes, processes and further details of the implementation of the demonstrator application.

apps.MeasuringInstrument

This class is the main class of the implemented demonstrator application. Here all functions are assembled to build up a Traffic Monitoring System demonstrator. The `main`-method is here the calling method joining all functions. This class can be seen as the TMS itself.

First, the context and the TPM driver is initialized using the `init`-method. Then, all needed AIKs are generated with the `createAllNeededAIKs`-method. That is a default AIK, an AIK for a signing key and an AIK for the time stamping signing key. The signing key is used for all general signing operations. The time stamping signing key is only used for the signing operations related to the `tools.TickStampUnit` methods. Up next, the creation of the just named signing keys is started with the `createAndCertifyKeys`-method. This method also certifies the generated signing keys with the intended AIKs. Now, all keys needed for this demonstration are generated, stored and certified. Henceforth, further processing can be initiated since for all upcoming processes certified signing keys have been generated.

As next step, the acquisition of an initial real time value has highest priority. In consequence, a new instance of the `tools.TickStampUnit` class is created. Using this instances `getInitialTimeValues`-method a connection to a Timing Authority is established and a real time value respectively time stamp is obtained. At this point it can be either chosen to connect to a Timing Authority equally like a PCA or to connect to a TA using remote attestation. The values handed over to the `getInitialTimeValues`-

method must be set either to 'PCA' or 'RA' depending on the chosen Timing Authority alternative. The connection is achieved by calling either the `apps.TAClient` class for the PCA alternative or the `apps.TAClient_RA` class for remote attestation. The TA generates the time stamp, encrypts and signs it. The signature of the time stamp is carried out by the TA with its RSA private key. The time stamp and its signature are sent back to the `TickStampUnit`. There the received real time value is bound with the help of the TPM to a special time interval defined through two TickStampBlobs. All timing values, including the first and the second TickStampBlob as well as the time stamp and its signature, are passed to the `apps.MeasuringInstrument` class (compare to Section 4.6.3 and to the `tools.TickStampUnit` class description 5.1.2).

For the validation if the whole data set has been transmitted each created picture must be saved to PCR 11. This is done with the `saveDataToPCR11`-method. The process is executed in a familiar way by hashing a new value and concatenating it to the old hash value contained in PCR 11.

The `getXMLValues`-method provides the TMS with all values needed for the creation of the XML-file. Besides the real time acquisition this is the next important process. All collected values are saved as String as they are written in this format to the XML-file. The 'str' abbreviation of the variables indicates that they are saved as String. All variables written to the XML-file are listed in table 11.

The next step is to actually create a XML-file which constitutes the mentioned Measurement Record. This is done via calling the `createXMLFile`-method of the `xml.MyXMLBuilder` class (for further information the `xml.MyXMLBuilder` class description 5.1.2). As this is a demonstrator application and we were mostly concerned about the measuring process the transmission process to a remote entity is neglected. Therefore, the secure transmission of the created XML-file is omitted. Finally, the context must be closed using the `close`-method.

It must be noted that the location value mentioned in Section 4.6.2 is not explicitly set. It is assumed that the location can be determined implicitly through the device ID or the Measurement Values transmitted. Aside from that, the created XML-file contains PCR values. That is to explicitly document the system configuration. Out of that reason, the signing keys must not be bound to any PCRs.

apps.TAClient and apps.TAServer

This part describes both the `apps.TAClient` and the `apps.TAServer` class since both are strongly connected and follow a joint TA protocol. Furthermore, it is to mention that both classes are similar to the ethemba `PCAclient` and `PCAserver` classes. Only the main functions are modified for realising the purpose of a Timing Authority.

The `TAClient` can be used to receive a real time stamp from a `TAServer`. The public part of a formerly created AIK is encrypted and sent to the `TAServer` using the

Parameters	**Description**
int deviceID	The device ID of the TMS.
int recordID	The ID of the currently generated record.
String[] strPcrRequest	Quote answer object that contains the data, the external data and the validation data field values.
String strPcr11	The validation value of PCR 11.
String[] strSml	The Stored Measurement Log (SML) of PCR 10 and PCR 11.
String strValueList	A list of all included values or pictures.
String strHash	A hash of all values or pictures.
String strSignature	A signature over the hash.
String strTickStamp[0]	The current tick nonce related to the current ticks value.
String strTickStamp[1]	The current ticks value used for calculating the event time.
String strTickStamp[2]	The current tick rate related to the current ticks value.
Object[] tickStampBlob1[1]	The first TickStampBlob obtained during the real time acquisition process (first boundary defining the time interval). It contains the tick value, the related tick nonce and the related tick rate.
Object[] tickStampBlob2[1]	The second TickStampBlob obtained during the real time acquisition process (second boundary defining the time interval). It contains the tick value, the related tick nonce and the related tick rate.
Object timeAndSig[0]	The time stamp obtained from the TA.
Object timeAndSig[1]	The signature of the time stamp obtained from the TA.

Table 11: MeasuringInstrument Parameters

TAServer's public key. Having received the request, the `TAServer` decrypts and verifies the contents. If the verification succeeded the `TAServer` creates a random nonce that he wraps for the `TAClient`. This is done with the help of the `tools.TimingAuthority` class method `state3_sub`. The response contains two parts:

1. a symmetric session key and the hash of the AIK public key, both encrypted with the EK public key
2. the nonce, encrypted with the session key

The nonce can be decrypted by the TPM with the TPM_ActivateIdentity command. Merely, the `TAClient` possessing the TPM which was used to create the AIK is capable to decrypt the nonce. The decrypted nonce and the data to time stamp is sent back to the `TAServer`. The server verifies the correctness of the nonce. Upon correct receipt, the `TAServer` takes the data and creates a time stamp. Both is encrypted and wrapped using the same scheme as for the nonce. This is done with the help of the modified `state5_sub` method of the `tools.TimingAuthority` class. The response sent to the client consists of five parts:

1. a symmetric session key and the hash of the AIK public key, both encrypted with the EK public key
2. the AES key encrypted with the session key
3. the time stamp encrypted with the AES key
4. the data to be time stamped encrypted with the AES key
5. the signature of the time stamp carried out with the TAServers RSA private key

The AIK is activated by the `TAClient` with the TPM_ActivateIdentity command. The AES key is decrypted and is used to decrypt the time stamp and the data. If data decrypted does not match the formerly sent data the time stamp received is not correct. And a new time stamp must be acquired. The same holds true for the signature validation. If the client is not able to verify the signature a new time stamp must be obtained.

These two methods are implemented to demonstrate the flow of the first part of the time stamping process the real time value acquisition. Figure 24 presents the executed TA protocol.

apps.TAClient_RA and apps.TAServer_RA

The `apps.TAClient_RA` and the `apps.TAServer_RA` classes are both described together since both are strongly connected following a TA protocol with remote attestation. Additionally, it is to be noted that both classes are related to the ethemba `RAclient` and `RAserver` classes. The main functions have been modified for realising the purpose of a Timing Authority with remote attestation. Nevertheless, in the end both classes are definitely distinguishable to the ethemba classes. These classes implement the basic

Parameters	**Description**
RSAPrivateKey TAserverPrivateKey	Contains the private key of the TAServer.
RSAPrivateKey TAserverPublicKey	Contains the public key of the TAServer.

Table 12: TAServer Parameters

Parameters	**Description**
String TAServerIP	Contains the TAServer IP.
int TAServerPort	Contains the TAServer port.
String AIKtag	Contains the label of the AIK used for certifying the time stamping signing key.
String sAIKPwd	Contains the AIK password.
String sSRKPwd	Contains the SRK password.
String sOWNPwd	Contains the TPM owner password.

Table 13: TAClient Parameters

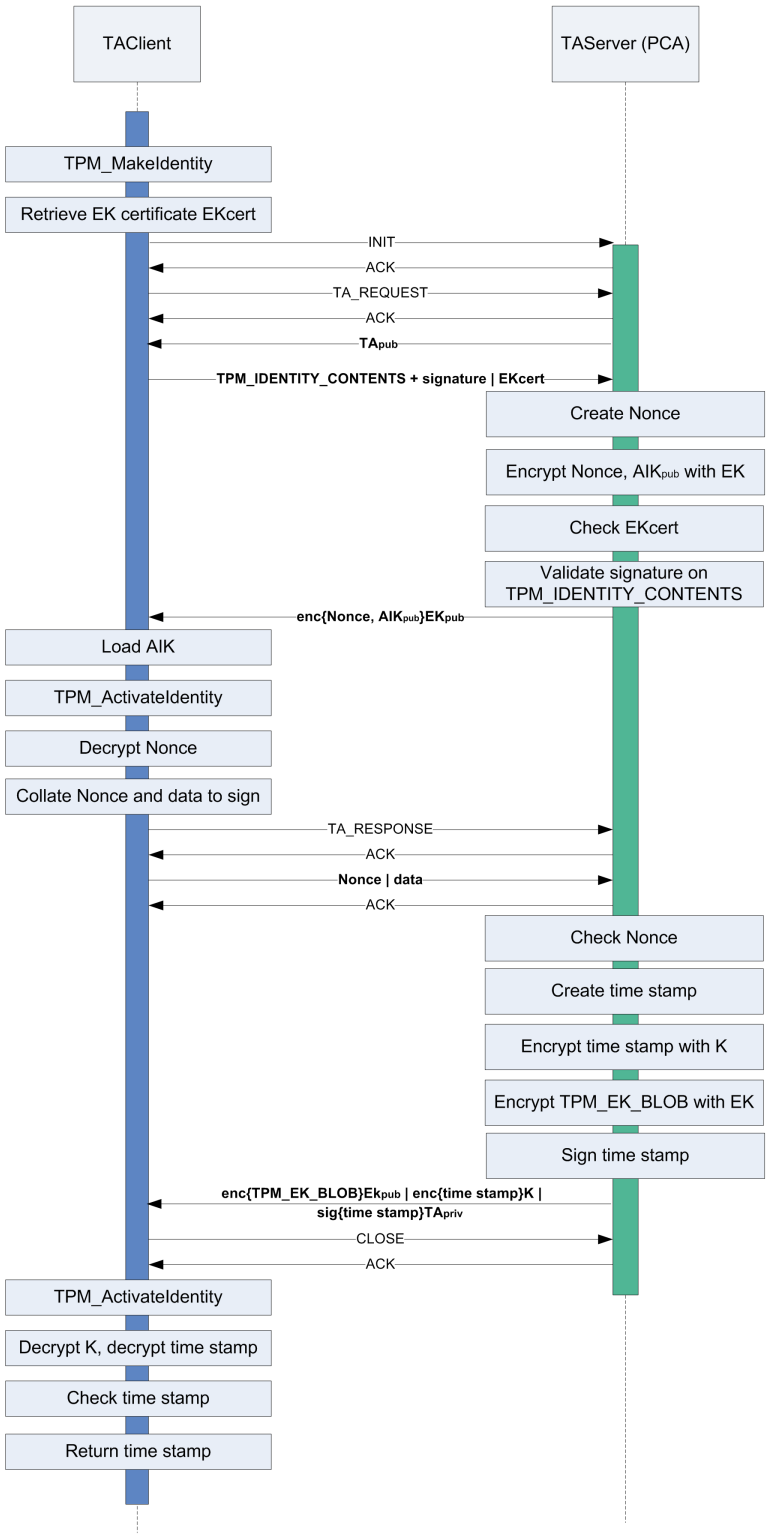

Figure 24: TA Protocol

features but are not yet totally well-engineered. For example, the encryption process is only basically implemented and the validation of the quote is neglected.

The `TAClient_RA` can be used to receive a real time stamp from a `TAServer_RA` which uses remote attestation to identify its communication partner. The client sends the server a time stamp request so that the communication process can start. Having received the request, the `TAServer_RA` creates a nonce and sends his public key back to the `TAClient_RA`. The client responds in sending a quote of PCR 10 signed with an AIK together with the corresponding AIK certificate, the Stored Measurement Log and a session key encrypted with the TA public key to the `TAServer_RA`. Upon, receipt the `TAServer_RA` verifies the certificate, the quote and the SML contents. If the verification terminates correctly the session key is encrypted and an attestation certificate is created. This attestation certificate is encrypted with the session key and sent to the `TAClient_RA`. From this moment on, the client is attested to be qualified to communicate with the TA server. The client decrypts and stores the certificate and generates an encrypted response. The attestation certificate together with the data to time stamp is encrypted using the session key. Together, both form the response. The `TAServer_RA` receives the response and decrypts its contents. Then the server must verify the validity of the certificate and subsequently create a time stamp. Note that the certificate must not be invalid after a short period of time as the clients integrity status might have changed. The time stamp is encrypted together with the received data. This is done to bind the time stamp to one distinct value. Moreover, the `TAServer_RA` creates a signature over the time stamp with its private RSA key. Upon receipt, the `TAClient_RA` must decrypt the time stamp and the related data. The session key is employed to decrypt the time stamp and the data. If data decrypted does not match the formerly sent data the time stamp received is not correct. The process must be executed again. The same applies for the signature verification. If the signature is not correct the whole process must be executed again.

These two methods are implemented to demonstrate the flow of the first part of the time stamping process the real time value acquisition, but this time with remote attestation. Figure 25 presents the executed TA protocol with remote attestation.

tools.CryptoFunctions

The `tools.CryptoFunctions` class provides some cryptographic function like encryption and decryption. These functions are only used by the `apps.TAClient`, `apps.TAServer`, `apps.TAClient_RA`, `apps.TAServer_RA`, and `tools.TimingAuthority` classes and do not necessarily require TPM functionalities.

- `public hybridEncryption(TcBlobData, RSAPublicKey)` – Performs a hybrid encryption on a data blob. A random AES key is generated and encrypted with the given RSA public key. Then again, the AES key encrypts the data blob.
- `public decryptSymmetricKey(SealedObject, RSAPrivateKey)` – Decrypts a symmetric key using the specified RSA private key.

Parameters	Description
RSAPrivateKey TAserverPrivateKey	Contains the private key of the TAServer.
RSAPrivateKey TAserverPublicKey	Contains the public key of the TAServer.
X509Certificate attCert	Attestation certificate created if integrity values of the client are correct.

Table 14: TAServer_RA Parameters

Parameters	Description
String TAServerIP	Contains the TAServer IP.
int TAServerPort	Contains the TAServer port.
String AIKtag	Contains the label of the AIK used for certifying the time stamping signing key.
String sAIKPwd	Contains the AIK password.
String sSRKPwd	Contains the SRK password.
String sOWNPwd	Contains the TPM owner password.
SecretKey sessionKey	AES session key created during the communication process. Used for encrypting communication contents.

Table 15: TAClient_RA Parameters

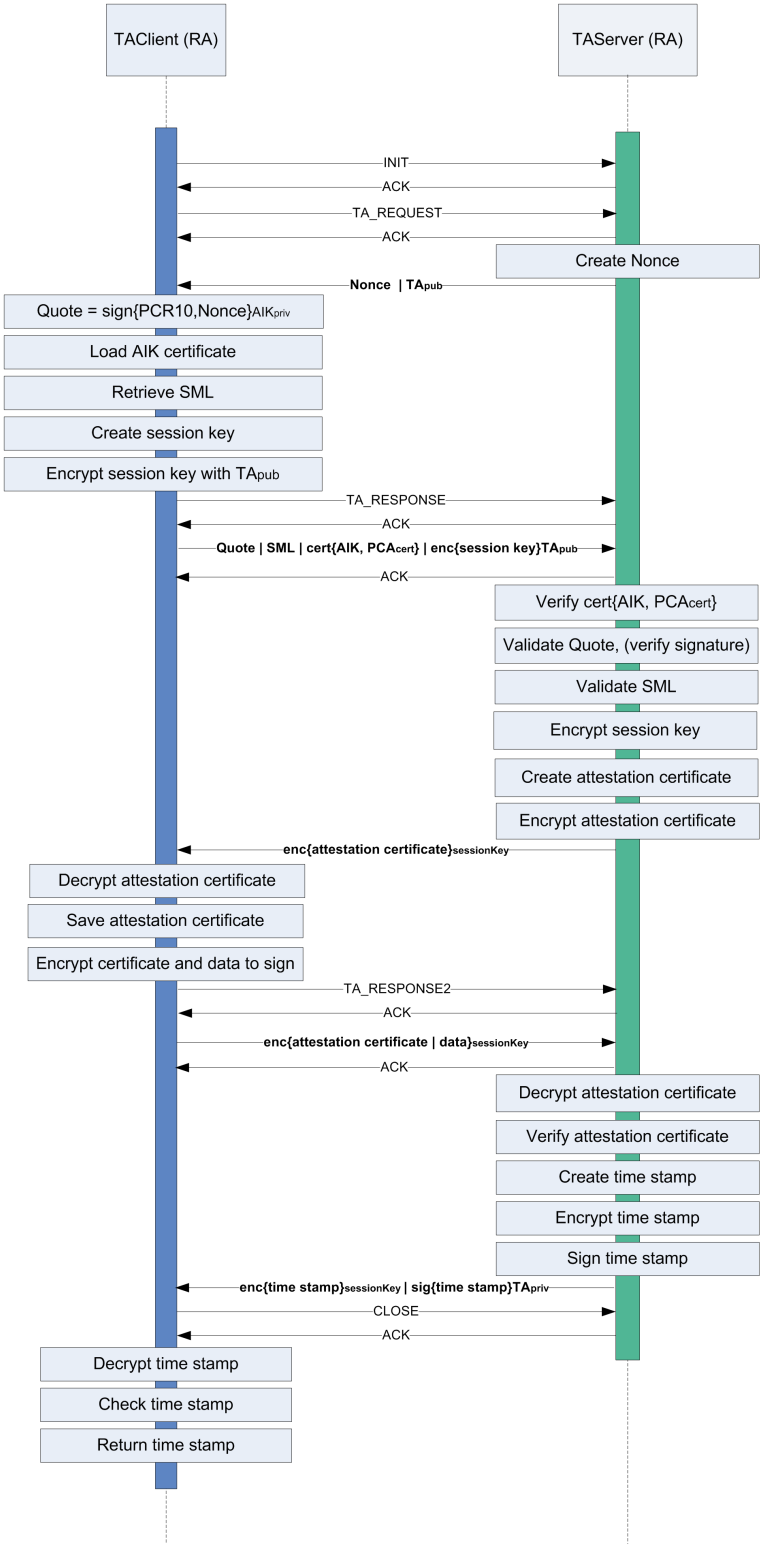

Figure 25: TA Protocol with Remote Attestation

- `public decryptData(SealedObject, Key)` – Decrypts a sealed data object using the given AES symmetric key.
- `public createRSAKey(int)` — Creates an RSA key with a specified key size.
- `public createAESKey(int)` – Creates an AES key with a specified key size.
- `public encryptKey(SecretKey, RSAPublicKey)` – Encrypts a given AES key with the known RSA public key.
- `public encryptData(TcBlobData, SecretKey)` – Encrypts a given data blob with the given AES key.
- `public encryptDataByteArray(byte[], SecretKey)` – Encrypts a given data byte[] with the given AES key.
- `public sign(byte[], RSAPrivateKey)` – Signs the given byte[] with the specified RSA private key.
- `public verify(byte[], RSAPublicKey, byte[])` – Verifies the correctness of a given signature. This is done using the given byte[] and RSA public key.

tools.HelpersForMI

`tools.HelpersForMI` provides some useful methods for data conversion and serialization additional to the ethemba `Helper` class methods.

- `public getByteArrayFromFile(File)` — Returns the content of a File as byte[].
- `public getByteArrayFromMultipleFiles(String[])` — Returns the content of multiple files as byte[].
- `public appendByteArrays(byte[], byte[])` — Appends two byte[] into one.
- `public removeSpaces(String)` — Removes all spaces from a String.
- `public replaceLineBreak(String)` — Removes all \n from a String and replaces them with ” ”.
- `public formatString(String)` — Removes both spaces and line breaks in a given String.
- `public formatStringArray(String[])` — Removes both spaces and line breaks in a given String[].
- `public StringToByteArray(String)` — Converts a String to a byte[].
- `public StringArrayToByteArray(String[])` — Converts a String[] to a byte[].
- `public StringDoubleArrayToByteArray(String[][])` — Converts a String[][] to a byte[].

tools.MyCertifyKey

This class creates and certifies a signing key with the following key attributes:

- TSS_KEY_SIZE_2048
- TSS_KEY_TYPE_SIGNING
- TSS_KEY_NOT_MIGRATABLE
- TSS_KEY_AUTHORIZATION

tools.MyCreateKey

The `tools.MyCreateKey` class creates a TPM signing key with the subsequent key attributes:

- TSS_KEY_SIZE_2048
- TSS_KEY_TYPE_SIGNING
- TSS_KEY_MIGRATABLE
- TSS_KEY_AUTHORIZATION

tools.MyNetCommands

`tools.MyNetCommands` give some additional mappings between symbolic names and command codes to the NetCommands already provided in the ethemba `NetCommand` class. The defined commands are only used in the communication process between the `apps.TAClient` and the `apps.TAServer` classes respectively the `apps.TAClient_RA` and `apps.TAServer_RA` classes.

- TA_REQUEST
- TA_RESPONSE
- TA_RESPONSE2

tools.MySettings

`tools.MySettings` provide some helpful features used for globally setting variables as well as for logging and debugging special events.

tools.TickStampUnit

The `tools.TickStampUnit` class mainly consists of three methods concerned with all kinds of time stamping processes. With the help of these methods it is possible to obtain a real time value from a Timing Authority or to get the current ticks value out of the TPM. The constructor of this class is utilised to set all standard values.

- `public getInitialTimeValues(TcBlobData, String)` —
  This method returns all initial timing values needed to bind the ticks of a TPM to a real time value. These values are the first tickStampBlob, the second tickStampBlob and the time stamp and its signature received from the TA. To be able to return these values, the first TickStampBlob is retrieved using the `getTickStampBlob`-method. As input value for the `getTickStampBlob`-method a nonce is used since at this time it is not important what is signed. The returned current ticks structure object and the signature is saved as the first TickStampBlob. Then, it is necessary to obtain the inital real time value from a Timing Authority. This is done using the `getTimeFromTA`-method. As input values for this method the hash value of the first TickStampBlob and a String defining which TA to enquire ('PCA' or 'RA') are given. The time stamp and its signature will be returned by this method. Whereas, the time stamp will be used next to generate the second TickStampBlob with the `getTickStampBlob`-method. Using the real time value as input for the `getTickStampBlob`-method the last step is taken to bind the real time value to the tick count of the TPM. Finally, the session nonces of both TickStampBlobs are compared. If they match and also the current tick values of the TickStampBlobs are valid the acquisition of the real time value is valid. Otherwise, the whole process must be carried out again as no relation between tick count and real time was established. Then, the generated TickStampBlobs and the time stamp and its signature are returned. It is to mention that at this point no real examination of the current ticks structure can be carried out. This is because the used TPM is only an emulation. Thus, no real hardware-tick-binding is existent which can be used for the verification. The set maximum time exceeded (1 minute) cannot be checked for its correctness.

- `public getTimeFromTA(TcBlobData, String)` —
  This method actually only creates a new instance of a Timing Authority client which communicates with the Timing Authority server. The correct Timing Authority client is chosen through the input String which either equals 'PCA' or 'RA'. Having chosen the PCA alternative an instance of the `apps.TAClient` class is created. If the remote attestation alternative is chosen an instance of the `apps.TAClient_RA` class is created. Both alternatives carry out a real time value acquisition. Finally, the Timing Authority clients' `getTimeStampAndSig`-method must be applied to retrieve the time stamp and the signature that then are returned.

- `public getTickStampBlob(TcBlobData)` —
  This method gets a data blob that it uses to create a TickStampBlob. This is done

by first loading the required keys for signing, creating a validation data object and the hash of the given data blob. Everything is combined forming an iaik TcIHash object which is used to carry out the `TcIHash.tickStampBlob`-method. The `TcIHash.tickStampBlob` returns the required current ticks structure object and a signature over the set hash value carried out with the loaded signing key. Both are used as return values for this method. It is to be noted that at this point of development the jTSS library iaik_jtss_tsp.lib has been modified to suppress the authorisation validation in the `TcIHash.tickStampBlob`-method.

tools.TimingAuthority

This class was copied and modified from `jTPMtools.PrivacyCa`. It is used within `TAServer` to process the collateIdentity request blob received from the `TAClient`. After successfully processing the request a server generated nonce is wrapped inside a blob that only can be accessed by the TPM itself. In a later state of the TA protocol a second collateIdentity request blob received from the `TAClient` must be processed. If processing succeeds an effective time stamp is created. The time stamp is wrapped inside a TPM blob that can only be accessed by the TPM itself. Moreover, a signature over the created time stamp must be carried out. This is done using the TAServers RSA private key. Everything is handed back to the `TAServer`. The server sends it back to the `TAClient` as final answer on the client's request.

The method `state5_sub` was modified to provide a real time value instead of an AIK certificate to the `TAServer`. The time stamp is linked to the data which must be time stamped. The result contains a symmetric session key and the hash of the AIK public key, both encrypted with the EK public key, the AES key encrypted with the session key and the time stamp and the data, both encrypted with the AES key. Then, the `TimingAuthority` sends the generated response back to the `TAServer` which sends it to the `TAClient`. For more complete information on the TA protocol see the `apps.TAClient` and `apps.TAServer` class description.

xml.MyXMLBuilder

`xml.MyXMLBuilder` is the class responsible for the creation of a XML-file. The `createXMLFile`-method uses a formerly generated template to create an XML-file structure. After creation finished the newly created file is compared against a pre-defined schema to detect any errors and to reject data if it is in a wrong format. The XML_TEMPLATE variable globally sets the location of the template used for the basic structure generation of the XML-file. The SCHEMA_NAME variable globally sets the location of the schema which is used to compare the created XML-file.
`createXMLFile` gets its input values as a String[] and sorts the correct values in the correct XML-file field. Various setter-methods help to structure the processing:

- `public setDeviceID(String)` – Sets the device ID to the correct location.
- `public setRecordID(String)` – Sets the record ID to the correct location.

Parameters	Description
String[] xmlValues	Contains all values needed for the creation of the XML-file.
String SCHEMA_NAME	Contains the location for the schema. (Globally defined.)
String XML_TEMPLATE	Contains the location for the XML-template. (Globally defined.)

Table 16: MyXMLBuilder Parameters

- `public setPCRvalues(String, String, String, String)` – Sets the PCR values to the correct location.
- `public setSML(String, String)` – Sets the Stored Measurement Log to the correct location.
- `public setMeasurementValues(String, String, String)` – Sets the measurement values to the correct location. In detail, this means the list of all values, the hash of all values and the signature of the hash.
- `public setTimingValues(String[])` – Sets the timing information to the correct location.
- `public createNodes(NodeList, String, String[], String[])` – Creates a main node and its children for the XML-file.

After setting the values the XML-file is created. Then, the schema is loaded with the loadSchema-method so that the XML-file can be validated against the schema with the `validateXml`-method. Upon success, the created XML-file can be transmitted.

Different Classes

Subsequent classes have been created for testing purposes and for easier handling the ethemba class invocation. As a result, they are similar to or just invoke the related ethemba class.

- `apps.MyTakeOwnership` class – Invokes the ethemba `TakeOwnership` class with some pre-defined values.
- `apps.MyPCAServer` class — Invokes the ethemba `PCAserver` class.
- `tools.MyCertDB` class – Equals the ethemba `CertDB` class and was especially created to test the time stamping and key creation methods.
- `tools.MyTpmKeyDB` class — Equals the ethemba `TpmKeyDB` class and was especially created to test the time stamping and key creation methods.

5.2 Security Analysis

The design of the Traffic Monitoring Systems (TMS) considers the TPM as security anchor. For this reason, if the TPM is compromised the whole system design would be obsolete. The Traffic Monitoring Systems' security analysis is based on conceptual and security assumptions developed at the beginning of the design process. Next, the most important assumptions are summarized:

- A secure environment is offered by the manufacturer.
- Sensors measure and transmit data correctly i.e. integrity, authenticity and privacy of Measurement Values are guaranteed.
- The storage capacity of the device is considered to be adequate high to store all accruing data.
- The AEU stores transmitted data correctly and in the correct environment. So that, no damage or compromise can occur.
- The AEU enforces all security requirements. It can be assumed that this part is not compromised.
- The TA enforces all security policies required to provide a certified time stamp. No compromise of this unit is assumed.

These assumptions define the scope of the design of the TMS. Security risks arising to or out of these parts are neglected. Following the security of the demonstrator TMS application is analysed.

Adressed Threats:

The demonstrator application tries to consider the mentioned threats and requirement of Section 4.2.1. Subsequently, the preparations made are described.

T1, T2 + T3 Security precautions are not considered at this state. These threats must explicitly be considered during the hardware and software realisation of the TMS. At this point, implementing a demonstrator application, it is not possible to take reasonable precautions. Using the ethemba TPM software simulation a trusted boot was included but not adequately tested.

T4 The security relevant data, respectively the Measurement Values, are protected against tampering through the creation of the Measurement Records. A Measurement Record is generated as result of the `apps.MeasuringInstrument` class. The `MeasuringInstrument` methods collect and protect data through the creation of hash values, signatures and time stamps. The hash value over the MVs serves to secure the integrity of data. The signature of the hash value is used to create authenticity of data. And the time stamps are used to determine the actual time of the data creation so that to a later state it can be determined if the

system was in a correct working configuration. Whereat, the system status also has been included in the MR to offer additional evidence for a correct working status of the system.

T5 The authentication and transmission process towards the PCA and the TA is secured through the encryption of data. Data transmitted to these units is encrypted using their public keys or specially created session keys. So, the privacy of data can be achieved. An entity not possessing the correct key can decrypt the transmitted data. For the implementation of the other direction, the entities use the EK public key to encrypt the session key which then is used to encrypt a nonce, an AES key or additional data transmitted. Also, the AES key can be used to encrypt the data necessary to protect. Consequently, only the TPM owner or better the TMS with the correct keys can decrypt the contents. The authentication and transmission process to the AEU is not implemented as the focus was set upon securing data during the editing process. Communication and thus, authentication and transmission are realised through the `PCAclient` and `PCAserver` ethemba classes and the `apps.TAClient`, `apps.TAServer`, `apps.TAClient_RA`, `apps.TAServer_RA` classes.

T6 The completeness of data is protected through appending the PCR 11 hash value to the MR. First, each time a new data set is created the hash value of this data set is saved to the PCR 11. At this point, it is not relevant if reference data values or values generated in the normal processing are saved. Then, during the creation of the MR this PCR 11 value is added. The described processes takes place in the `apps.MeasuringInstrument` class.

T7 It is assumed that the identity cannot be forged since the demonstrator builds upon a TPM simulation software.

T8 Programming errors can always occur even though they should be prevented. For this, some tests were written to prevent at least the obvious programming errors in the demonstrator application. However, a TPM with implemented trusted boot and dynamic trust functionality cannot prevent those failures. It can only prevent the system from known attacks and failures. If inside effects have not been considered to be harming the system they only can be documented and not actively prevented.

T9 Outside effects are not considered at this point. The demonstrator is a pure software based application with the result that no hardware protection can be incorporated.

To conclude this, the Traffic Monitoring System and its demonstrator are designed for documenting any correct and faulty system configuration. Only those failures and threats that are previously defined can possibly prevented in not handing control to the compromised component. However, threats and failures not previously defined and considered during the implementation cannot be recognised by the TPM respectively the TMS and thus, not prevented. Those failures are only reported in the Measurement Records. To the demonstrated extent the desired security goal of non-repudiation is

realised. Authenticity and integrity of data and the system, as well as the distinct association of a time to an event are established. In addition, some primary steps to secure privacy of data are done.

6 Conclusion

In this thesis, a concept for trusted Traffic Monitoring System has been successfully developed. The main idea was to build a TMS that is able to automatically create data admissible as digital evidence in a court of law. It is believed that the produced data withstands all critics in court and thus, is seen as reliable evidence in trials. Always bearing in mind that each country has its own legislation and handles digital evidence differently a variety of security techniques were considered for the construction of a TMS. The chosen technology forming the security anchor for the TMS is a Trusted Platform Module (TPM) as defined by the TCG. The TPM provides all capabilities for the protected creation of digital evidence. To guarantee admissibility, the measured data was secured through hash values and signatures. New is that signatures were utilized to authenticate an instrument but not a person. Moreover, to assure admissibility the collected measurements were furnished with a certified retraceable time stamp and the system configuration status. With the intention that anyone can verify the correctness of the data generated at a specified time. All in all this provides non-repudiation of the measured data and the system and in consequence admissibility at court. Meaning that integrity, authenticity, time and optionally privacy of data as well as the location, integrity and authenticity of the system are granted.

Those parts of the concept helping to establish trust in digital evidence have been realised in a demonstrator application. Mainly, this is the creation of a Measurement Record containing all necessary security information making data admissible in a court of law. It is accompanied by the communication with a PCA as well as the communication with a Timing Authority for the acquisition of a time stamp. The demonstrator application was implemented to substantiate the feasibility of the desired system. Although the demonstrator application fulfils its purpose it still leaks some points. First of all, correlated to the missing hardware realisation, there exist some problems with the hardware emulation as this is not yet fully mature. Also, no sensors were applied to measure data. Therefore, data was obtained from the local file system. As mentioned, the demonstrator does not consider a transmission procedure to a remote entity. Enlarging the coverage of the demonstrator it is desirable to implement secure transmission. Accurate authentication methods using cryptographic keys and correctly implemented certificates can finish off transmission. As privacy only played a minor role in this concept, it was not regarded as the most important security goal. Thus, privacy was only implemented basically. If privacy is in concern encryption in its whole extent must be employed. Another issue is the Stored Measurement Log for other PCRs than PCR 10. The used SML value for PCR 11 is just a randomly generated value since other SMLs do not yet exist. The present SML realisation is simple. For a complete system implementation the full SML functionality must be realised. For SML verification new hash values of known components must be collected. Related to this, the PCR quote could not yet be successfully verified since not all executed components are registered with their hash values. There are surely other matters that can be improved, especially considering a convenient handling of the demonstrator. But within the scope of this thesis the realisation is sufficient to show the achievability of a trusted TMS.

This thesis presents an academic view on the security of digital evidence what required to take some assumptions. The conception rests on a variety of security assumptions which do not hold true invariably. If the assumptions cannot be warranted the system cannot be considered to be trustworthy and to function correctly. For instance, it is assumed that the created instrument is based upon a secure environment. This assumption totally ignores the known attacks on a TPM as they were described in Section 3.6. Not fully developed yet, in an actual realisation further research must be conducted actually eliminating these attacks and other possible threats arising through the system design to warrant the instruments' trustworthiness. Another assumption is that physical attacks on the instrument cannot be carried out unnoticed since the casing is adequately secured through for example seals. Therefore, physical attacks are disregarded. Despite this, the manufacturer must take care of physical attacks and warrant protection against. Since this work is mostly theoretical it remains to be seen how the whole concept is put into practice. Nevertheless, it was attempted to give a new perspective on a technology not yet effectively used.

Giving a brief outlook, in using a trusted computing approach for the realisation of a trustworthy Traffic Monitoring System there arise plenty of starting points for new research topics or a more detailed analysis of already examined aspects. Since the concept was kept mostly general there exist different possibilities to narrow the system functions and thus, creating a more compact one-purpose system. Specifically, when designing a TMS for one distinct country only obeying the local legislation. Depending on the local jurisdiction, methods can be simplified or must be analysed more thoroughly. Apart from this, the conception of a TMS is not the only imaginable field of application of the designed environment. Actually, the system design can be used in a variety of measuring instruments where it is especially necessary to prove the non-repudiation of collected data. Specially, distributed systems that cannot be maintained at all times can profit from this idea. For that reason, other fields of application can be derived. Conceivably are instruments utilised for the remote measuring of consumption data like water, electricity or gas. Households as well as the goods provider appreciate in correct billing. With this concept they can be definitely ensured that the measuring instrument was running correctly and no faults occurred. Another application could be the measuring of emissions in factories. It could especially help to enforce, for example, restrictions on carbon dioxide. Transgressors exceeding the carbon dioxide limits could be penalising as they the collected evidence is admissible at court. On the contrary, dutiful operators could be rewarded. Once more, both sides, the factory operators and the jurisdiction, benefit from a correct measuring. Countries are able to control the restrictions more exactly and forcefully but also operators get a worthy reward for obeying laws and save the environment. Certainly, there exist additional fields of application not yet mentioned. This demonstrates that there are plenty of new research topics in the area of trusted computing and digital evidence worth focusing on.

7 List of Abbreviations

ACPO	Association of Chief Police Officers
ACPI	Advanced Configuration and Power Interface
AEU	Archiving and Evaluation Unit
AIK	Attestation Identity Key
ANPR	Automatic Number Plate Recognition
API	Application Programming Interface
BIND	Binding Instructions aNd Data
CA	Certification Authority
CRTM	Core Root of Trust for Measurement
DAA	Direct Anonymous Attestation
DIR	Data Integrity Registers
D-RTM	Dynamic Root of Trust for Measurement
EAL	Evaluation Assurance Level
EFTA	European Free Trade Association
EK	Endorsement Key
FIPS	Federal Information Processing Standard
GRUB	Grand Unified Bootloader
HMAC	Hash Message Authentication Code
IAIK	Institute for Applied Information Processing and Communications
IMA	Integrity Measurement Architecture
LPC	Low Pin Count
MAC	Message Authentication Code
MBR	Master Boot Records
MID	Measuring Instruments Directive
MOR	Memory Overwrite Request
MR	Measurement Record
MV	Measurement Value
NIST	National Institute for Standards and Technology
NMI	National Measurement Institute
OIML	International Organization of Legal Metrology
OS	Operating System
PCA	Privacy Certification Authority
PCR	Platform Configuration Register
PTB	Physikalisch-Technische Bundesanstalt
RFC	Request for Comments
RTM	Root of Trust for Measurement
RTS	Root of Trust for Storage
RTR	Root of Trust for Reporting
SHA-1	Secure Hash Algorithm - 1
SML	Stored Measurement Log
SRK	Storage Root Key
S-RTM	Static Root of Trust for Measurement

TA	Timing Authority
TBB	Trusted Building Blocks
TC	Trusted Computing
TCG	Trusted Computing Group
TCPA	Trusted Computing Platform Alliance
TCS	TSS Core Services
TCSI	TSS Core Services Interface
TCV	Tick Count Value
TDD	TPM Device Driver
TDDI	TPM Device Driver Interface
TDDL	TCG Device Driver Library
TDDLI	TCG Device Driver Library Interface
TIR	Tick Increment Rate
TMS	Traffic Monitoring System
TPM	Trusted Platform Module
TSN	Tick Session Nonce
TSP	TCG Service Provider
TSPCF	TSP Cryptographic Functions
TSPCM	TSP Context Manager
TSPI	TCG Service Provider Interface
TSPI	Trusted Service Provider Interface
TSS	TCG Software Stack
TTP	Trusted Third Party
UTC	Universal Time Clock
VMM	Virtual Machine Monitor
WELMEC	European Cooperation in Legal Metrology (formerly: Western European Legal Metrology Cooperation)

References

[1] Alexander Delp. (n.d.). Privatheit durch Manipulation der Datenqualität. Retrieved March 10, 2009, from Web site: http://www.ipd.uka.de/~oosem/sensor0607/ausarbeitungen/ausarbeitung-delp.pdf.

[2] AuthentiDate. (n.d.). TimeStamp Server SP250. Retrieved April 12, 2009, from AuthentiDate Web site: http://www.authentidate.de/index.php?id=111.

[3] Bundesamt für Sicherheit in der Informationstechnik. (n.d.). Artikel zu Common Criteria - Evaluation Assurance Level (EAL). Retrieved April 26, 2009, from Bundesamt für Sicherheit in der Informationstechnik Web site: http://www.bsi.bund.de/cc/eal_stufe.htm.

[4] Civil Procedure Rules (CPR 35). (n.d.). Retrieved November 20, 2008, from Ministry of Justice Web site: http://www.justice.gov.uk/civil/procrules_fin/contents/parts/part35.htm.

[5] DTRUST. Retrieved April 12, 2009. Web site: http://www.d-trust.net.

[6] IBM Research. (n.d.). Integrity Measurement Architecture (IMA). Retrieved December 8, 2008, from IBM Research Web site: http://domino.research.ibm.com/comm/research_people.nsf/pages/sailer.ima.html.

[7] Institute for Applied Information Processing and Communications (IAIK) TU Graz. (2009). Trusted Computing for the Java(tm) Platform. Retrieved April 26, 2009, from IAIK Web site: http://trustedjava.sourceforge.net/.

[8] nCipher. (2009). nCipher Time Stamp Server. Retrieved April 12, 2009, from nCipher Web site: http://www.ncipher.com/Products/Time

[9] OIML International Organization of Legal Metrology. Retrieved April 26, 2009. Web site: http://www.oiml.org/information/presentation.html.

[10] OpenTC. (n.d.). Retrieved January 10, 2009, from OpenTC Web site: http://www.opentc.net.

[11] Practice Direction (PD 35). (2005). Retrieved November 20, 2008, from Ministry of Justice Web site: http://www.justice.gov.uk/civil/procrules_fin/pdf/practice_directions/pd_part35.pdf.

[12] Stefan Krempl and Peter-Michael Ziegler. (2006). Was ist die Privatheit in der digitalen Welt noch wert?. Retrieved March 10, 2009, from heise online Web site: http://www.heise.de/newsticker/meldung/70736.

[13] The Indian Evidence Act 1872, 1872.

[14] Evidence Act 1938, May 1938.

[15] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, November 1995.

[16] Requirements of Writing (Scotland) Act 1995, 1995.

[17] The European Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures, December 1999.

[18] Electronic Communications Act 2000, 2000.

[19] Electronic Signatures in Global and National Commerce Act, June 2000.

[20] ELECTRONIC TRANSACTIONS ORDINANCE, 2000.

[21] The Information Technology Act, 2000 (No. 21 Of 2000), June 2000.

[22] Verordnung des Bundeskanzlers über elektronische Signaturen (Signaturverordnung - SigV), 2000.

[23] Gesetz über Rahmenbedingungen für elektronische Signaturen (Signaturgesetz - SigG). BGBl. I S. 876, May 2001.

[24] Gesetz über die Erhebung von streckenbezogenen Gebühren für die Benutzung von Bundesautobahnen mit schweren Nutzfahrzeugen (Autobahnmautgesetz fr schwere Nutzfahrzeuge - ABMG), April 2002.

[25] The Electronic Signatures Regulations 2002. 2002 No. 318 ELECTRONIC COMMUNICATIONS, 2002.

[26] Verordnung zur Erhebung, zum Nachweis der ordnungsgemäßen Entrichtung und zur Erstattung der Maut (LKW-Maut-Verordnung - LKW-MautV), June 2003.

[27] Verordnung zur Festsetzung der Höhe der Autobahnmaut für schwere Nutzfahrzeuge (Mauthöheverordnung - MautHV), June 2003.

[28] Evidence Act 1995. Evidence Act 1995 Act No. 2 of 1995 as amended, July 2006.

[29] Federal Data Protection Act, November 2006.

[30] Queensland Evidence Act 1977 Reprint No. 8, July 2006.

[31] Verordnung zur Ausdehnung der Mautpflicht auf bestimmte Abschnitte von Bundesstraßen (Mautstreckenausdehnungsverordnung - MautStrAusdehnV), December 2006.

[32] Intelligenter Sabotagealarm. *S&I-Select: Videoberwachung 2008, Video, Alarm- & Sicherheitssysteme*, page 45, 2008.

[33] Straßenverkehrs-Ordnung (StVO) in der Fassung des Inkrafttretens vom 01.01.2008. Letzte Änderung durch: Siebzehnte Verordnung zur Änderung der Straßenverkehrs-Ordnung vom 28. November 2007 (Bundesgesetzblatt Jahrgang 2007 Teil I Nr. 61 S.2774, ausgegeben zu Bonn am 07. Dezember 2007), January 2008.

[34] Verordnung zur Änderung autobahnmautrechtlicher Vorschriften und der Fahrzeug-Zulassungsverordnung, November 2008.

[35] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161, August 2001.

[36] Arne Ansper, Ahto Buldas, Meelis Roos, and Jan Willemson. Efficient Long-Term Validation of Digital Signatures, 2001.

[37] Association of Chief Police Officers (ACPO). Good Practice Guide for Computer based Electronic Evidence, August 2007.

[38] Philip T. Blythe. Congestion Charging: Challenges to Meet the UK Policy Objectives. *Review of Network Economics*, 3(4):356–370, December 2004.

[39] Matthew Braid. Collecting Electronic Evidence After a System Compromise, 2001.

[40] Andreas Brett and Andreas Leicher. Ethemba Trusted Host Environment Mainly Based on Attestation, 2009.

[41] Danilo Bruschi, Lorenzo Cavallaro, Andrea Lanzi, and Mattia Monga. Replay Attack in TCG Specification and Solution. In *21th Annual Computer Security Applications Conference (ACSAC 2005), 5-9 December 2005, Tucson, Arizona, USA*. IEEE Computer Society, 2005.

[42] Bundesamt für Sicherheit in der Informationstechnik. Common Criteria ISO/IEC 15408.

[43] Bundesamt für Sicherheit in der Informationstechnik. Schutzprofile nach Common Criteria.

[44] David Challener, Kent Yoder, Ryan Catherman, David Safford, and Leendert Van Doorn. *A Practical Guide to Trusted Computing*. IBM Press, Pearson plc, Upper Saddle River, New Jersey 07458, 2008.

[45] L. Chen and M. D. Ryan. Offline dictionary attack on TCG TPM weak authorisation data, and solution. In D. Grawrock, H. Reimer, A. Sadeghi, and C. Vishik, editors, *Future of Trust in Computing*. Vieweg & Teubner, 2008.

[46] P.M. Cheng, S. Shekhar M. Donath, X. Ma, and K. Buckeye. Evaluation of Digital Maps for Road User Charging Applications. 2006. In Proc. 85th Transportation Research Board Annual Meeting.

[47] Inc. Chet Hosmer, President & CEO WetStone Technologies. Proving the Integrity of Digital Evidence with Time. *International Journal of Digital Evidence*, 1(1), 2002.

[48] Ernest Cockle. *Cockle's Cases and Statutes on Evidence*. London: Sweet & Maxwell, 10th edition edition, 1963.

[49] Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice. Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, July 2002.

[50] David N. Cottingham, Alastair R. Beresford, and Robert K. Harle. A Survey of Technologies for the Implementation of National-Scale Road User Charging. *Transport Reviews, Volume 27 Issue 4*, July 2007.

[51] Uta Deffke. Stra{ssengeb¨hr in den Niederlanden: Die Maut, die alle glücklich macht, February 2009.

[52] Barbara Fichtinger, Eckehard Herrmann, Nicolai Kuntze, and Andreas U. Schmidt. Trusted Infrastructures for Identities. In Rüdiger Grimm and Berthold Hass, editors, *Virtual Goods: Technology, Economy, and Legal Aspects. Proceedings of the 5th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods, Koblenz, October 11-13, 2007*, Hauppauge, New York, 2008. Nova Publishers.

[53] Peter Gerstbach and Andreas Tomek. Trusted Computing. July 2003.

[54] Jinhua Guo. Security and Privacy in Vehicular Networks.

[55] Sigrid Gürgens, Carsten Rudolph, Dirk Scheuermann, Marion Atts, and Rainer Plaga. Security Evaluation of Scenarios Based on the TCG's TPM Specification. In *ESORICS*, pages 438–453, 2007.

[56] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest We Remember: Cold Boot Attacks on Encryption Keys. In *Proc. 17th USENIX Security Symposium*, 2008.

[57] Petra Hollweg and Herbert Reinke-Nobbe. Alle unter Verdacht: Die Superradarfalle wird wohl kommen gegen die Bedenken von Juristen und Datenschützern. *FOCUS Nr. 6 (2009)*, 2009.

[58] Bernhard Kauer. OSLO: improving the security of trusted computing. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–9, Berkeley, CA, USA, 2007. USENIX Association.

[59] Markus Kien. Videoüberwachung - Rundum abgesichert. *funkschau*, 23, 2007.

[60] Roderick Moreland Kramer. *Organizational Trust: A Reader*. Oxford University Press, 2006.

[61] Klaus Kursawe, Dries Schellekens, and Bart Preneel. Analyzing trusted platform communication. In *In: ECRYPT Workshop, CRASH CRyptographic Advances in Secure Hardware*. CRASH Workshop: CRyptographic Advances in Secure Hardware, 2005.

[62] Stephen Mason, Philip N Argy, Ruth Cannon, Stephen Coughlan, Robert J Currie, Brian W Esler, Lorna Goodwin, Julien Hofman, Manisha T Karia, Tejas D Karia, David Leung, Iain G Mitchell, Laura O'Gorman, Damian Schofield, Daniel Seng, and Bryan Tan. *Electronic Evidence: Disclosure, Discovery and Admissibility*. LexisNexis Butterworths, first edition edition, 2007.

[63] Chris Mitchell. *Trusted Computing (Professional Applications of Computing)*. IEEE Press, Piscataway, NJ, USA, 2005.

[64] Thomas Müller. *Trusted Computing Systeme Konzepte und Anforderungen*. Springer Verlag, Berlin Heidelberg, 2008.

[65] Richard Nolan, Colin O'Sullivan, Jake Branson, and Cal Waits. First Responders Guide to Computer Forensics, March 2005.

[66] OIML International Organization of Legal Metrology. Radar equipment for the measurement of the speed of vehicles. OIML R 91, 1990.

[67] OIML International Organization of Legal Metrology. General requirements for electronic measuring instruments. OIML D 11, 2004.

[68] John Patzakis. Maintaining The Digital Chain of Custody, April 2003.

[69] Physikalisch-Technischen Bundesanstalt (PTB). PTB-Anforderungen - Messgeräte im Straßenverkehr Geschwindigkeitsüberwachungsgeräte. PTB-A 18.11, November 2006.

[70] Jason F. Reid and William J. Caelli. DRM, Trusted Computing and Operating System Architecture. In Rei Safavi-Naini, Paul Montague, and Nicholas Sheppard, editors, *Third Australasian Information Security Workshop (AISW 2005)*, volume 44 of *CRPIT*, pages 127–136, Newcastle, Australia, 2005. ACS.

[71] A. Menezes S. Vanstone, O. van Oorschot. *Handbook of Apllied Cryptography*. CRC Press, 1997.

[72] Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote Attestation on Legacy Operating Systems With Trusted Platform Modules, 2008.

[73] Geoffrey Schmitt. Acting as an Expert Witness, 2007.

[74] Fred B. Schneider. *Trust in Cyberspace*. National Academies Press, 1999.

[75] Marcel Selhorst, Christian Stüble, and Felix Teerkorn. TSS-Studie Einführung und Analyse des quelloffenen TCG Software Stacks TrouSerS und Werkzeuge in dessen Umfeld, 2008.

[76] Marcel Selhorst, Christian Stüble, and Felix Teerkorn. TSS Study Introduction and Analysis of the Open Source TCG Software Stack TrouSerS and Tools in its Environment, 2008.

[77] Elaine Shi, Adrian Perrig, and Leendert Van Doorn. BIND: A Fine-grained Attestation Service for Secure Distributed Systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P05)*, 2005.

[78] Evan R. Sparks. A Security Assessment of Trusted Platform Modules Computer Science Technical Report TR2007-597. Technical report, Department of Computer Science Dartmouth College, June 2007.

[79] The Sedona Conference. The Sedona Principles for Electronic Document Production (July 2005 Version), July 2005.

[80] Thilo Kootz (Physikalisch-Technische Bundesanstalt). Forschungsvorhaben Radar-Geschwindigkeitsmessanlage. May 2007.

[81] TollCollect. TollCollect. Retrieved March 10, 2009. Web site: http://www.toll-collect.de/.

[82] Transport for London. London Congestion Charging Technology Trials Stage 1 Report Version 1.0, February 2005.

[83] Trusted Computing Group. TCG PC Specific Implementation Specification Version 1.1, August 2003.

[84] Trusted Computing Group. TPM v1.2 Specification Changes, October 2003.

[85] Trusted Computing Group. TCG Infrastructure Working Group Architecture Part II - Integrity Management Specification Version 1.0 Revision 1.0, November 2006.

[86] Trusted Computing Group. TCG Software Stack (TSS) Specification Version 1.2 Level 1 Part1: Commands and Structures, January 2006.

[87] Trusted Computing Group. TPM Main Part 2 TPM Structures Specification version 1.2 Level 2 Revision 103, October 2006.

[88] Trusted Computing Group. TPM Main Part 3 Commands Specification Version 1.2 Level 2 Revision 103, October 2006.

[89] Trusted Computing Group. TCG Specification Architecture Overview Revision 1.4, August 2007.

[90] Trusted Computing Group. TPM Main Part 1 Design Principles Specification Version 1.2 Level 2 Revision 103, July 2007.

[91] Trusted Computing Group. TCG Platform Reset Attack Mitigation Specification Specification Version 1.00 Revision 1.00, May 2008.

[92] United Nations Commission on International Trade Law. UNCITRAL Model Law on Electronic Signatures with Guide to Enactment 2001, July 2001.

[93] U.S. Department of Justice . Forensic Examination of Digital Evidence: A Guide for Law Enforcement, April 2004.

[94] U.S. Department of Justice. Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition. April 2008.

[95] WELMEC European cooperation in legal metrology. Development of Software Requirements. WELMEC 7.1 Issue 2, May 2005.

[96] WELMEC European cooperation in legal metrology. Measuring Instruments Directive 2004/22/EC - Generalities on the Assessment and Operation of Notified Bodies performing Conformity Assessment. WELMEC 8.0 Issue 1, May 2007.

[97] WELMEC European cooperation in legal metrology. Software Guide (Measuring Instruments Directive 2004/22/EC). WELMEC 7.2 Issue 3, May 2008.

[98] WELMEC European cooperation in legal metrology. WELMEC An Introduction. WELMEC 1 Issue 5, January 2008.

[99] Carrie Morgan Whitcomb. An Historical Perspective of Digital Evidence: A Forensic Scientists View. *International Journal of Digital Evidence*, Volume 1, Issue 1, Spring 2002.

[100] Thomas Winkler and Ronald Gregor Tögl. jTSS - TCG Software Stack for the Java (tm) Platform, 2007.

A XML Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
    <xsd:documentation xml:lang="EN">
       Measurement Record schema for the creation of
       admissible digital evidence.
    </xsd:documentation>
</xsd:annotation>

<xsd:element name="MeasurementRecord" type="MRType"/>

<xsd:complexType name="MRType">
    <xsd:sequence>
        <!-- without min/max it occurs exactly once -->
        <xsd:element name="DeviceID"
                type="xsd:positiveInteger"/>
        <xsd:element name="RecordID"
                type="xsd:positiveInteger"/>

        <xsd:element name="data">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="PCRvalues"
                                type="PCRType"/>
                                <xsd:element name="SML"
                                        type="SMLType"/>
                                <xsd:element name=
                                        "MeasurementValue"
                                        type="MVType"/>
                                <xsd:element name=
                                        "TimingValues"
                                        type="TVType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="PCRType">
    <xsd:sequence>
            <xsd:element name="values">
                <xsd:complexType>
```

```
                    <xsd:sequence>
                        <xsd:element name="data"
                                type="xsd:hexBinary"/>
                        <xsd:element name="externalData"
                                type="xsd:hexBinary"/>
                        <xsd:element name="validationData"
                                type="xsd:hexBinary"/>
                        <xsd:element name="pcr11"
                                type="xsd:hexBinary"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SMLType">
    <xsd:sequence>
            <xsd:element name="SMLvalues">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="SML10"
                                type="hashedType"/>
                        <xsd:element name="SML11"
                                type="hashedType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MVType">
    <!-- entweder so: -->
    <xsd:sequence>
            <xsd:element name="seqValues">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="valueList"
                                type="valueListType"/>
                        <xsd:element name="hash"
                                type="hashedType"/>
                        <xsd:element name="signature"
                                type="signatureType512"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
```

```
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TVType">
    <xsd:sequence>
            <xsd:element name="current">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="tickNonce"
                                type="hashedType"/>
                        <xsd:element name="currentTicks"
                                type="xsd:positiveInteger"/>
                        <xsd:element name="tickRate"
                                type="xsd:positiveInteger"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="initial_1">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="tickNonce"
                                type="hashedType"/>
                        <xsd:element name="currentTicks"
                                type="xsd:positiveInteger"/>
                        <xsd:element name="tickRate"
                                type="xsd:positiveInteger"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="initial_2">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="tickNonce"
                                type="hashedType"/>
                        <xsd:element name="currentTicks"
                                type="xsd:positiveInteger"/>
                        <xsd:element name="tickRate"
                                type="xsd:positiveInteger"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="timeAndSig">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="time"
```

```
                                    type="timeType"/>
                            <xsd:element name="timeSig"
                                    type="signatureType128"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<!-- type for hash value: SHA-1 hash is 130 bits long -->
<!-- hexBinary length is counted in octets -->
<!-- ->length = 1 = 8 bits -->
<xsd:simpleType name="hashedType">
        <xsd:restriction base="xsd:hexBinary">
                <xsd:length value="20" fixed="true"/>
        </xsd:restriction>
</xsd:simpleType>


<!-- creates list type for storing the path to -->
<!-- the hashed data -->
<!-- must contain at least one value -->
<xsd:simpleType name="valueListType">
        <xsd:restriction base="xsd:string">
                <xsd:minLength value="1"/>
                <!-- <xsd:maxLength value="1"/>  -->
                <!-- can be uncommented to restrict the -->
                <!-- hash and signature to just one value-->
        </xsd:restriction>
</xsd:simpleType>

<!-- creates the time type -->
<!-- must contain at least one value -->
<xsd:simpleType name="timeType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern
                value="[A-Z|a-z]{6}[0-9]{4}(:)[0-9]
                {2}(:)[0-9]{2}(CEST|UTC)[0-9]{4}"/>
                <!--SatApr1820:08:35CEST2009-->
       <xsd:minLength value="1"/>
       <xsd:maxLength value="29"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- creates the signature type for 512 digests -->
```

```
<!-- must contain at least one value -->
<xsd:simpleType name="signatureType512">
    <xsd:restriction base="xsd:hexBinary">
       <xsd:length value="256" fixed="true"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- creates the signature type for 128 digests -->
<!-- must contain at least one value -->
<xsd:simpleType name="signatureType128">
    <xsd:restriction base="xsd:hexBinary">
       <xsd:length value="64" fixed="true"/>
    </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

B XML Output Example

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!-- MyXmlFile.xml --><MeasurementRecord>
        <DeviceID>1</DeviceID>
        <RecordID>1</RecordID>
        <data>
        <PCRvalues><values>
                <data>
                        0101000051554f541fc00964584435df
                        66d3a03f9df3dac49cbdb8bc0cc8aa98
                        5e5e951051593e8dba80ab1aa318d99e
                </data>
                <externalData>
                        0cc8aa985e5e95105159
                        3e8dba80ab1aa318d99e
                </externalData>
                <validationData>
                        36cdf42ad57d1bd07ca44b20048078c0
                        dc82025589e6c9642e9b4875543257b7
                        938f7ebf8a23d72ad9c70fef069ce29f
                        a36d559f8c6058fefb996165ca97eadc
                        44b675f1368e3c5b5c5b1bb9f0495828
                        5b5a138a602dcccdcc4735e092f29e67
                        e8e8065360273022f4981ef085d94042
                        c444eef821efeda2ab7d7453259f8972
                        82a605e15fc6eb4ab8b4fff372bfdb3e
                        171c81e46302974cf942c0c9df34b459
                        5aae647c0c9b1a9faca85f915ca591a1
                        d546b98f9d2dbc4d63b2ca6f7a41fdb4
                        1bc0665b8d7efc063700b0b41190933e
                        cdb1efeac21b37baaccfc437a60227d9
                        8eb59cb2d78bb96490de8a92310a7a5a
                        5bbb32453780acf4ea3fb67fac487359
                </validationData>
                <pcr11>
                        d7e34b5758e816b9d50f
                        214ac9bd6b555c6ce674
                </pcr11>
        </values></PCRvalues>
        <SML><SMLvalues>
                <SML10>
                        860655a52281472fd91b
                        054e8d6306c6aaeb5ecc
                </SML10>
```

```
            <SML11>
                    7b71a104b80fb47c11f5
                    6fafaa64a351896f5f81
            </SML11>
    </SMLvalues></SML>
    <MeasurementValue><seqValues>
            <valueList>
                /root/svn/trans/MeasuringInstrument.java,
                    /root/svn/trans/MyPCAServer.java,
                    /root/svn/trans/MyTakeOwnership.java,
                    /root/svn/trans/TAClient_RA.java,
                    /root/svn/trans/TAClient.java,
                    /root/svn/trans/TAServer.java
            </valueList>
            <hash>
                    b7213d39a0d8fcd80f2f
                    c113c7be7a1c24c25000
            </hash>
            <signature>
                    065ff060a8debd70590d2e06f22201e33
                    b1cc08123851f21096d47b3c529da1a81
                    91a73dae54d0687453af6bc0a04a1be48
                    b6451856d13ed825435f00b9507b33f0b
                    8965c7dd35d4f94543013d6dd10e91b05
                    5fb96cfbd1c96ce5616085fc9ebc6662a
                    dbbc30a8285dab97ef8a79a2470e814b0
                    aafc8cb5dbe555e053df2e0ade7db85e5
                    5cad0ec0859dfec5a7bb24823ac543053
                    3984eba9a31e4c16c0aa36fb400aa426a
                    8f2f9f9cbb0927d134f95665541cdc840
                    2f1391289ad4b0483a18f6228180d3ca4
                    1f5eb5181d043880ad62bea16baf8faa1
                    1d2ca792db8a16c5874fc9e2da9c432f4
                    1e32b960f4981fa4d6d85f44f7e1c7450
                    911f979ecd7b45b4f
            </signature>
    </seqValues></MeasurementValue>
    <TimingValues>
            <current>
                    <tickNonce>
                            164ccb16439ed1472db3
                            8b202ad8c4b4351734ab
                    </tickNonce>
                    <currentTicks>
                            105748999823
```

```
                        </currentTicks>
                        <tickRate>1</tickRate>
                </current>
                <initial_1>
                        <tickNonce>
                                164ccb16439ed1472db3
                                8b202ad8c4b4351734ab
                        </tickNonce>
                        <currentTicks>
                                105741360548
                        </currentTicks>
                        <tickRate>1</tickRate>
                </initial_1>
                <initial_2>
                        <tickNonce>
                                164ccb16439ed1472db3
                                8b202ad8c4b4351734ab
                        </tickNonce>
                        <currentTicks>
                                105747127714
                        </currentTicks>
                        <tickRate>1</tickRate>
                </initial_2>
                <timeAndSig>
                <time>TueApr2108:21:11UTC2009</time>
                <timeSig>
                        b79d16123a1f7227a34c07d54b04c672
                        aaaedb190bf71a6cd254e5388e127a02
                        4d619827eaac5742fcb90fb48891d9ab
                        ed811025471d7764f09ab88bc56462b1
                </timeSig>
                </timeAndSig>
        </TimingValues>
        </data>
</MeasurementRecord>
```