Diplomarbeit

Trusted Ticket Systems

von
Andreas Leicher

zur Erlangung des akademischen Grades
Diplom-Informatiker

eingereicht bei
Prof. Dr. Detlef Krömker
Prof. Dr. Claudia Eckert

am

13. März 2009

Fachbereich Informatik und Mathematik
Johann Wolfgang Goethe Universität Frankfurt am Main

Fraunhofer Institut für Sichere Informationstechnologie (SIT)
Darmstadt

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen die aus anderen Quellen stammen als solche kenntlich gemacht habe. Diese Versicherung bezieht sich auch auf Zeichnungen, Tabellen und dergleichen.

Frankfurt am Main, 13. März 2009

Andreas Leicher

Vorwort

Die vorliegende Diplomarbeit beschäftigt sich mit dem Einsatz von Trusted Computing Technologien im Kontext von Ticket Systemen. Sie enstand in Zusammenarbeit mit dem Fraunhofer Institut für Sichere Informationstechnologie (SIT) in Darmstadt und der Goethe-Universität Frankfurt am Main. Ich stelle ein Konzept für die Integration von TPM basierten Tickets in das bestehende System Kerberos auf und zeige die Realisierbarkeit des Konzepts an Hand einer Beispielimplementierung.

Zu Beginn erfolgte eine umfassende Einarbeitung in das Thema Trusted Computing, insbesondere auch eine Betrachtung der Funktionen des TPMs. Das TPM, das inzwischen in einer Vielzahl von Rechnern vorhanden ist und auch im mobilen Sektor mehr und mehr an Bedeutung gewinnt, stellt meiner Meinung nach eine gute Grundlage dar, um die Sicherheit bestehender Anwendungsszenarien zu erhöhen.

Als Ziel für das Konzept wählte ich das Kerberos Protokoll. Es findet vielfältige Anwendung und bietet über die vorhandene Infrastruktur viele Anknüpfungspunkte. Durch die weite Verbreitung von Kerberos, insbesondere auch die Nutzung für die Verwaltung von Geräte- und Ressourcenidentitäten, ist das entwickelte *Trusted Kerberos* ein vielversprechender Ansatz für weitere Forschung und Entwicklung.

Zum Thema Trusted Computing, und vor allem zu konkreten Anwendungen, gibt es noch nicht sehr viel Literatur. Die Spezifikationen der Trusted Computing Group waren vor allem für die technische Recherche sehr hilfreich, darüber hinaus bilden die Bücher „Trusted Computing Systeme“ von Thomas Müller und „A Practical Guide to Trusted Computing“ von David Challener eine gute Einstiegslektüre in das Thema Trusted Computing.

Im Anschluss stand der Entwurf und die Implementierung einer Demonstrations- und Entwicklungsumgebung. Gemeinsam mit Andreas Brett entstand die Plattform *Ethemba*. Ethemba stellt ein virtualisiertes System bereit, das mit einem emulierten TPM ausgestattet ist. Diese Architektur erlaubt es, auch ohne einen vorhandenen TPM-Chip, Anwendungen zu entwickeln und zu testen. Neben der Bereitstellung einer TPM Testumgebung wurde auch ein Software-Framework zu Ethemba entwickelt.

Es stellte sich heraus, dass es kaum Ergebnisse gibt, auf die man aufbauen konnte. So zeigte sich, dass das Paket TPM/j, das wir für die erste Implementierung nutzten, nicht die benötigte Funktionalität bereitstellt. Die Wahl des jTSS als neue Basis zeigte sich als

sehr sinnvoll, da damit sowohl die Konformität zur Spezifikation gewahrt bleibt als auch eine Portabilität auf z.B. den Trousers TSS möglich ist.

Bei der Erarbeitung des Konzepts konnte ich auf die Diplomarbeit „Trusted Infrastructures for Identities“ von Barbara Fichtinger und insbesondere auch auf die wissenschaftlichen Arbeiten von Nicolai Kuntze und Dr. Andreas U. Schmidt zurückgreifen.

Da sich Englisch als Sprache für wissenschaftliche Literatur in der Informatik durchgesetzt hat, wurde die Diplomarbeit bewusst in englischer Sprache verfasst, um sie einem breiten Leserspektrum zugänglich zu machen.

Ein großer Dank gilt Nicolai Kuntze und Dr. Andreas U. Schmidt für ihr durchgängiges Engagement und die fachliche Unterstützung bei der Durchführung der Diplomarbeit. Sie hatten für alle Ideen und Konzepte ein offenes Ohr und durch ihre Anregungen und Fragen entstanden viele neue Ansatzpunkte. Darüber hinaus gaben sie mir wertvolle Einblicke in die Forschungstätigkeit. Insbesondere die Koordination des Gedankenaustauschs der Diplomanden erzeugte große Synergieeffekte. Daher gilt mein Dank auch Jennifer Richter, Gökhan Bal und vor allem Andreas Brett. Des Weiteren gilt mein Dank Prof. Krömker von der Goethe-Universität in Frankfurt für die Betreuung meiner Diplomarbeit. Dank gilt dabei natürlich auch allen Korrekturlesern der Diplomarbeit.

Vor allem gilt aber ein besonderer Dank meinen Eltern, Ingrid und Gerhard, sowie meinem Bruder Christian und meiner Freundin Gesine, für die Unterstützung während des Studiums und insbesondere auch in der Zeit der Diplomarbeit.

Zusammenfassung

Die zunehmende Verbreitung von Internet-basierten Dienstleistungen führt zu einem Wandel der Geschäftsbeziehungen zwischen Anbietern und Kunden. Diese Beziehungen werden nicht mehr über physikalisch greifbare Identitäten hergestellt sondern digital abgebildet. Für die Erzeugung und Verwaltung der digitalen Identitäten wird in steigendem Maße auf Identity Management Systeme (IDM) zurückgegriffen. IDM beschränkt sich dabei nicht nur auf die Identitäten von Personen sondern wird in vielen Anwendungsszenarien insbesondere z.B. in der Machine-to-Machine Kommunikation auf Geräte und Ressourcen übertragen.

Einhergehend mit dem mannigfaltigen Einsatz von Identity Management Systemen zeigt sich ein gesteigertes Bedürfnis nach Sicherheit. Darüber hinaus ist es maßgeblich für den erfolgreichen Einsatz von IDM, dass der Schutz der Privatsphäre des Individuums gewährleistet ist.

Trusted Computing ist eine der Schlüsseltechnologien, die es erlaubt, Vertrauensbeziehungen zwischen mehreren Parteien aufzubauen. Durch den Einsatz von Konzepten und Ansätzen des Trusted Computing können die Kernpunkte Sicherheit und Privatsphäre sinnvoll unterstützt werden. Insbesondere der Einsatz des von der Trusted Computing Group spezifizierten Trusted Platform Moduls (TPM) steht hierbei im Vordergrund. Die Kombination von Identity Management und Trusted Computing ist vielversprechend im Hinblick auf die Erschließung neuer Anwendungsfelder.

Die meisten bestehenden Identity Management Ansätze basieren auf dem Konzept softwarebasierter Zugangstokens, sogenannter Tickets, um die Verwaltung der Identitäten zu ermöglichen. Die vorliegende Diplomarbeit soll ein Konzept aufzeigen, das es einem Individuum, d.h. Benutzer oder Gerät, erlaubt, eine Dienstleistung mit einer selbst gewählten Identität in Anspruch zu nehmen.

Der Zugang zum angebotenen Dienst soll unter der Verwendung von Pseudonymen ermöglicht werden. Das Individuum hat die Möglichkeit ein Ticket von einem Identity Provider zu erhalten. Dieses Ticket enthält eine Aussage über die Identität sowie die mit dieser Identität assoziierten Attribute. Durch den Einsatz von Trusted Computing ist es dem Identity Provider möglich, die Systemintegrität zu überprüfen und zu bewerten. Bei erfolgreicher Validierung des Systems wird das Ticket ausgestellt. Das Ticket kann dann wiederum bei dem Anbieter des Dienstes eingelöst werden, um Zugriff zu dem Dienst zu erhalten.

Dabei ist es von wesentlicher Bedeutung, das das Ticket fest an das TPM, und damit an die Hardware des Systems, gebunden ist. Diese Verbindung erhöht die Sicherheit maßgeblich, da die Verwendung des Tickets mit einem anderen, nicht validierten System unmöglich ist. Damit ist ein entscheidender Beitrag geleistet um all jene Angriffe abzuwehren, die darauf basieren, dass Kopien von Tickets angefertigt werden können.

Diese Diplomarbeit stellt zwei mögliche Anwendungsfälle vor, die aufzeigen, wie durch den Einsatz von Trusted Computing Technologie die Sicherheit in Identity Management Szenarien erhöht werden kann. Abgeleitet von diesen Beispielfällen wird gezeigt, welche Bedingungen und Aufgaben ein Trusted Ticket System erfüllen muss.

Das Konzept der Trusted Tickets wird in eine Kerberos Netzwerk Authentifizierung eingebettet. Eine Anpassung der Ansätze der AIK Zertifizierung mit Hilfe einer Trusted Third Party, der Privacy CA, erlaubt es, vertrauenswürdige Aussagen über digitale Identitäten zu treffen. Durch die Verwendung der Ideen zur Systemvalidierung und zur Attestierung eines Systemzustands kann die Ausgabe der Tickets an den vertrauenswürdigen Zustand gekoppelt werden. Dabei kann auf das vorhandene Konzept der Remote Attestierung aufgebaut werden.

Um die Entwicklung einer Proof-of-Concept Implementierung zu ermöglichen, wurde eine komplette TPM-Demonstrationsumgebung aufgesetzt und konfiguriert. Basierend auf dem TPM Emulator, kombiniert mit der Virtualisierungssoftware QEMU, TPM Treibern und existierender Software wurde das *Ethemba* Framework entwickelt.

Ethemba ermöglicht den einfachen Zugriff auf TPM Funktionen, ist in Java geschrieben und basiert auf jTSS. Neben Hilfsprogrammen enthält *Ethemba* Beispielanwendungen zur AIK Zertifizierung und zur Remote Attestation als Client-Server Infrastruktur. Damit bietet *Ethemba* einen Ausgangspunkt für weiterführende Entwicklungs- und Forschungsarbeit im Bereich Trusted Computing.

Die Referenzimplementierung des Trusted Kerberos Konzepts, inklusive der notwendigen Client und Server Anwendungen, zeigt, dass es möglich ist, ein Trusted Ticket System aufzubauen und in eine bestehende IDM Lösung zu integrieren. Es ist damit gezeigt, dass sich durch den Einsatz von Trusted Computing die Sicherheit bestehender Systeme erhöhen lässt und gleichzeitig der Schutz der Privatsphäre gewährleistet bleibt.

Abstract

Driven by the shift from physical to digital identities and the expansion of network based services, the deployment of identity management solutions is increasing. To support the creation, management and possible destruction of digital identities, various solutions exist and are widely promoted and used. The escalating need for identity management raises privacy and security concerns.

Trusted Computing, as a key technology to establish trust between entities, introduces concepts and architectures that can be used to address these concerns. By combining trusted computing technology with existing IDM solutions, new use cases can be supported. Especially the utilization of the Trusted Platform Module (TPM) as specified by the Trusted Computing Group plays a key role in the realization concept.

Most existing identity management solutions include the use of software based assertions, so called tickets, to manage identities. The goal of this thesis is to develop a concept that allows an individual, either a user or a device, to access a service from a service provider with a chosen identity. The concept shall allow for pseudonymous access to the service provider. The individual therefore retrieves a ticket from an identity provider, making an assertion of the identity. The identity provider verifies the integrity of the system, and based on this assessment issues the ticket. This ticket can then be presented to a service provider to access the service. The service provider establishes a direct trust relationship with the identity provider, relying on the assertions that the identity provider makes. Thus, a chain of trust is generated, allowing the service provider to indirectly lay trust in the individual. It is essential for the tickets to be bound to the TPM, and thus the hardware of the system. This increases the security of conventional systems by inhibiting attacks involving copies of issued tickets.

The thesis presents two use cases to show how trusted computing can be used to increase the security in IDM scenarios. Derived from these use cases, it is shown which requirements must be met and how trusted tickets can be used to represent digital identities.

The idea of trusted tickets gets integrated into a trusted Kerberos environment. By adapting the notions of AIK certification with a trusted third party, the Privacy CA, it is shown how identity statements can be established. Incorporating the concept of system attestation, essentially the network based remote attestation, the presented implementation is able to assess the system state and issue the tickets based on this assessment. The

mechanisms needed to provide measurements and reporting are provided by trusted computing technology.

To enable the development and testing of an implementation, a complete demo environment was set up. Based on concepts of virtualisation, a software TPM emulator and existing TPM drivers and software stacks, the framework *ethemba* was developed. The *ethemba* framework provides high level access to TPM functions, based on jTSS in Java. Besides helping applications, the two main concepts of AIK certification and remote attestation were implemented as demonstrators. The *ethemba* framework provides itself an entry point for a wide variety of further applications.

The reference implementation of a trusted Kerberos protocol, including the necessary client and server applications shows that it is possible to build a trusted ticket system, based on trusted computing technology, increasing privacy and security of existing IDM solutions.

Contents

Chapter 1

Introduction

The increasing significance of identity management solutions reflects the shift from physical to digital identities in service oriented environments. The use of sophisticated identity management systems enables the establishment of identifier domains in which individuals can access services using a certain identifier and the corresponding credentials.

1.1 Problem Description

With the growing use of identity management systems, new problems arise and have to be addressed appropriately. The main worries can be summarized under privacy and security related aspects. To cope with privacy concerns, the use of multiple, partial, pseudonymous identities plays an important role. Without IDM solutions, the user is required to maintain the pseudonymous identities and keep track of their usage. With an established identifier domain, the user is not required to create an account, and thus a new pseudonym, for every service of which he makes use. The identity related information is stored centrally by an identity provider who issues assertions about the individuals identity. These assertions are conventionally embodied in software tickets, which can then be redeemed at a service provider in the domain.

The concept of trust plays an important role in the establishment of the identifier domains. While the service provider is no longer required to maintain a local user management, trust has to be established indirectly. For the establishment of trusted identifier domains several concepts exist.
With the realization of ideas from the Trusted Computing Group, mainly based on the use of the Trusted Platform Module, the thesis shall show how a trusted ticket system can be established as a building block for IDM.

1.2 Motivation and Goals

Targeting the integration of trusted computing ideas into existing IDM solutions, this thesis examines how trusted tickets can be incorporated into an IDM solution. While the core task is to establish an architectural design and appropriate data structures to build

tickets that can be used as authentication credentials, the solution shall include an option to verify and attest system integrity. Relying on the TCG's attestation identity keys, the tickets shall provide means to build pseudonymous identifiers that are tightly bound to a single platform. As a consequence, this concept should increase security, as the credentials cannot be copied and used on another platform. The binding of credential tokens to a system's hardware and state enables many use cases.

With the integration into an existing IDM system, such as Kerberos, the concept shall show how trusted tickets can widely be applied. By building upon previous work and research from Nicolai Kuntze and Dr. Andreas U. Schmidt, the solution shall push forward the usage of trusted computing technology in business use. They showed a general concept to establish transitive trust in mobile scenarios as well as the use of trusted computing for user authentication. The previously published thesis from Barbara Fichtinger, "Trusted Infrastructures for Identities" serves as a base for the establishment of trust between multiple identifier domains.

The decision to build upon trusted computing is based on a variety of reasons. First, with the increasing deployment of the Trusted Platform Module by hardware manufacturers, and the growing acceptance amongst software vendors and users, the trusted computing architecture is already deployed. As of today many laptops are equipped with a TPM, the concepts can be applied to existing infrastructures with minimal costs. Furthermore, as the TCG works on concepts for mobile devices to integrate the TPM functionality in the form of a Mobile Trusted Module, new and different business cases can be enabled using this technology.

The thesis provides a general overview of trusted computing concepts, as well as the architecture and capacities of the TPM. A short introduction to IDM is given and shows possible threats on existing solutions. Kerberos, chosen for the development of a realization concept, is presented in detail. The developed concept is integrated into the Kerberos protocol, and a proof of concept implementation is given.

1.3 Outline

Chapter two provides an introduction to the concepts of trusted computing technologies. In addition to a detailed description of the Trusted Platform Module and its components, the main ideas of AIK certification and system measurement and reporting are discussed. Furthermore, existing software solutions are presented. The main concepts were implemented as a framework in java during the course of this thesis. The ethemba framework has a separate documentation, giving a technical description of the functionality. The chapter concludes with a discussion of security threats and attacks on the TPM.

Identity Management is presented in chapter three. Apart from the discussion of physical vs. digital identity, trust and privacy in the context of IDM are considered. The following discussion of threats on IDM systems nourishes the need for an increase in security.

The fourth chapter presents two use cases, which make use of the synergistic combination of IDM and TC ideas. These use cases shall provide an overview of the potential applicability of trusted ticket systems.

Derived from the use cases, chapter five discusses the requirements for the establishment of a trusted ticket system. It is shown how trusted tickets can be generated and how they are bound to the platform.

The concept for an integration into the Kerberos protocol is presented in chapter six. Beginning with a description of the Kerberos protocol and possible security risks, the realization concept of a Trusted Kerberos protocol is detailed. All three steps in the protocol are described and conceptual details of the implementation are shown. The technical description of the proof of concept implementation is given separately in the Trusted Kerberos documentation in the annex.

Chapter seven analyses the concept, based on the previously discussed requirements and threats. Mainly the topics of privacy and security are covered.

The conclusion given in the last chapter will summarize the main parts of the thesis and provide an overview of possible tasks for future research.

Chapter 2

Trusted Computing

This chapter gives an introduction to the development and fundamental features of Trusted Computing. The Trusted Platform Module is presented in detail and its components are analyzed. Moreover, a brief description of software enabling trusted computing technology is presented. The chapter finishes with a presentation of existing software and a discussion of known security issues.

2.1 Introduction to Trusted Computing

As computer systems play nowadays a more and more important role in private and business environments, the aspects of security and privacy are to be covered more thoroughly. With the growing presence of computer systems in ubiquitous environments such as mobile phones, machine-to-machine communication and sensor networks the need for an increase in security arises. On the other hand, the systems considered are getting more and more complex. The enormous increase in system complexity inhibits a formal verification of the whole system during the development. As a consequence, other means have to be established to encounter the risks and dangers to which every single system gets exposed.

In a networked scenario, where multiple instances communicate, it is important to determine whether a communications partner can be trusted. This is the point where *Trusted Computing* comes into play. First published by the US Department of Defense in the context of *Trusted Computer System Evaluation Criteria* (TCSEC), also known as *Orange Book* [52] a standard for the assessment and certification of computer systems has been released.

2.1.1 Trusted Computing Platform

In order to establish the concepts from Trusted Computing, some changes to the hardware have to be carried out to obtain a *Trusted Computing Platform* (TCP). It can be achieved by adding hardware components, such as add-ons to the mainboard or by extending the capabilities of chipset and CPU. The *Trusted Computing Group* (TCG) shows a possible solution in their specifications of the *Trusted Platform Module* (TPM).

2.1.2 Trusted Computing System

A *Trusted Computing System* essentially consists of the TCP combined with a *Trusted Operating System* (TOS). The TOS, while not being explicitly specified, is way more complex than the TCP. It has to include functions and concepts to transfer trust into the operating system and application level. A TOS can be built around a *Trusted Kernel* and trusted drivers and applications. The ethemba framework [40] shows some progress in the concepts of a TOS.

2.2 The TPM

As an implementation of a TCP, the TCG specifies the *Trusted Platform Module* (TPM) [64]. The TPM is a passive hardware component providing several cryptographic functions, a sealed storage and registers to store measurements of the system state.

In detail, the following capabilities are provided:

- key generation for asymmetric and symmetric cryptography
- signing and encrypting
- calculation of hash values
- key management and storage
- random number generator
- storage for system state measurements
- capabilities to sign and report system measurements
- operations to take control of the TPM, (de)activate the TPM

As the TPM is firmly bound to the system's hardware, it provides a stronger binding than other security tokens (e.g. smart cards) and thus allows for a direct identification of the whole system. As every TPM has a unique identifier it is possible to identify the platform via the TPM, whereas a smart card could be used on different systems thus allowing only for personal identification of a user.

The TPM is specified as a passive element, so that it cannot influence by itself the boot process or the operation of the system. The TPM has to be activated by the owner of the platform, but it is possible to bind certain information to the TPM (e.g. keys for an encrypted harddrive) so that applications or even the operating system will not start unless the TPM is in a predefined state.
Currently TPMs are manufactured by various hardware vendors such as Atmel, Infineon, Broadcom, Winbond and STMicro.
Several vendors provide software solutions that are based on or make use of the TPM, mainly laptop, server and standalone PC manufacturers such as Lenovo, Acer, Asus, Dell, HP, Sony, Toshiba and Fujitsu Siemens Computers.

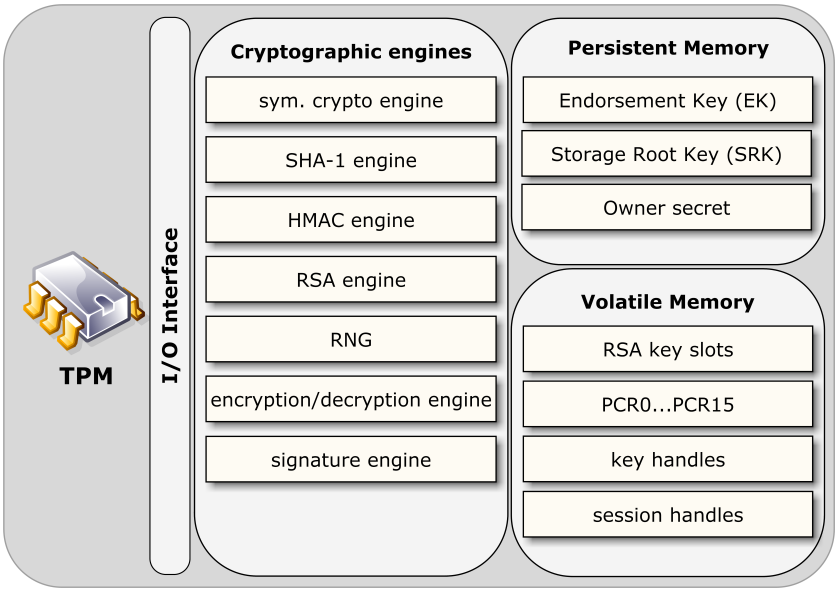

Figure 2.1: Components of the TPM

2.3 Components of the TPM

The components of the TPM can be divided into four logical units. There is an I/O unit responsible for the communication over the LPC bus, the encoding and decoding of messages as well as the routing of the messages to the inner components of the TPM [65, 64].
The core is divided in three main groups, the cryptographic subunits and two storage areas of which one is volatile and the other persistent. The figure 2.1 shows the units and their contents.

2.3.1 Cryptographic engines

Each TPM provides internal functions for cryptographic functions that are performed inside the TPM:

Symmetric Cryptography Engine The TPM can provide symmetric encryption for internal TPM use but is not supposed to expose those functions to general users of the TPM. The symmetric engine is used to encrypt authentication information, transport sessions and for the encryption of data stored outside of the TPM. The TCG specifies only a Vernam one-time pad using 160bit as mandatory implementation. AES can be implemented by the manufacturers.

HMAC engine The HMAC (*keyed-hash message authentication code*) is used to generate a MAC and is used to check the integrity of messages when the TPM communicates

with the host system. A MAC is based on a cryptographic hash function and a secret. The HMAC engine serves two distinct purposes. First, it ensures the integrity of messages sent to the TPM to make sure that no modifications have been made to the command during transit. Secondly, it is required to ensure that the sender of the command is authorized. The session key of the current OIAP/OSAP session is compared to the key the sender used.

SHA1 engine The SHA1 (*secure hash algorithm*) implements a cryptographic hash function that is primarily used by the TPM. It is considered a trusted implementation of the hash algorithm. The functionality is exposed to the outside to support measuring during the boot phase of the system and to allow hash calculations for systems with limited resources. As the TPM is not considered a cryptographic co-processor, there is no minimum throughput requirement for the SHA1 engine in the TPM specification.
The SHA1 engine is mainly used by the HMAC engine to generate MACs and in the process of measuring system states using the PCRs. During the execution of a hash calculation, no other commands are accepted by the TPM. The output of SHA1 is 160 bits long.

Random Number Generator (RNG) The TCG specification requires every TPM to provide a RNG. The RNG is the source of randomness for TPM operations, such as nonce and key generation, and randomness in signatures. The RNG is based on a state-machine that mixes unpredictable data and post-processes the data using a one-way function such as SHA1. The goal is to provide a good source of randomness without the requirement for an external source of hardware entropy. The RNG can be seen as three entities, namely: (1) an entropy source and collector, (2) a state register, holding the state of the RNG and (3) a mixing function to add entropy to the RNG.
The state-machine has a non-volatile initial state with unpredictable random data during manufacturing and can accept further data or entropy at any time. The TPM uses system properties such as thermal noise, keyboard strokes or mouse movements to provide entropy for the RNG.
Using the TPM_GetRandom command, 32 random bytes are returned to the caller. They can then be used as salt for software based random number generators, as they are normally much faster than the TPM's RNG.

RSA engine The RSA engine implements the RSA algorithm in the TPM for encryption/decryption and digital signatures. The TPM provides support for 512, 1024, 2048 bit keys, where 2048 bits are the recommended key size. The provided functions can be used internally and externally. Using the TPM_Sign command, arbitrary data can be signed by the TPM.
The asymmetric de-/encryption of the TPM together with the Storage Root Key provide the means to establish a key hierarchy. The RSA engine also generates secure RSA key pairs using the RNG. The generated keys are then stored and protected in the key hierarchy. Before a generated key can be used by the TPM, it has to be loaded into the TPM

using the LoadKey command. Using this indirection prevents that the private part of the key leaves the TPM.

2.3.2 Persistent TPM storage

The persistent TPM storage allows to store credentials that are not to be changed unless the TPM is reset and a new owner is set.

Endorsement Key (EK) The *endorsement key* (EK) is a 2048bit RSA key pair. It is brought into the TPM by the manufacturer into the non-volatile storage. The TPM can generate the EK internally using the TPM_CreateEndorsementKey. Once an EK is generated, future calls to this command will fail.
To assert that the EK has been generated by the authorized manufacturer, the EK certificate is generated. It contains information on the TPM model and manufacturer as well as the public portion of the EK.
The two parts of the EK, public PUBEK and private PRIVEK are unique and allow for identification of the platform. The PRIVEK may never leave the TPM, and the PUBEK can be read by the TPM_ReadPubEk command. Due to its nature, both key parts have to be considered sensitive in the context of privacy and security. The TPM_ReadPubEk command therefore requires owner authentication.
To deal with privacy concerns, the EK cannot be used to sign arbitrary data. For this purpose, an indirection is used by the introduction of AIKs that can be used for the generation of signatures. The EK is only used to decrypt the owner password in the TPM and it is used to sign and decrypt messages during the AIK creation process.

Storage Root Key (SRK) The *storage root key* (SRK) is a 2048bit RSA key building the root of trust for storage. It is generated inside the TPM during the TakeOwnerShip of the TPM and stored in the protected TPM storage. The SRK is required to be held in the internal storage and similar to the EK, the SRK may not be stored outside the TPM. Figure 2.2 gives an overview of the key storage hierarchy. The SRK is used to protect the key hierarchy of externally stored keys and data by encrypting them.

Owner secret The *owner authorization secret* is the 160bit hash value of the owner password, which is set during the TakeOwnerShip process. It is stored securely inside the TPM and encrypted using the PUBEK. It is thus bound to the TPM which can decrypt the secret when a command requires the owner authentication. The user is then asked to enter the owner password and the TPM can internally compare it to the stored value. Upon successful authentication, the command will be executed by the TPM.

2.3.3 Volatile TPM storage

The volatile TPM storage provides an area to store information which is discarded when the system reboots.

Key slots The TPM provides key slots which are used to load keys for cryptographic operations such as signing and encryption tasks. If a key that is stored in the key hierarchy (see section 2.3.5) is to be used by an operation it is first loaded into the TPM. As all keys are encrypted with their parent keys in the key hierarchy, all required keys are loaded subsequently. They are then used to decrypt the desired key inside the TPM. This concept ensures that the private part of the keys is protected by the key hierarchy and will not leave the TPM in plaintext. If a key is no longer used, it must be unloaded from the TPM to free up space in the key slots.

Platform Configuration Registers (PCRs) The *Platform Configuration Registers* (PCRs) are 160bit storage registers for the integrity protection of system configuration measurements. According to the specification [64] each TPM must provide at least 24 PCRs, shielded inside the TPM. They are used to store SHA1 hash values of measured data during system startup and runtime. In order to protect the integrity, it is impossible to overwrite an existing PCR value. The specified usage of the PCRs is shown in table 2.1. Each PCR is designed to hold an unlimited number of measurements, which is achieved by the *extend* operation. All updates are hashed to the PCR using the equation 2.1.

$$\text{PCR}_i\,new = \text{SHA1}\,(\text{PCR}_i\,old \,||\, \text{SHA1}\,(\text{data or file to be measured})) \tag{2.1}$$

The *extend* function has two main properties: (1) ordering, so *extend* operations are not commutative and (2) it is a one-way function. Thus it is impossible to calculate the input of an *extend* operation from a given $\text{PCR}_i\,new$ value. To track the order of the *extend*

PCR index	PCR value
0	CRTM, BIOS
1	Host platform configuration
2	Option ROM code
3	Option ROM configuration and data
4	IPL Code (usually MBR)
5	IPL configuration and data
6	state transition and wake events
7	host platform manufacturer control
8-15	to be used by the static operating system
16	debug
17-23	localities and dynamic operating system

Table 2.1: Usage of PCRs [58]

operations, a *Stored Measurement Log* (SML) is kept by the system. Whenever a program or file is measured by the TPM and extended to the PCR, its hash value and the path to the file is written to the SML. The SML can then be used to verify which programs where started and the order in which they were called. The SML is used in the remote attestation process as described in section 2.7.2.

2.3.4 Command and Object Authorization

Every communication with the TPM has to be authorized. Therefore, a secure communication channel has to be established. The user has to provide a password to authenticate and authorize. Every object (such as keys and commands) can be protected by a password, e.g. the owner secret.
There are two protocols to access the TPM:

OIAP The *Object Independent Authorization Protocol* allows to establish a communications channel that does not depend on the TPM object that will be accessed. It can be used to prove knowledge of multiple authorization secrets associated with different TPM objects. Once the channel is established, it can be used until it is terminated by either the user or the TPM.

OSAP The *Object Single Authorization Protocol* is used to establish a communication context to a single TPM object. Multiple commands can be given, but only one single object inside the TPM can be targeted.

2.3.5 Keys

All keys are held in the in the TPM key hierarchy and protected by the *Root of Trust for Storage* (RTS). A small amount of keys can be loaded into the TPM key slots and then be used for signing and decrypting operations. All inactive keys are stored to an external storage device. When they are stored externally, the keys are encrypted in a key hierarchy.

The SRK encrypts the first level of the hierarchy. Additional storage keys can be used to establish new levels in the key hierarchy. Each key at level *n* is encrypted using the parent key at level *n-1*. In order to access a key, the private part of the corresponding parent is loaded into the TPM and decrypted inside it. This process has to be done recursively until the SRK is reached. Finally, the key is then loaded into one of the TPM's RSA key slots. Thus the SRK provides the root of trust for storage for the keys. The private part of any key will never leave the TPM and is thus never revealed to another entity.
The *Key Cache Manager* is responsible for the management of the key slots and builds an interface to the external storage.

There are two main key attributes defined by the TCG, *migratable* or *non-migratable*. A *non-migratable* key cannot be transferred from one TPM to another. This attribute is set during key creation and cannot be altered. A typical example of a *non-migratable* key is an AIK. They cannot be moved to another TPM.

The EK and SRK keys are embedded in the TPM and cannot be removed from the TPM. However, a new SRK is created during the *TakeOwnerShip* process. When a new SRK is created, all data encrypted with the old SRK will be rendered inaccessible.

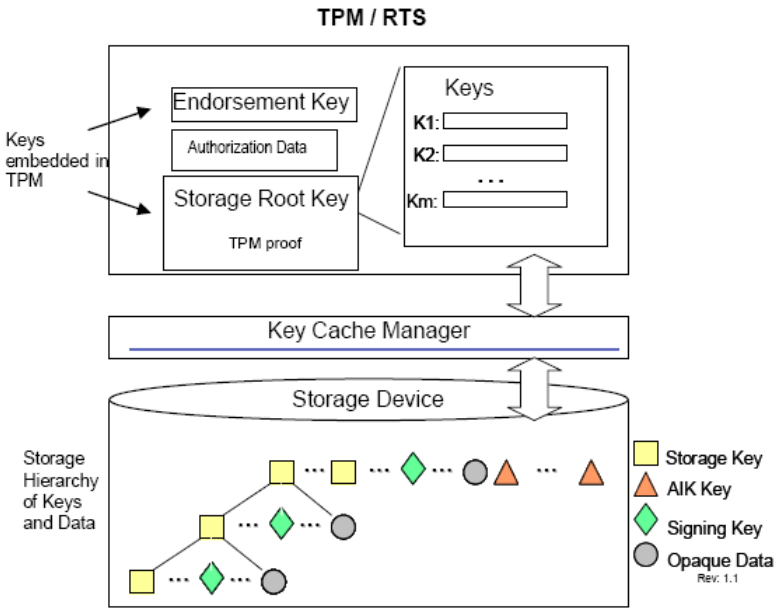

Figure 2.2: Architecture of the TPM key hierarchy [65, p. 17]

In [65] 7 distinct key types are specified, each of them defining different restrictions on the usage of the keys.

Signing Keys can be migratable or non-migratable and are general purpose asymmetric keys used to sign application data and messages.

Storage Keys are another type of asymmetric general purpose keys, used to encrypt data or other keys that are stored outside the TPM.

Endorsement Key is the unique identifier of the platform. It is non-migratable and used to decrypt the owner authorization data and decrypt messages during AIK creation. It cannot be used for encryption or the signing of arbitrary data.

Binding Keys can be used to encrypt data on one platform to be decrypted on another.

Legacy Keys are created outside the TPM and can be imported to be used for signing and encryption. Legacy Keys are always migratable

Attestation Identity Keys (AIK) are non-migratable 2048bit RSA keys that are dedicated to sign data that originates from the TPM and can not be used to sign user supplied data. AIKs are used in the attestation process to sign PCR register values in the TPM quote. They provide a possibility to authenticate the TPM without revealing the EK

and thus the real identity of the platform. An arbitrary number of AIKs can be generated. As key generation has to take place inside the TPM and takes quiet a long time, the TPM generates up to eight AIKs when it is idle.

2.3.6 Certificates and TPM Credentials

There are five different credentials defined by the TCG. They are expected to be in the form of ASN.1 certificates:

- EK credential
- Conformance credential
- Platform credential
- Validation credential
- AIK credential

The relationships and details of the TPM credentials can be seen in figure 2.3 on page 16.

EK credential Issued by the vendor or manufacturer who generates the EK. The EK certificate is used to claim that the EK was properly created and that it is embedded inside a valid TPM. It is to be noted that the PUBEK is privacy sensitive as it allows to identify the TPM and thus the whole platform. As the PUBEK is included in the EK credential, the credential is to be considered privacy sensitive, too. It is used during the request of an AIK certificate and presented to the Privacy CA (PCA). In section 2.6 a detailed description of the AIK certification process is given.

Conformance Credential After evaluating the TPM or the platform containing the TPM, either by the vendor or a third party, the conformance credential states that the TPM implementation is compliant to TCG guidelines.

Platform Credential Issued by the platform manufacturer, this certificate identifies the platform's manufacturer, describing platform properties. Thus the Platform Credential can be considered privacy-sensitive as it refers to a single platform configuration.

Validation Credential Third party components such as processors, video adapters, memory controllers, etc. can be provided with reference measurements from a clean-room environment. These reference measurements can then be compared to the actual runtime measurements of the components so changes to the components can be detected. The vendor signed reference measurements are the Validation Credentials.

Attestation Identity Credential The Attestation Identity Credential is used to attest the private part of an AIK. When a new AIK is created in the TPM using the TPM_MakeIdentity command, a request for certification is sent to a PCA as trusted third party.
The request includes the public part of the AIK, the EK and the platform credential and it is signed using the PRIVEK. After checking the given credentials, the PCA issues a certificate for the AIK, stating that this AIK is held in a shielded location inside a compliant

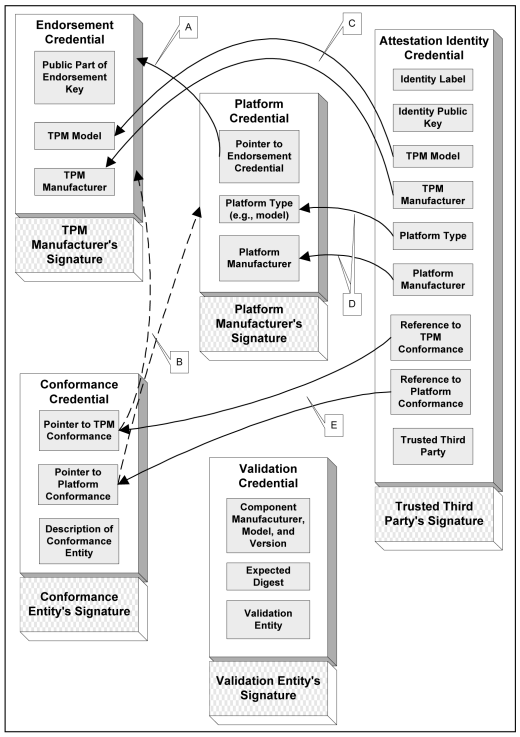

Figure 2.3: TPM credentials and their relationships. [65, p.14]

TPM. As the PCA is the only entity that gains both pieces of information, the public AIK and the EK credential, the PCA is able to link the platform's real identity to the AIK.
The TPM can create an arbitrary number of AIKs, thus it is possible to register an unlimited number of pseudonymous identities, represented by certified AIKs. These pseudonymous identities can only be resolved to real identities by the PCA, and not by any other external party.

2.4 Data Binding

The Tspi_Data_Bind operation [67, p.363] is used to encrypt data using the public portion of a TPM stored key. The encrypted data can later be decrypted with the TPM using the Tspi_Data_Unbind operation.
As the private part of the key used for encryption is not released by the TPM, it is guaranteed that the data can only be decrypted on the platform hosting the TPM. Data binding can be used to encrypt data without a TPM, e.g. by external applications, to encrypt messages in a way such that only the targeted client platform can decrypt it (see server.ExternalDataBinding in ethemba [40]).

2.5 Data Sealing

The Tspi_Data_Seal operation [67, p. 299] can be used to encrypt data such that it can only be decrypted using the unseal operation (Tspi_Data_Unseal) of the TPM. The data is encrypted using the public portion of a TPM stored, non-migratable key.

In addition to the data encryption, the sealing of data allows to include a statement about a PCR configuration the platform must be in to decrypt the data. Thus it is assured that the data will be revealed only if the platform is in a predefined state.
The seal operation can be used to protect authentication keys. For example, if a file is encrypted using key k, sealing this key k to a set of PCR values ensures that this file can only be read when the system is in the desired, trusted configuration.

2.6 AIK Certificate Provisioning

If the client wants to certify an AIK using a PCA as trusted third party, the client first creates a new AIK using the TPM_ActivateIdentity command. The TPM then generates a IdentityRequest, including the public AIK and the EK certificate. The contents are encrypted using the PCA's public key. The PCA server's public key can be provided by a PKI. This ensures that only the trusted PCA can decrypt the request contents.

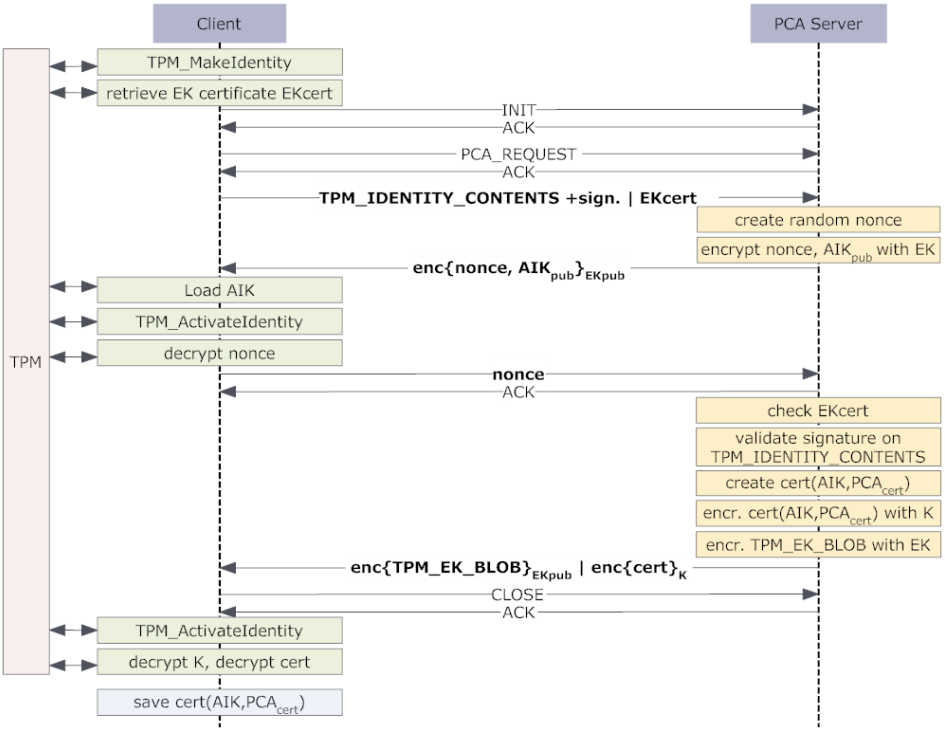

Figure 2.4: Flow of the AIK Certification Protocol [40, p. 13f]

In order to ensure that the AIK is stored in a real TPM, the PCA generates a random nonce and encrypts the nonce with a symmetric session key. The symmetric session key is encrypted using the EKPUB, provided in the EK certificate. Only the TPM, that is in possession of the EKPRIV is able to decrypt the session key (using the TPM_ActivateIdentity command) and thus the nonce. Then, the nonce is sent back to the PCA.

Omitting this step enables several attacks on the certification process as stated by Gürgens and Rudolph in [26]. There would be no reliable connection between the information about the EK of the TPM platform and the given AIK. As a consequence, an attacker who gets hold of the EK and the credentials would be able to request certificates for keys generated outside a TPM. This could lead to either a DoS attack on the PCA or, given the PCA has established quota policies for certification, to an invalidation of the TPM.

Upon receipt of the nonce, the PCA

1. checks the nonce
2. verifies the EK and platform credential
3. generates the AIK certificate cert(AIK,PCA)
4. generates a symmetric key *K*
5. encrypts the cert(AIK,PCA) with K
6. encrypts *K* using the EKPUB

The client can then decrypt *K* using the TPM_ActivateIdentity command, which is in turn used to decrypt the cert(AIK,PCA). The process as described by Gürgens and Rudolph [26] and implemented in ethemba is shown in figure 2.4. A more technical description of the implemented protocol can be found in the ethemba documentation [40].

2.7 Integrity Measurement and Reporting

The process of integrity measurement and reporting is an integral part of the TPM specifications as it allows to prove a platforms integrity to a third party. This enables the establishment of a trust relationship between the challenger and the attesting platform.

2.7.1 Measuring the System State

One of the most important concepts of a Trusted Computing System is the generation of a chain of trust. The chain of trust must be without gaps and extend from system boot up to the current system state. All executed instructions and programs must be included. Therefore every component is required to measure and report the following component before executing it. The measurement of the direct successor prevents unmonitored execution of code between the measurement and the actual execution.

This can be achieved by the generation of integrity measurements using the TPM. As described in [57, 29, 44, 48] hash values are calculated using the extend operation (see section 2.3.3) and stored securely in the PCRs. These measurements alone do not provide

the means to assess a system state. In addition to the measurements, a Stored Measurement Log (SML) is generated. Every measurement gets logged to the SML. The SML together with the measurements allows for a later verification of the chain of trust.

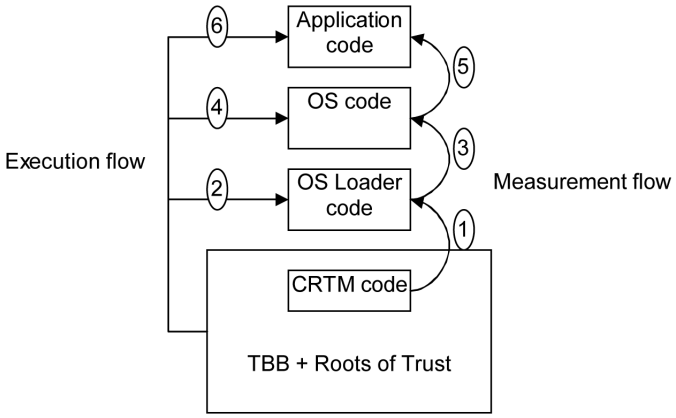

Figure 2.5: Chain of Trust as described by [65]

As a base for the iterative generation of measurements, it is essential to implement a hardware root of trust, the *Core Root of Trust for Measurement* (CRTM). Normally implemented as an extension to the system BIOS, the CRTM contains the first instructions after system start and is able to measure the components that are involved in the system boot. The CRTM can initiate the chain of trust from the BIOS to the MBR.

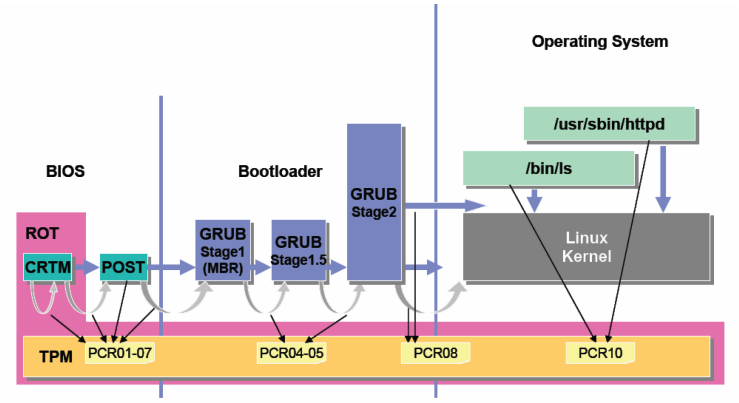

Figure 2.6: Concept of Trusted Grub [58]

The establishment of transitive trust relationships allows to extend the trust boundary. The chain of trust can be divided in two parts, a static and a dynamic chain of trust. The static part is specified by the TCG and is platform and OS independent. As soon as the OS is loaded, it is the responsibility of the OS to continue the chain of trust in a dynamic

way. Several concepts for the extension of the chain of trust exist, such as AEGIS, Copilot, Trusted Grub, IBM IMA [1, 53, 54, 29].

Trusted Grub can be used as gateway between the MBR and the OS. As an enhanced version of GRUB, Trusted Grub allows to measure the stages of the bootloader as well as the initial ramdisk of the OS to be loaded. Thus, Trusted Grub is used to close the gap between the static root of trust and the operating system [44].

Developed by IBM Research, the *Integrated Measurement Architecture* (IMA) [57] allows to extend the measurements to the runtime of the OS. IMA measures every executable instruction upon first load and is implemented as a Linux Security Module in the kernel. Thus, every executed file is measured only once. The SHA1 hash value of the file or program is calculated and stored inside PCR 10, using the extend operation from equation 2.1, to provide an integrity protection of the SML. Due to the nature of the *extend* operation, the measurements are ordered, i.e. alterations in the load sequence will result in a different PCR value. Additionally, the measurements are written in plain text to the SML, an example is shown in listing 2.1.

```
10 9797edf8d0eed36b1cf92547816051c8af4e45ee boot_aggregate
10 ea8239dfed9dd11bd538f9c3234e0d7b71672fff /bin/sh
10 ebb4f3db0b83c1e717e3d05f702e4608a9c2ea08 /lib/ld-linux.so.2
10 671aba5cb6df951463e57c963e8c327fc6cfb5ab /lib/libcrypt.so.1
10 445babe91e586090cb7f9782b44ba115ceec6b7f /lib/libm.so.6
10 411ab19e995e06b1d7378e1daea9926ccba1ea20 /lib/libc.so.6
10 68b297cd8fe07a3e54e6ce9d2e66afa799e472a3 /sbin/depmod
10 407285ba377ea035f2ff6d61c47ead8befa7cd5f /sbin/modprobe
10 b3f9dc0a1cc001f88d43609c7dea156d280deca8 /sbin/udevd
```

Listing 2.1: Example of first lines of a SML generated by IMA

As the IMA module measures every executable only once when it gets first loaded, IMA does not provide a realtime measurement of the system state. The SML makes only the statement that a certain application with the specified hash value has been executed but it does not provide any runtime information.

2.7.2 Reporting and Attestation

Validation and attestation of the system state plays an integral role in TC concepts. It allows to prove a platform's integrity to a third party and thus enables the establishment of trust between the two parties. The often cited statement that the Trusted Computing Platform ensures system integrity must be considered incomplete. The TCP provides means to measure and report system integrity measurements. In order to make a statement about the system state it is essential to assess the generated measurements. This has to be accomplished by an additional entity that is able to compare the given measurements to reference values.
The TCG has defined the two attestation mechanisms of remote attestation and direct

anonymous attestation of which a detailed analysis is given in [7, 48, 57, 56],[47, page 55ff].

Remote Attestation with PCA

In the *Remote Attestation* protocol, the user can attest platform integrity to a verifier. The system state will be reported by a message containing a signed statement of PCR values and the SML. The external verifier will then have to compare the provided data to predefined configuration measurements.
The user has to generate a certified AIK (see section 2.6) that will be used to sign the PCR values describing the system state. This AIK with the certificate provides the proof that the integrity measurements were calculated by a TPM.
Firt, the verifier creates a random nonce and sends it as a challenge to the user. The nonce is used as a replay protection and provides a statement of freshness for the signed PCR values. Using the TPM_Quote command, the PCR values and the nonce are signed with the AIK.

The signed structure is then sent together with the SML to the verifier who is able to check (1) the freshness of the statement, (2) the integrity of the reported measurements and (3) the trustworthiness of the executed programs as listed in the SML. By verifying the AIK certificate and the signature on the PCR values, the verifier can be sure that the measurements were calculated by a certified TPM and have not been altered.

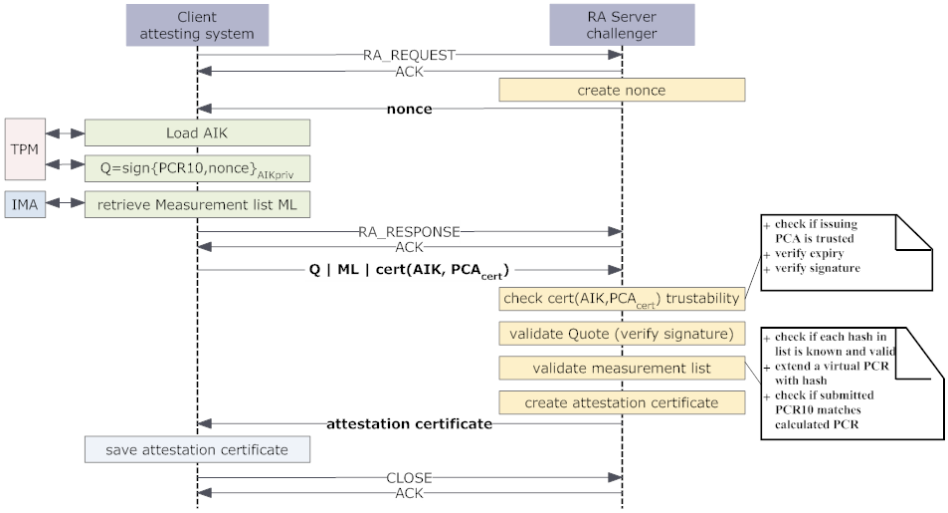

Figure 2.7: Protocol of the Remote Attestation Protocol [40, p. 17f]

In order validate the platform's integrity, two steps are necessary: First, all entries in the submitted SML are compared subsequently to reference values for the given programs or files. Therefore the verifier has to keep and maintain a database of reference measurements. If an entry from the SML cannot be found in the reference database or if the

reported value in the SML differs from the reference value, the platform must be considered compromised. This will be the case if a program has been changed, e.g. by a virus, or if a malicious software has been loaded.

As a second step, a virtual PCR (vPCR) is initialized with zeroes and then extended (using equation 2.1) for all entries in the SML. By comparing the calculated vPCR to the signed value reported by the platform it can be verified that the programs listed in the SML have indeed been executed.
The protocol has been implemented in the ethemba framework [40]. In the documentation of ethemba, a detailed technical description of Remote Attestation is given.

Relay attack on Remote Attestation Stumpf et. al. [62] show a possible relay attack on the remote attestation protocol. The attacker needs to be in control of a malicious and an honest client. He can then forward all attestation queries from the malicious to the honest client reporting a trustworthy state. The response from the honest client is then forwarded to the challenging verifier. This attack is based on the properties of the AIKs used to sign the reported values. To respect privacy, the AIK is not directly linked to the EK and thus does not contain enough information to uniquely identify a single system. It is proposed to create a shared session key that will be used for the encryption of the following communication.

Time-Of-Check vs. Time-Of-Use attack As the current architecture only provides loadtime guarantees the attestation is prone for other attacks. Every program is measured at load time and it is assumed that the code won't change once loaded into memory. If an attacker successfully exploits this time gap he might be able to introduce run time vulnerabilities. This could be achieved in many ways, e.g. by using a malformed input to a program leading to a buffer overflow and thus to further code execution that is not measured. This problem is more thoroughly discussed by Bratus, Sparks et. al. in [4]. They propose to modify the MMU to "seal" memory objects, so that the TPM can become aware of changes in memory.

Direct Anonymous Attestation

Introduced in the TPM specification version 1.2, *Direct Anonymous Attestation* (DAA) aims to increase the level of privacy [7], [20, page 30f].

DAA is a cryptographic protocol, relying on group signatures, that prevents identification of the signer but provides the possibility to detect rogue TPMs.
Three entities are involved in the protocol: the TPM platform, the DAA issuer and the DAA verifier. If the platform wants to perform an attestation, first the DAA issuer has to sign a new key pair generated by the TPM. This can be done by the TPM manufacturer or the distributor. The DAA issuer checks the platform's EK credentials before signing the DAA key.
During attestation, a new AIK is generated and signed with the DAA key. This AIK is used to sign the PCR values needed for attestation. Using a zero-knowledge proof, the DAA

verifier checks that the platform possesses the DAA key without the need to reveal the key to the DAA verifier.
The idea is to increase privacy by letting the DAA issuer gain knowledge about the platform identity and the DAA key, whereas the DAA verifier only knows the AIK. Thus, even if DAA issuer and verifier work together, it is impossible to link two transactions if two AIKs are used.

Flaws in Privacy Protection in DAA Leung et. al [41] discuss a possible privacy flaw in the DAA protocol. The DAA issuer can include covert identity information into the DAA key certificate. Thus, colluding DAA issuers and verifiers could be able to link to one another and identify the platform.

2.8 Selection of existing Software

The following section gives a brief description of existing software packages that have been assessed and used during the course of the diploma thesis. It shall provide a brief overview of available software and reflects thoughts and experiences gained in the practical work. A more in depth discussion and technical description is given in a report from BSI [59].

2.8.1 Trusted Software Stack (TSS)

As the TPM provides only a small amount of storage and computational power, interfaces and capabilities of the TPM are quiet unhandy. Therefore the TCG defined the TSS [67] to provide high level access to TPM functions. The TSS consists of multiple layers that are shown in figure 2.8.

2.8.2 Trousers

Trousers is a open source TSS written in C/C++. It is available at `http://trousers.sourceforge.net/` and provides the TpmTools, a software collection implementing some basic TPM commands. It is noted here for reference but has only been used to test the functionality of the TPM emulator during the course of this thesis.

2.8.3 TPM/j

TPM/j has been developed at MIT and provides an object oriented API in Java for TPM access. It is available at `http://projects.csail.mit.edu/tc/tpmj/`. Although TPM/j provides an easy to access interface to the TPM in Java, it does not claim to be a complete implementation of the TCG TSS. The development seems to be suspended, and the latest release is from April 2007. It was first used as a base for the ethemba framework. The use of TPM/j can not be encouraged due to the lack of some fundamental TPM functions such as TPM_CertifyKey.

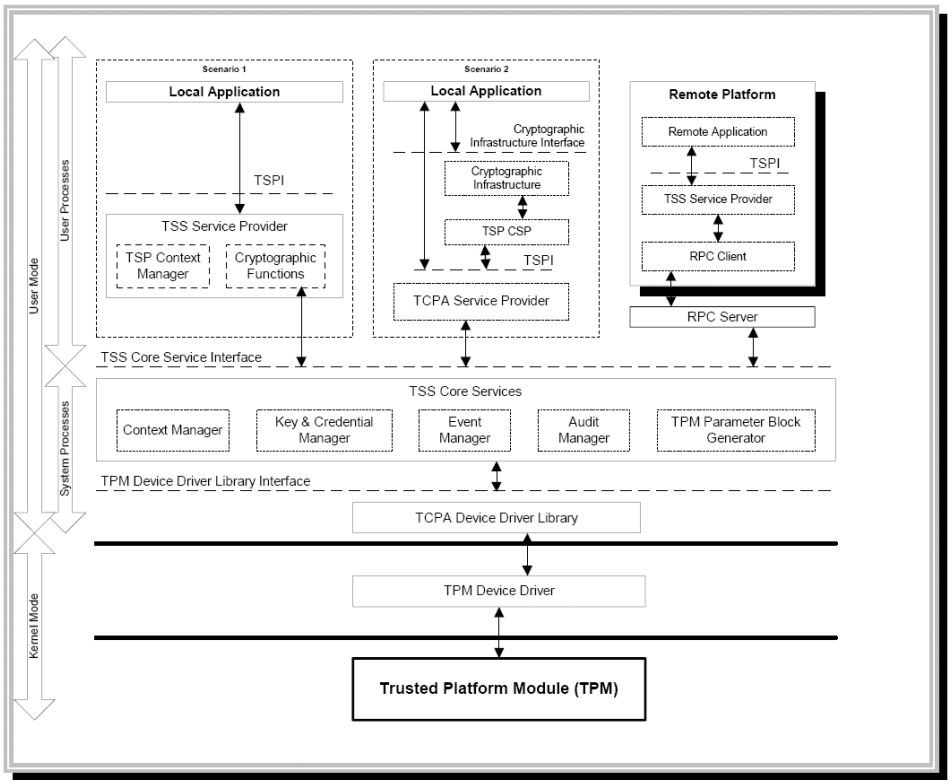

Figure 2.8: The Trusted Software Stack [67]

2.8.4 jTSS

Developed in the course of the EU OpenTC project at IAIK Graz, the jTSS aims to implement a complete and compliant TCG TSS in Java language. It is under steady development and builds the base for the ethemba framework. It is available at `http://trustedjava.sourceforge.net/`, recently version 0.4 has been released. A wrapper for the previously mentioned Trousers TSS is available allowing for coexistence of C/C++ and Java based TPM software.

2.8.5 jTpm Tools

Stemming from the jTSS project, jTpm Tools aims to provide a set of command line tools to access the TPM. Some parts of the jTpm Tools were integrated and reworked for the development of ethemba [40].

2.9 Security Aspects

In addition to the previously mentioned attacks on the DAA and Remote Attestation protocols, the increasing use of TPM technology leads to the exploration of further attacks against the TPM.

2.9.1 TPM Reset Attack

Kursawe et. al present a 'Reset Attack' [39], a method to reset PCR values. This attack leads to default PCR values, which can be exploited by an attacker to produce arbitrary PCR values. As the PCRs are intended to provide integrity protection for the SML, these values can be constructed such that a remote attestation might succeed even if the SML has been tampered.

Evan Sparks provides evidence for this attack [61] and shows a video of the attack on his website [19]. Connected to the system over the Low Pin Count Bus (LPC) the TPM can be reset by connecting the LRESET pin of the LPC bus to ground. This attack requires hardware access to the TPM and the TCG does not claim to provide protection against hardware attacks. As some TPM application scenarios assume that the TPM operates in a hostile, uncontrolled environment, one must be aware of this possible attack.
The physical access to the LPC, which is a key requirement for this attack, will become more and more difficult with the increasing integration of the TPM in other components such as chipsets (e.g. Broadcom integrates a TPM in an ethernet controller [8]).

Bernhard Kauer [32] advises to use a dynamic root of trust for measurement (DRTM). The DRTM removes the BIOS and boot loader from the chain of trust. By making use of modern processor technologies, the CPU executes a small secure loader which is measured into a reset PCR 17. The TPM can distinguish between a reset and a DRTM since a reset would initialize PCR 17 with '1'. Thus, the hash of the secure loader can only be put into PCR 17 by the atomical CPU operation. He implemented the Open Secure

Loader (OSLO) to show how this concept is supported by processors from AMD and Intel.

2.9.2 Cross Certification Vulnerability

If a user calls the TPM_CertifyKey command to obtain a certificate for a key k using AIK_s as a signing key, the attacker intercepts the command and replaces the key handle and HMAC parameters for k with those for a key k_a he owns. Thus the TPM generates a certificate for the attacker's key k_a using the legitimate user's signing key AIK_s. Sigrid Gürgens et. al discuss this attack in more detail [27].

2.9.3 MITM Replay Attack on TPM Authorization Sessions

Danilo Bruschi [10] shows how a MITM attack can be mounted on authorization sessions. The attacker intercepts an authorized OIAP command session and stores the authorized commands. Then a reset message is sent to the user, telling him that the session ended. By replaying the captured packages when the user reconnects the attacker might overwrite TPM protected resources. As the attacker does not get hold of any authorization secrets, he is not able to execute arbitrary commands.

2.9.4 Platform Reset Attack

Also known as cold boot attack, this side channel attack relies on the fact that data stored in volatile memory remains readable for a small amount of time after the system power has been removed. The attacker needs physical access to the platform. He then powers off the platform and immediately powers it on again. The attacker loads a small system using a CD or USB boot media to immediately dump the memory contents. The memory dump might reveal encryption keys or other secrets, depending on the system state at the time of shutdown.

2.9.5 Offline Dictionary Attack on Authorization Data

Another attack is presented by Liqun Chen and Mark Ryan [11]. They present how an offline dictionary attack can be driven by targeting the user chosen passwords which can be weak. Once broken, the passwords allow an attacker to unbind data or migrate keys.This attack requires the attacker to observe the TPM data flow, e.g. it is not possible to gain access to data on a stolen, switched-off laptop. If strong authentication passwords are used for TPM keys, this attack is rendered improbable.

2.9.6 Timing Attack

Evan Sparks [61] also discusses a possible Boneh-Brumley timing attack [9] on the TPM. Boneh and Brumley measured the time cryptographic operations take and were able to extract private keys from an OpenSSL based web server. Sparks used high resolution

timing samples of TPM operations. By analysing the required time for TPM_Seal operations he claims that it would be possible to 'guess' a private key bit by bit. He states that the attack requires about 2100 timing samples and thus needs an overall time of about 40 days to complete. No practical evidence of this attack is given.

Chapter 3

Identity Management

3.1 Introduction to Identity Management

With the increasing relevance of the internet, the use of network oriented services enables new business models. Relationships between business owners and customers are shifting from physical to software based relationships. By using electronical means, customers can access a wide variety of services.

In both business models, physical and digital, the customer has an identity. A real life identity consists of all attributes that belong to that individual person, e.g. name, address, date of birth, health status and so on. This identity is split into multiple, partial identities containing only a subset of the attributes. The different partial identities are distributed among different consumers of the identity data.

Every individual has multiple partial identities in real life. For example a partial identity presented towards a bank will contain the name, address, account number and balance. The partial identity of the same individual towards the health insurance might include information about the medical state as well as the name and address. This separation of data by assigning subsets of attributes to the identities shown to other parties allows to provide every partner with the information needed to provide the service. On the other hand it allows for the selection of the attributes to disclose to another party.

In general *Identity Management* (IDM) deals with the aspects of creation, usage, management and possible destruction of digital identities [69]. In the context of IDM, digital identities remain not limited to persons but are extended to resources, machines and services. As these identities play an important role in emerging business processes it is necessary to provide an infrastructure which supports the management of digital identities throughout their whole lifecycle. With IDM it is possible to establish trust domains inside which all participants can trust each other. By providing an appropriate credential, e.g. sufficient identity attributes, an individual can enter the trust domain. As an example consider an online-shop which requires users to register with a real name and address to enter the trust domain of customers.

3.2 Identity, physical vs. digital

In traditional scenarios trust is based on the fact that the participants know each other, e.g. because they belong to the same company. As the customer-business relation shifts from physical to electronic means it is necessary to develop and establish new ways of trust relationships between enterprises and their customers.

The usage of internet based services increases the number of identities, represented by, e.g. different accounts for mail, online-shops, auctions and so on. This enables the user to select which information to provide to create such partial identities and depending on each situation the appropriate identity can be used to reveal only the necessary information to the service provider. To keep track of all partial and distributed identity related information, the importance of IDM grows.

In general, IDM covers several aspects:

- **Trust** is linked to a set of identity credentials, allowing an individual to be part of a trust domain.
- **Anonymity** and **pseudonymity** play an important role in IDM. The level of identity needed for a relationship in a trusted domain has to be considered.
- **Authentication** is needed to prove that a claimed identity really does belongs to the agent. Examples are passwords, biometric devices or smart cards.
- **Authorization** describes the process of either granting or denying access to a certain service or resource.
- **Integrity** ensures that a message cannot be changed once it has been sent.
- **Non-repudiation** has to be ensured, this means the evidence for the existence of a certain message is given.

Several use cases of IDM can be imagined and are currently being widely promoted. In most web based services, the user needs to sign in for an account in order to access the service. This implies the need for an account management on behalf of the service provider. In an IDM scenario, the provider only has to take care of the authorization and has to establish a trust relationship to an identity provider. The client retrieves a ticket incorporating his identity from the identity provider. He then uses the obtained ticket and presents it to the service provider without the need of an additional registration. This lowers entry barriers for users who want to access the service.

3.2.1 Identity Claims

In general, the following steps are performed when an individual (*the user*) wants to access a service (provided by *the server*):

1. First, the user chooses the partial identity he wants to use to access the service. This association of the digital credentials and the individual is the **identification** phase. The user claims to be in possession of several attributes connected to the identity. This claim is targeted towards the *Identity Provider* (idP) as a third party authority.
2. In the second phase, the **authentication** phase, the individual provides a proof of possession of this identity by submitting data to the idP. The idP then issues an assertion to back up the user's claim, authorizing the user to make use of the claimed identity.
3. The authorized identity claim is then presented to the service provider (SP). Thus, the SP is not required to rely on the user's claim alone, but can rely on a third party assertion from the idP. The SP can rely on this assertion because he can expect that the idP has verified that the claimed identity is owned by the user.

As the SP trusts the idP's verification process for claimed identities, a chain of trust is formed. This shows that the user is trusted indirectly by the SP based on the user's relationship with the idP.

For the idP to make assertions on claims about user's identities to service providers, the SPs have to establish a trust relationship with the idP. The identity provider mediates trust between the individual and the service provider. In this centralized concept, service providers obtain user information from the identity provider. This reduces the amount of trust relationships that have to be established.

Trust relationships can be established by the exchange of attributes during the authentication process, e.g. a username and a password. To authorize an individual to enter the trust domain a token is issued to the individual after successful authentication. The tokens used for authentication and authorization are called credentials. In existing IDM solutions such as Kerberos, these credentials are embodied in software tickets.

3.2.2 Trust

As mentioned beforehand, the establishment of trust in digital identities plays a fundamental role in IDM. Trust, the "assured reliance on the character, ability, strength, or truth of someone or something", as defined by Merriam-Webster [30] is always based on the assessment of evidence and the sum of experiences of prior interactions.
Trust can be laid in individual persons and things such as computers, to behave in a particular way without the ability to enforce or monitor this behaviour beforehand.
This definition of trust introduces the willingness to take risks, especially when the other party does not behave as expected, or as stated by McKnight et. al in [45]:

> "...the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible"

In digital environments trust can be established by providing attributes that are related to an individual's identity. Depending on the use case a predefined set of attributes, e.g. name, age, gender, etc. is required to enter the target trust domain.

As an example, consider an online shop who asks users to enter address and bank account data. The business owner has to trust that the buyer will pay and therefore requires the user to provide data to make him accountable. On the other hand this example shows the concept of mutual trust. The user "trusts" the online shop provider to protect his data. Although he can not monitor further usage of data he supplied, he relies on assertions made by the online shop, often published as privacy policies.

This example shows that under given circumstances trust can be mutual, but by itself, trust is not symmetric. Given the fact that A trusts B does not provide that B has to trust A. Another property of trust is its transitivity. If A establishes trust in B and B trusts C, A may trust C. This is a key property for the establishment of trust chains.

All participating entities that trust each other form a domain of trust. By transmitting authentication data an agent can enter the domain of trust of a principal. As a second factor to authentication, an attestation of the system can be provided. This attestation should provide a statement on the system's integrity in terms of unmodified software or hardware. Within this combined authentication and attestation, the agent's trustworthiness can be confirmed, thus making a statement about the agent's identity and its state. A more in depth trust model for identity management scenarios is given in [15].

3.2.3 Privacy

Anonymity and pseudonymity play an important role in the context of digital identity management. While in anonymous service access no information about the user's identity is unfolded, pseudonymous access allows the use of services without revealing the real identity. Pseudonymous access still allows to perform accounting processes by the service providers as well as the establishment of local user profiles that are not necessarily related to the user's real identity.

To establish a trust relationship the individual has to disclose a certain amount of identity attributes. As every communication requires another level of individual information, this should be supported by IDM systems. In general, the user has to supply a lot of personal information to the identity provider. The identity provider can then reveal the data to the service providers. Worries about privacy concerns are to be considered in this context.

Privacy laws and regulations are established to protect user's data, such as the European Data Protection Directive for the EU. If an IDM system used by a company fails to protect privacy, the company will face legal liability, a damaged reputation and a loss of trust.
In this context the security of communication and transactions has to be considered in depth, as the picking of unique properties, such as an uncommon name, geographic information or health status information can lead to the linking of a digital identity to

identifiable personal data.
Privacy policies have to be established to control the usage and spread of identity related information and inform the user about it. They represent a mutual agreement on how the information is stored, distributed and deleted between the user and the identity provider.

3.3 Threats

With the shift from physical present identities to digital identities, attacks have changed, too. Criminal's targets have moved from money to information about the user's digital identities. Capturing this information can allow an attacker to impersonate someone else. This attack is called identity theft and is often the first step to commit further crimes under the name of someone else. Thus the security of IDM systems is of special concern.

3.3.1 Stealing of Authentication Data

For the security of IDM solutions it is essential to provide protection of the authentication data. If an attacker is able to get hold of the authentication data, identity theft can easily be committed and often even without the user's notice. Several attack forms exist which aim at stealing the authentication data.

Phishing attacks

One of the most prominent forms is the phishing attack. To authenticate the user, the identity provider uses a login form, prompting the user for his credentials. An attacker can rebuild the original login form of an identity provider and direct the user to this forged page. Believing that he logs into his identity provider, the user enters his authentication data, e.g. password and username. The page is prepared in such a way that the login information is directly forwarded to the attacker who can then exploit the authentication data. Phishing attacks are widely used by criminals to gain access to the authentication data of online services such as eBanking, online shopping, mail accounts that are then abused to send spam mails, etc..

With the use of centralized IDM solutions, an attacker can shift focus to one single point of attack, instead of targeting multiple services. This increases the need to protect IDM solutions with reliable security measurements that help to prevent phishing attacks.

Several options can be considered to counteract on the stealing of authentication data. As they focus on the protection of the authentication data they can be used in most IDM solutions but are not limited to this use. Some of the solutions include the use of two-factor authentication with additional security tokens. The authentication data is divided in a thing the user knows (e.g. the password or a PIN code) and something the user has (e.g. USB security token, smart card, fingerprint).

Another option to prevent this attack is the mutual authentication. Before the user provides his authentication data, he challenges the identity provider to prove its identity. Assuming that an attacker is not in possession of the secret needed to answer the challenge correctly, the user will not provide his authentication data to an attacker.

3.3.2 MITM Attack on the Authentication Process

While being effective against phishing attacks or similar ways that aim to get the authentication data, the methods mentioned above might not hold for a Man-In-The-Middle (MITM) attack. The attacker performs an active eavesdropping. He captures all network traffic between the user and the server the user authenticates to and relays it between them.

To mitigate MITM attacks a secure channel between the participating parties has to be established. This can be achieved by public key infrastructures or strong mutual authentication mechanisms. Ensuring end to end data integrity can play an important role in the prevention of MITM attacks that rely on the attacker's ability to change the intercepted data before relying it.

3.3.3 Compromised Machine

As IDM is not solely related to the management of real person's identities but can be leveraged to manage identities of devices, the danger of compromised hardware comes into play (e.g. the use case in section 4.1).

When IDM is used to manage devices such as set-top boxes, mobile equipment or sensors, an attacker who exploits design flaws and compromises the device can bypass restrictions, modify sensitive data or gain access to services without paying for them. A possible scenario is the change of a firmware in set-top boxes for digital television to access copyrighted material. An attacker could change the firmware of the device, allowing him to retrieve media at no cost without a valid subscription or recording it to external devices.

Ideally an IDM system should be able to detect such misbehaving devices and disallow the access to the network by denying the identity assertion necessary to access the service. Therefore the identity provider must be equipped with means to verify a system's internal state in a way such that a reliable statement on the trustworthiness can be made.

3.3.4 Building Identity Profiles

In addition, the spread of different, somehow loosely linked partial information leads to certain risks. As users tend to use same login information when registering with different services, gathering and linking pieces of information contained in different identities

can lead to a complete profile of an individual. This poses huge privacy concerns and potentially enables further attacks such as identity theft where an attacker obtains enough key pieces of personal information to impersonate someone else [42].

Chapter 4

Use Cases for a Trusted IDM

In this chapter I am going to present two use cases using IDM that can be enhanced using TC technology. Wherever suitable I will give references to the chapters describing (partial) implementations and to chapters deriving requirements from these use cases.

4.1 Automatic, Distributed Provisioning of Software to Networked Devices

This use case presents a way to provision a new software (e.g. a firmware-update) to a predefined set of standalone devices running the same configuration (e.g. set-top boxes) over a network. In such an environment there are always clusters of devices that are located very closely. Communication between the devices in a cluster can be considered cheap, whereas communication between a single device and a centralized server must be considered expensive.

The goal is to allow for a secure, trustworthy but still inexpensive way to distribute software through the networked devices. In every cluster, a designated device will provide the software locally. The concept is shown in general in figure 4.1. As shown, there can be sub-clusters, allowing for a finer grade of locality based distribution.

4.1.1 Environment

The operator O of the set-top boxes wants to deploy a new software to all devices. Therefore, the operator establishes at least one initial, trusted software distributor, referred to as S_0. A second trusted instance is needed to provide the IDM functionality, referred to as Identity Provider idP_0. The idP_0 and S_0 are usually managed and provided by the operator. Thus they are considered trusted in this environment. The idP_0 and S_0 can be operated by other parties. They must then establish a trust relationship with the operator via external legal contracts.

The operator generates a *Reference Integrity Measurement* (RIM) for the new software, this is the SHA1 hash of all new software components. The RIM is then transferred in a secure manner to the idP_0. The idP_0 has a database of the old measurement value (RIM') and the new RIM. Every device can report its current state, the so called *Target Integrity*

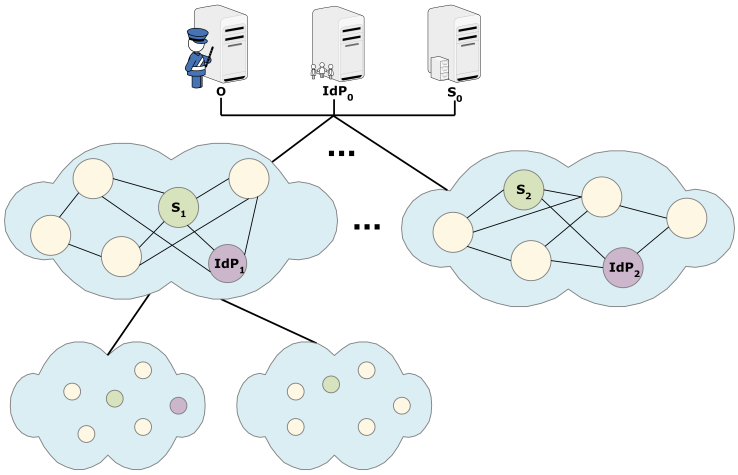

Figure 4.1: Sample Architecture for Use Case 1, 'Automatic, Distributed Provisioning of Software to Networked Devices'

Measurement (TIM). The idP_0 is thus able to verify a system's integrity by comparing the TIM to either the RIM' or the RIM, depending on whether the update has already been applied or not.

In order to reduce the load on the single source of the new software (S_0), we want the set-top boxes to become trusted devices and enable them to provision software to the other devices in the local cluster. They will then be referred to as software providers S_i.

Assuming that there are different hardware configurations for the networked devices, we will allow the more powerful devices in a cluster to take the idP_0 function locally. They will then be referred to as idP_i. This allows for further load balancing and distribution of computational effort and bandwidth usage.

Devices in sub-clusters can retrieve the update from their parent cluster, allowing for shorter ways of communication. This establishes a hierarchy of clusters. If no suitable device for software distribution can be found in a cluster, all devices in this cluster will have to access the service provider and identity provider at the parent cluster. A hierarchy of trusted devices can be built using this concept, allowing for a distributed provisioning of the new software.

4.1.2 Requirements

To establish this hierarchy, certain requirements must be met:

- the update will only be deployed to trustworthy devices (the TIM before the update must match the RIM')
- after the update, the device's TIM must equal the RIM

- only devices, where the TIM equals the RIM, can become local software distributors S_i
- only devices, where the TIM equals the RIM, can become local identity providers idP_i

In order to establish the trust relationship, we will use concepts of an IDM system. Each set-top box has a unique identity (e.g. serial number, geographical information, etc.). This identity is registered at the operator O. This unique identity is provided to the device by a ticket t_1 from O. The ticket t_1 must be cryptographically bound to the hardware of the device.

4.1.3 Application

In the following, the process for a single set-top box is presented. Note that only the idP_0 and the S_0 are considered as servers. They can be replaced by the local idP_i and S_i respectively.

Each set-top box can individually claim its identity at the idP_0, by requesting a ticket certifying the trustworthiness of the device. The set-top box provides t_1 from O stating, that the device is registered at the operator and has been manufactured and deployed in a trustworthy process.

After validation of the given ticket t_1, the idP_0 will request the TIM from the device. The device creates a signed report of the current TIM (see section 2.7.2, page 20). Given the signed TIM, the idP_0 will verify that TIM equals RIM' and then issue a ticket t_2 that is bound to the device. The validity period of ticket t_2 has to be short. Otherwise it would be possible to change the software on the device during the verification at the idP_0.

The ticket t_2 certifies that:

- the device is manufactured by a trustworthy manufacturer
- the device is in trusted state of RIM'

The device can then in turn use the ticket t_2 to obtain the software update from S_0.

After installing the update, the device can contact idP_0 again. This time idP_0 verifies that the update has been deployed successfully onto the device, that means TIM equals RIM. Then a ticket t_3 can be issued to the device. It is mandatory that the idP_0 sends a notification message about the successful update to O. This enables the operator O to track the distribution of the update among the network.

The ticket t_3 certifies that:

- the device is manufactured by a trustworthy manufacturer
- the device is running in trusted state of RIM, thus the update has been applied successfully
- the device can become a local software provider S_i

- the device can become a local identity provider idP_i

Ticket t_3 can be presented either at O to obtain a global legitimation or individually to the other devices in the cluster. The first option is preferable, because O can then send an authorized message to all devices in the cluster to use the new S_i or idP_i. Otherwise all devices would a-priori be required to be able to verify tickets.

4.1.4 Open Topics

Note that the last two points imply further requirements towards the software update. For the device to become a software provider S_i, the update must include the update package and the means to distribute it. Especially a function to verify incoming tickets is required. In order to become a local identity provider idP_0 additional requirements must be met. The software update must include (1) a means to verify measurements (TIMs) from other devices, (2) a function to issue tickets that are accepted at any service provider and (3) a database and management interface for further updates of RIM values.

4.1.5 Summary

This use case allows for localized, distributed provisioning of software updates to a predefined network of devices.
The devices can vary in the hardware but they are required to run in the same software configuration. Extending this use case and allowing different (or even custom) software configurations would require much larger databases for different RIMs. The transport of these databases to the local clusters would require more bandwidth and powerful device nodes to take the function of an idP_i.

4.2 Mobile Service Access

This use case considers the scenario of a user accessing a video portal using his mobile device (e.g. mobile phone, UMTS device). The service provider must be able to charge the user for the given service. The user's privacy towards the service provider and the operator is to be protected. Furthermore, the use case shall allow for service access without prior registration of the user at the service provider.

4.2.1 Environment

The following roles are present in this scenario:

- the mobile device M, used to access the service
- the user U using M
- the mobile network operator O
- the service provider S running the video portal
- an independent identity provider idP providing U with credentials to access S

The following points will be considered:

1. S must be able to charge U for the service
2. only O knows the real identity of U, e.g. by the contract of the mobile device
3. *idP* has to establish a trust relationship with O. This can be done via legal contracts between the two business partners providing the services
4. S wants to be assured that no illegal software is running on the device, e.g. a video download software
5. S has to establish a trust relationship with *idP*
6. the privacy of U must be preserved
7. S is not required to hold a local user database
8. thus U is not required to register at S

4.2.2 Application

The application will run as follows: As a prerequisite, we assume that O has integrated a certificate $cert_{dev}$ in the device, certifying that the device is operated by O. U decides which pseudonymous identity to use for the service access. For every pseudonym a ticket t_{id} must be retrieved from O using $cert_{dev}$. This ticket states that this pseudonymous identity is registered at O and thus can be resolved to a real identity at O. The ticket t_{id} must be cryptographically bound to the platform to ensure that the pseudonym can not be copied or misused by another device.

1. the user visits the page of S
2. S redirects the user to *idP*
3. *idP* challenges U to attest integrity and trustworthiness of M
4. a measurement of the running software is signed as well as the initial service request
5. the response is encrypted using a key provided through t_{id}
6. the encrypted response together with the t_{id} is sent to *idP*
7. if M's system state matches the requested state, *idP* issues a ticket t_s for the service request
8. U can access the service using the given ticket t_s.
9. When the service is provided, S can issue a charging request. This request is sent to *idP*, because S is not able to gain any further information than the pseudonym used for service access.
10. As *idP* is only able to resolve the correct operator O for this identity, the charging request is forwarded to O.
11. Finally O can resolve the pseudonymous identity to real customer data and thus charge U.

4.2.3 Discussion

In particular, the following points have to be considered :

The tickets t_{id} and t_s are cryptographically bound to the device, so pseudonyms cannot be copied to another device or stolen. Locally they can be protected by a password supplied by the user.

The service provider only has to maintain a contract with *idP*, agreeing about who is allowed to access the service and to agree on the charging process. Thus, service providers don't need to maintain a local user database. In addition, to enable further business cases, the user U is not required to (1) register explicitly at every service provider he wants to access and (2) U is not required to provide full identity information to S. Point (1) allows for quicker access by lowering the initial registration barrier, whereas (2) deals with the problem of untrustworthy service providers and information spread. Both can lead to privacy issues such as identity theft and profile linking.

To further enhance privacy, the *idP* can remove the actual service request when the charging request gets forwarded. Thus, O will gain no detailed information about the type of service U requested.

To counteract on rogue service providers, it has to be ensured that they can issue this charging request if and only if the service is provided. This has to be done externally by legal contracts between the *idP* and S and can be supported by the concepts of digital money and zero-knowledge proofs.

Chapter 5

Requirements for a Trusted IDM

In the following chapter I am going to outline the required aspects of a trusted ticket system. Most of the requirements can be derived from the previously described use cases. In addition, I will show the similarities between IDM systems and the concepts in Trusted Computing.

5.1 IDM and TC

Most IDM systems rely on tickets to establish trust between multiple parties. Tickets can be of various forms, such as certificates, tokens, cookies, etc. Previous work showed that tickets can be implemented using TC technology. The next section will outline this process in detail. The concept of Identity Federation can also be supported by TC technology. Usage of a client-centric solution is possible, as shown in the implementations. A key issue is the attestation of a client's trustworthiness together with user authentication and authorization.

5.2 Trusted Tickets

Derived from the use cases, the *Trusted Tickets* must meet several requirements. First they must be bound to the TPM platform. This enhances the overall security of ticket based systems. Secondly, every receiving party must be able to validate a Trusted Ticket. Furthermore the tickets must protect the integrity of the contained data. This can be achieved by digital signatures. To establish a ticket-based IDM it is inevitable to include identity information in the ticket.

As detailed in previous work from Nicolai Kuntze and Dr. Andreas U. Schmidt [35, 38], Trusted Tickets can be based on TPM generated AIKs. As AIKs can be used as identifiers they form the building block of Trusted Tickets. Using the process described in section 2.6 an AIK with the appropriate certificate cert(AIK,PCA) for a new identity can be retrieved. This AIK cannot be used to sign arbitrary data and thus a new key is generated inside the TPM. Once generated, the new key is stored in the TPM storage and it is ensured that its private part will never leave the TPM.

Using the TPM_CertifyKey command, the newly generated key is certified using the previously acquired AIK. The new key is therefore called Certified Signing Key (CSK) and its certificate is cert(CSK,AIK). Using the CSK it is possible to sign arbitrary data, such as a service request R, yielding the signature sig(R,CSK), see figure 5.1.

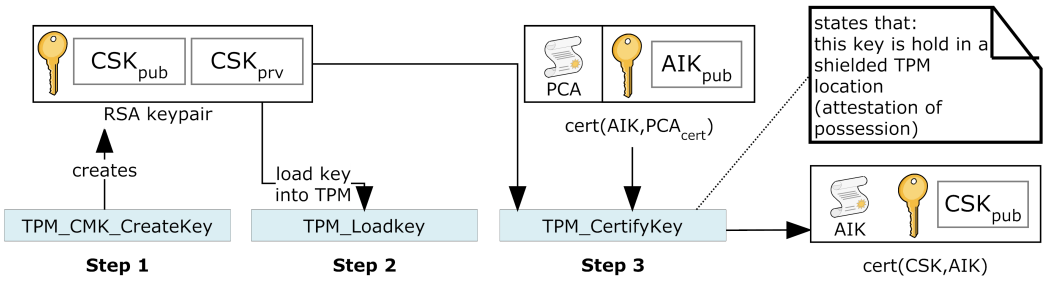

Figure 5.1: Creation of a Certified Signing Key

A Trusted Ticket now consists of four parts: (1) the cert(AIK,PCA), (2) the cert(CSK,AIK), (3) sig(R,CSK) and (4) the request R.

This ticket can only be generated with the TPM in possession of the AIK. The cert(AIK,PCA) ensures that this AIK is indeed generated and held in a TPM. The cert (CSK,AIK) extends the trust chain to the CSK and guarantees that the CSK is bound to the TPM. With the signature on sig(R,CSK) on the request, the proof of possession of the private part of the CSK is given. Furthermore, the CSK signature on R ensures that the data integrity is preserved.

To verify a given Trusted Ticket, a verifier can resolve the credential chain by: (1) validating the CSK signature on R, (2) verifying the AIK signature on CSK, (3) checking and validating the cert(AIK,PCA). The verifier has to establish a trust relationship with the PCA to provide a trust anchor for this credential chain. This credential chain could be extended with a root certificate from a root CA, certifying that the questioned PCA is indeed allowed to issue AIK certificates. In this case, the verifier could establish the trust relationship with the root CA which in turn enables him to verify tickets based on AIKs issued by all PCAs that are certified by this root CA.

5.3 Privacy and Security

One main topic in classic IDM environments is the conservation of privacy. As the tickets build the base for IDM solutions, the communication associated with retrieval, issuance and redeeming of the tickets as well as the tickets themselves shall not reveal more information than needed to any party involved. Furthermore, the protocol should be built to deal with an attacker collecting data on the communication channel. This can be supported by the use of cryptographic keys, provided by the TPM and bound to it.

To increase the security of existing IDM solutions, the ticket system shall provide an elevated protection against attacks mentioned in sections 3.3 and 6.1.1.

Combining the attestation of a platform with the authentication should be considered as a means to further improve the security. Thus tickets will not only be bound to a specific platform. They will only be issued to platforms that are considered trustworthy, e.g. are in a predefined system state, where only specified applications are running. This adds an additional benefit to TPM based trusted tickets.

Chapter 6

Trusted Kerberos

6.1 The Kerberos Protocol

The *Kerberos network authentication* [50, 51] provides an infrastructure for single-sign-on of managed identities within a predefined realm. The identities are represented by tickets that hold information about the identity. The information is protected using cryptography, allowing a client to prove its identity to a server over an unprotected network.

At the core of the Kerberos protocol is a trusted third party, the *Key Distribution Center* (KDC). The KDC, logically separable into the *Authentication Server* and the *Ticket Granting Server*, keeps a database of shared secrets between all participating parties. Once an entity registers with the KDC, the knowledge of the shared secret is used to prove the identity. To protect the communication between the networked entities, one time session keys are used.

The following roles can be identified in a typical Kerberos environment:

The User who has registered one or multiple identities within the Kerberos realm. The user can access services from service providers that are registered in the realm using one of the identities.

The Authentication Server (AS) is the central instance to provide IDM functionality. The AS issues *Ticket Granting Tickets* (TGT) to users upon successful authentication. Connected to a user database such as an LDAP connected Active Directory, information for every registered identity can be stored. The issued TGT has a long validity period, allowing the user to run multiple service ticket requests using one TGT.

The Ticket Granting Server (TGS) allows users to request *Service Tickets* (ST) for a specific service. The user has to present a valid TGT to the TGS. After verification of the TGT the TGS decides whether access to the requested service or resource can be granted. This decision can be based on policies or access control lists.

Service Provider accepts the ST and offers the desired service.

Each ticket contains two parts: one part is encrypted for the next target server and thus cannot be decrypted by the client. This part contains identity information and a session key that will be used by the next communication step. The session key is encrypted for the client in the second part of the ticket. If the client decrypts this part, he obtains the session key to request the next ticket.

The following four major steps can be identified, of which the protocol flow is shown in figure 6.1:

TGT retrieval The user connects to the AS and requests a TGT for his claimed identity. The AS looks up the key for this client (key_{client}) and for the TGS (key_{TGS}) in its database and generates the TGT. The TGT contains the identity and a session key ($key_{client,TGS}$) that will be used in the subsequent communication with the TGS. The user can decrypt this session key using his shared secret key key_{client}.

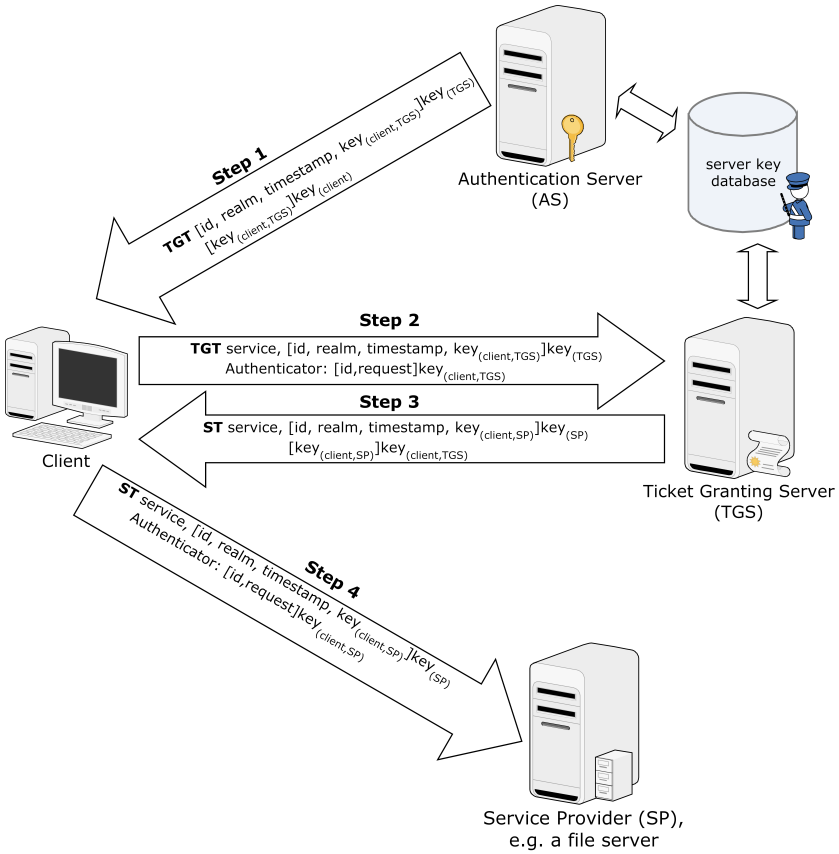

Figure 6.1: The standard Kerberos protocol: message diagram

ST retrieval To request access to a service, the user sends the TGT together with the service request message to the TGS. The TGS decrypts the TGT contents using key_{TGS} and

thus obtains the session key $key_{client,TGS}$ which is in turn used to encrypt a new session key, $key_{client,SP}$. The key $key_{client,SP}$ is encrypted using key_{SP} for the service provider inside the ST. The user can decrypt the new session key $key_{client,SP}$ from the TGS response using his key $key_{client,TGS}$.

Accessing the service The user encrypts its identity and request using $key_{client,SP}$ and sends it together with the obtained ST to the SP. The SP can decrypt $key_{client,SP}$ from the ST and thus verify that the request comes from the user being in possession of the identity. The session key can further be used to encrypt the response from SP to the user.

6.1.1 Possible Attacks on the Kerberos Protocol

The general threats mentioned in section 3.3 can be exploited in the Kerberos protocol by several forms of attacks. As presented, different attacks exist that aim to steal the authentication data and thus allow an attacker to impersonate the legitimate user. This allows the attacker to access all services with the user's privileges and may allow him to mount further attacks with the stolen identity.

In addition the following two attacks can be mounted against Kerberos:

Dictionary Attack Explicitly stated in RFC 4120 [51, chapter 1.6], Kerberos does not solve password guessing attacks:

> "Password guessing" attacks are not solved by Kerberos. If a user chooses a poor password, it is possible for an attacker to successfully mount an offline dictionary attack by repeatedly attempting to decrypt, with successive entries from a dictionary, messages obtained which are encrypted under a key derived from the user's password.

When the user chooses a poor password, an attacker can mount an offline dictionary attack trying to decrypt messages that were encrypted using the client's key. In order for an offline attack to be successful there has to be some plaintext to verify that the guessed password was correct. Kerberos messages contain preauthentication data, which is in most implementations an encrypted timestamp. Thus there is a structured plaintext that can be used to verify a password attempt. This makes password guessing attacks feasible on captured Kerberos messages.

Replay Attack Another attack that can be mounted against Kerberos is a replay attack [23]. As all tickets are electronic messages sent over an unsecured network, an attacker can capture the network traffic and copy tickets for later use. He replays them at a later time to impersonate someone else. The attack requires access to the network in order to succeed as the attacker listens and sends messages actively. The attacker cannot use captured TGTs (as they are encrypted using the user's secret key) but he can intercept STs and replay them later. Several options are available in the Kerberos protocol to mitigate this attack. Inclusion of the client's address and timestamping are the most prominent

methods to cope with this kind of attack.
This attack relies heavily on the fact that the tickets are not tightly bound to the platform.

6.2 Trusted Kerberos Protocol

6.2.1 Goal

The main goal of the Trusted Kerberos protocol is to show how Trusted Computing can be used to enhance a Kerberos authentication system. As mentioned before, privacy and security are the main targets of the concept. Security can be enhanced by providing means to bind the tickets closely to the hardware via the TPM. Privacy will be protected throughout the protocol by separation of duties between the AS and the TGS and by encrypting privacy sensitive communication.

The base of an identity in this context is an AIK generated inside the TPM. The concept of trusted tickets as described in section 5.2 is used to create TPM bound Kerberos tickets.

6.2.2 The Trusted Kerberos Protocol

Figure 6.2 provides a general overview of a typical service access using the Trusted Kerberos Protocol.

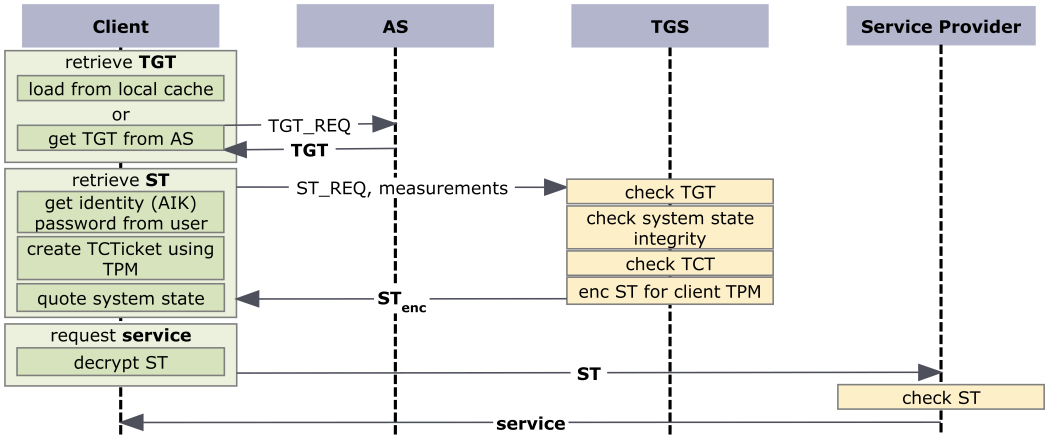

Figure 6.2: Trusted Kerberos: protocol message overview

The following steps are performed:

1. retrieve TGT
   - either load it from the local ticket cache, or
   - initiate a TGT request (see section 6.2.3 and figure 6.3) to obtain a TGT from the AS, then store the TGT in the local cache for further use.

2. retrieve ST from TGS (see section 6.2.4 and figure 6.4)
    - the user has to authenticate for the use of the claimed identity by providing the correct AIK and SRK password.
    - he creates a trusted ticket (TCT), containing the claimed identity and the service request both signed with the CSK.
    - the measured system state is signed by the TPM using the AIK.
    - the data is encrypted using the session key from the TGT and sent to the TGS together with the TGT.
    - the TGS checks the system integrity by validating the reported measurements and verifies the credential chain in the TCT including the signed request.
    - upon success, the TGS encrypts the ST for the client using the CSK public key and sends the ST to the client.
3. access the service
    - using the CSK in the TPM, the client can decrypt the session key from the ST and build the authenticator for the service request.
    - the authenticator together with the ST is sent to the service provider.
    - the service is provided to the client.

6.2.3 The TGT Request

The client initiates the connection to the AS requesting a TGT. In the current implementation the AS sends its public key as answer to the client. Normally this key is provided by a PKI, that is not present in this proof of concept. As the public key reveals no security critical information it can be implemented this way.

The client then issues the necessary commands to create an AIK for the claimed identity and sends the AIK'S public part together with the EK credentials to the AS. The message is encrypted using the AS public key. This ensures that only the AS is able to decrypt the message and thus link the claimed identity (represented by the AIK) to the real platform's identity (represented by the EK). Given that the AS public key is cryptographically strong, the identity information is sufficiently protected from eavesdroppers on the network.

The AS then generates a random, unpredictable nonce. As in the AIK certification process, described in section 2.6 on page 17, the AS uses this nonce to ensure that the AIK is indeed generated by a TPM and securely stored inside it. This message is encrypted using the PUBEK, and thus does not reveal any privacy sensitive information to an attacker on the network.

After decrypting the nonce using the TPM_ActivateIdentity command, the client is able to answer the challenge and sends the nonce back to the AS. The answer could be encrypted using the AS key. This would increase the computational effort on the client side, but it would not add to the overall security. The nonce is unique for each request and thus cannot be used again if intercepted by an attacker. If an attacker mounts a replay

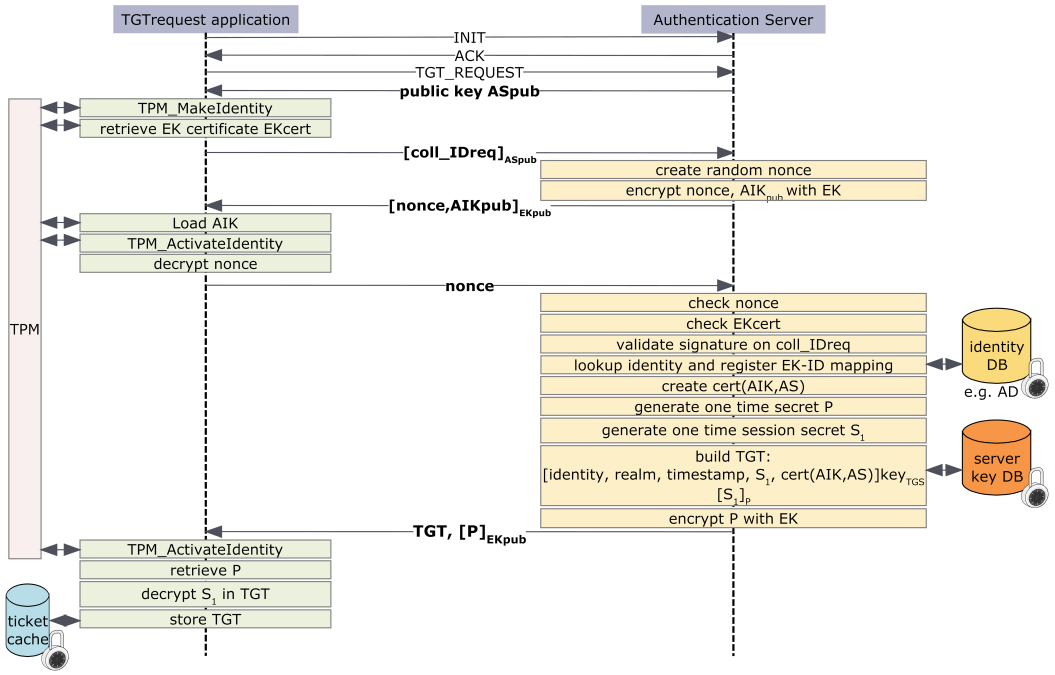

Figure 6.3: Trusted Kerberos: the TGT request

attack using the nonce, he will not gain any further information because the messages in the subsequent steps are encrypted for the legitimate client.

The AS first verifies that the returned nonce matches the challenge. Then, the EK credential is verified. This step could be done upon the first receipt of the certification request. As the verification of the EK certificate takes time, doing it at this stage prevents a possible DoS attack using a large number of fake certificates without a TPM.

The AS has to verify the signature on the identity request and after successful verification, the claimed identity is looked up in the identity database. This database could be implemented as a Microsoft Active Directory or similar directory services. The AS has to clarify that (1) the identity exists, (2) is not in possession by another platform and (3) is not revoked or invalidated. If the user is allowed to claim the identity in question, the AS stores the association of the EK to the claimed identity in the database. This allows the AS to resolve the claimed identity to the real platform if needed.

Two strong one time secrets are generated, a session key S_1 and the client secret P. The identity information together with S_1 and the AIK certificate cert(AIK,AS) is stored and encrypted in the TGT. The session key S_1 is encrypted using P separately. To ensure that only the requesting client is able to decrypt P and thus S_1, P is encrypted using the platform's PUBEK.

The client can then store the retrieved TGT for later usage. In the proof of concept im-

plementation, the session key in the TGT is decrypted in-place. This exposes all stored TGTs in the cache to a local attacker. Storing the still encrypted TGT and the encrypted P together is possible and would circumvent this danger.

During TGT request communication between the client and the AS, the client's real identity information is protected as it is encrypted using the AS public key. The response from the AS is encrypted using the public EK of the TPM and the private key of the TGS. Only the client and the TGS will be able to decrypt the given information.

6.2.4 The ST Request

If the client wants to access a service using the previously retrieved TGT, the TGS challenges him with a newly generated nonce. This nonce is then to be included in the quote of the PCRs. The quote is signed using the AIK for the claimed identity. For the implementation only PCR 10 was used, the concept can easily be extended to use multiple PCRs as well. Similar to the process described in sections 2.7.1 and 2.7.2, the signed quote and the SML are used to attest the system state to the challenging TGS. The nonce provides a statement about the freshness of the quote and a replay protection.

The client creates a trusted ticket (TCT) according to the process described in section 5.2. The TCT includes the signed service request and the claimed identity.
This ticket is generated each time a ST is requested. Thus, a new CSK is generated which is used to sign the service request and the claimed identity. The credential chain embodied in the TCT provides a proof of possession of the identity. Only the user who is in possession of the TPM which generated the certified AIK can generate this credential chain. Using the session key S_1 from the TGT, the TCT together with the signed quote and the SML are encrypted and sent to the TGS. It is crucial to encrypt the SML and the TCT to preserve the client's privacy. An eavesdropper will not gain any information about the internal system state when he captures this message. There's no need to encrypt the TCT for security reasons, as it can only be generated using the CSK which is bound to the TPM.

The TGS can decrypt the session key S_1 from the TGT using his secret key. In turn, S_1 is used to decrypt the quote, the SML and the TCT. Using the AIK certificate cert(AIK,AS) from the TGT, the TGS has to verify if the AIK matches the AIK in the TCT. This step is important as the cert(AIK,AS) builds the root of trust for the credential chain incorporated inside the TCT.

It assures the TGS that the identity is registered at the AS. As an optional step, the TGS can investigate if the claimed identity is still valid. As TGTs commonly have a validity of several hours, it might be necessary to revoke an identity. In this case, the TGS will not issue a ST to the client.

The TGS then has to verify the credential chain from the TCT. This credential chain has its root of trust in the cert(AIK,AS). As the TGS trusts the AS, the credential chain allows

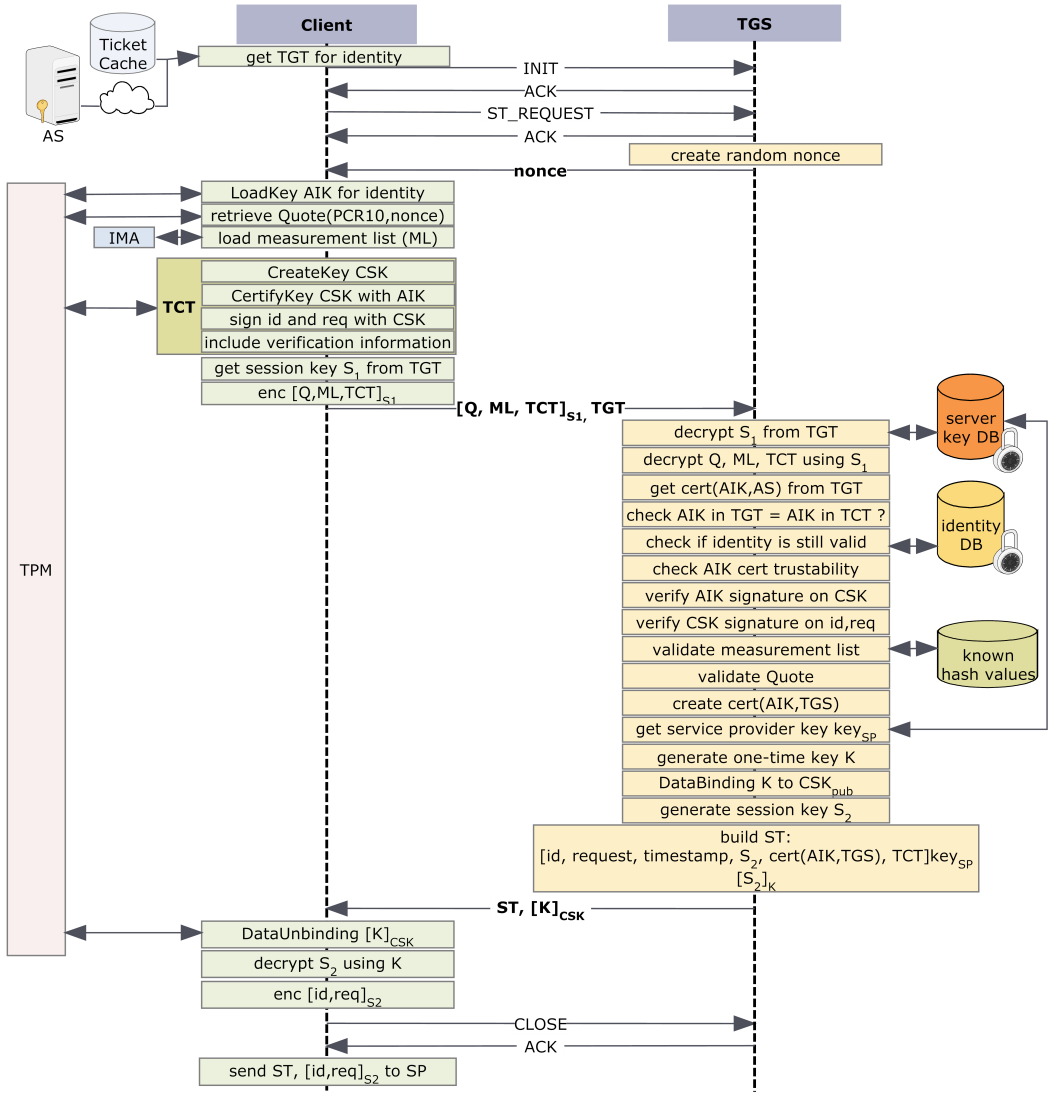

Figure 6.4: Trusted Kerberos: the ST request

to extend this trust up to the service request. Only if the signature on the identity and request has been generated by a CSK, certified with the appropriate certified AIK, can trust be laid into the identity of the requesting client.

Using the techniques described in section 2.7.2, the TGS validates the quote and the SML. This step enables the TGS to verify the system integrity of the requesting platform which allows to deny the issuance of STs to non trustworthy systems.

Then the TGS issues a certificate for this request and identity, cert(AIK,TGS). Again, two strong one time secrets are generated, a session key S_2 and K. K is used to encrypt the session key S_2 for the client and is in turn encrypted using a data binding to the CSK (see [40, server.ExternalDataBinding (ethemba)] and section 2.4). Therefore the CSK public part from the TCT is used by the TGS. The secret key for the service provider (key_{SP}) encrypts the cert(AIK,TGS), the TCT and S_2 for the service provider.
The client can then decrypt K using the TPM_DataUnbind operation and thus reveal S_2 (see section 2.4).

When the client uses the TGT to request a ST from the TGS, the communication is encrypted with the session key from the TGT. Only the TGS can decrypt the TGT, revealing the session key, and thus allowing the TGS to decrypt the data provided by the client. The response is cryptographically bound to the TPM by encrypting the session key with a one-time key bound to the CSK from the TPM. Only the user in possession of the TPM and the credentials for the key usage in the TPM can decrypt the ST and thus use it to access the service.

6.2.5 The Service Request

At this stage in the protocol, the client has retrieved a valid ST, with an included certificate from the TGS to attest the system's integrity. To prove possession of the CSK to the SP, the client has to decrypt the session key S_2 from the ST. This key S_2 from the ST is then used by the client to encrypt the identifier and the service request. The encrypted data is then sent as an authenticator together with the ST to the service provider.

The SP decrypts the session key S_2 from the ST using key_{SP} and can thus reveal the identifier and the request from the authenticator. If the identifiers in ST and authenticator match, the SP continues to verify the credentials included in the ST.

The SP is not required to check the system integrity again and relies on the cert(AIK,TGS) issued by the TGS. There has to be a trust relationship founded by external contracts between the TGS and the SP for this condition to hold.
Then the SP verifies the credential chain from the TCT. Finally, if the signed request in the TCT matches the decrypted request from the authenticator, the service is provided.

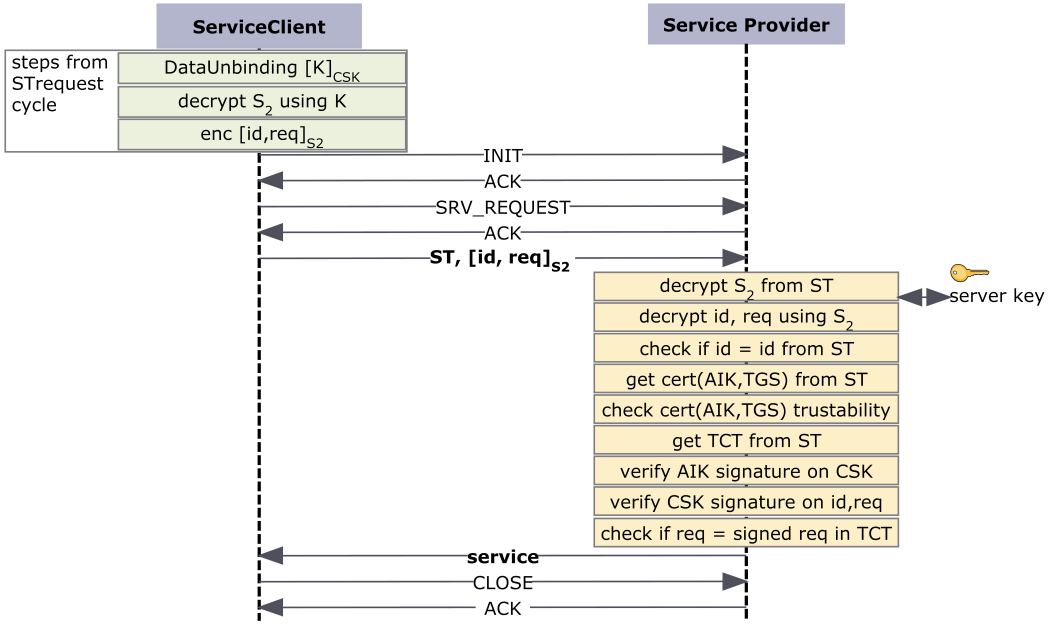

Figure 6.5: Trusted Kerberos: the service request

Chapter 7

Analysis

The following chapter shall provide a discussion of the Trusted Kerberos concept and implementation. Despite a general discussion, the two main topics security and privacy are discussed in depth.

7.1 System Integrity Measurement

During the ST retrieval, the TGS verifies the SML from the client. Therefore every submitted hash value is checked against a database of known and trusted hash values.

Every version of a program or library has a unique hash code. If multiple trustworthy versions of a program, an operating system or even completely different operating systems are used, the database must provide hash values for a large amount of different files.
Despite the effort to manage, store and provide the infrastructure for such a database system, several requirements must be met. A process for the certification of programs and libraries has to be established. It can be imagined that a central instance tests and certifies them and as a consequence adds the correct hash value to a database. This *Software Certifier* (SC) can then provide the hash values to subscribers. A participating TGS associates with the SC and updates his local database on a regular basis with the hash values.
As new security holes get discovered, the SC will remove previously trusted applications from the database and add the hash of the new version into the database. The SC then propagates the update to the subscribers. As the SC decides upon which software is considered trusted new problems arise. The SC can effectively restrict software usage and therefore take influence on market conditions by preferring or discriminating one or multiple software vendors.
The effort to maintain this process with changing applications and operating systems makes it impracticable to be implemented for a general, wide spread architecture (e.g. the client PC market). But in somehow restricted environments, where both factors, the used hardware and the software, can be controlled and are limited, this concept can be implemented.
The previously mentioned use in mobile phone environments or in embedded devices

such as in the described use cases seems to be practicable. If an operator deploys a set of devices that run without user interaction and operate on a machine to machine basis, he can use the implemented techniques to manage the device identities.

7.2 Enhancing security

In order to enhance the security it is ensured that all issued tickets can only be used by the legitimate user of the TPM. They are tightly bound to the user's hardware. A duplicate of the ticket cannot be created. In addition to that, there is no client secret shared between the Kerberos authentication server and the client.

For every TGT request a new and cryptographically strong one time password is generated by the server. We can assume that the server has enough hardware resources to generate appropriate passwords. As the passwords in the standard protocol are chosen by the user, this is one weak spot allowing eavesdroppers on the network to collect TGTs and try to decrypt them offline using brute-force and dictionary attacks. The chance of finding weak passwords if there is a large amount of clients in a Kerberos realm makes this attack feasible.

The impact of such an attack can become quite heavy. As the TGT represents the user's identity in the Kerberos realm, the attacker will gain access to all services the legitimate user has. By using one-time passwords that are cryptographically strong and can only be decrypted using the target TPM, the offline attack on captured tickets will become impracticable.

As a second point adding to the security on the server side there is integrity validation of the user's system included. When the user wants to acquire a Service Ticket (ST) to access a service, he is challenged by the TGS to attest system conformity. As a result, only clients in a certified system state will be able to access the service.

It has to be considered that this attestation does not provide a real-time attestation of the system state. The signed quote only represents the system state at a discrete point in time. All programs and processes started after the sending of quote and SML to the verifier are not included but still measured. Thus it is imaginable that an attacker launches a malicious program just after sending the quote. The TGS will attest the system's trustworthiness although it is not in a trusted state when accessing the service later on.
As stated in section 2.7.2 this protocol is heavily prone to TOCTOU attacks. Due to the network transport, the time between check (signature of the PCRs) and time of use (verification of data and use of issued ST) is prolonged. It is impracticable to freeze the client system hence stopping all user input during the whole process. So this concept might preferably be adopted by systems with no (or at least minimum) user interaction, such as embedded devices or mobile phones.
Another concept is the use of virtual appliances targeted for a single service. The client can measure and then start a virtual subsystem which can be measured completely. The integrity measurement of the whole subsystem is then reported to the TGS to obtain a ST. As only a virtual subsystem is affected by the attestation process it is more likely

to monitor and prevent possible modifications between measurement and use of the ST.

A further common threat to Kerberos systems are direct attacks on the database containing the user passwords. If an attacker is able to steal the passwords by compromising the system hosting the database, he can covertly use the passwords to impersonate the identities of the legitimate users within the realm. This threat is mitigated by the use of one time passwords. As the AS generates a new client password for each TGT request, a direct attack on the user database will not reveal any client passwords.
It is to be noted that an attacker might indeed alter the user database and add or remove users. This attack has to be prevented by further means and is out of scope of this thesis.

If an attacker gets unauthorized access to the server key database, he can decrypt all messages targeted towards either AS, TGS or SP. This attack has major impact on the whole infrastructure as it allows to impersonate every server in the protocol. Thus existing solutions to secure the database have to be applied thoroughly by administrators. Furthermore, all attacks targeting the hardware and operations of any of the servers are not discussed in this thesis but have to be considered in practical use.

In the current implementation, the session keys from TGTs and STs are decrypted in place, thus exposing them to attackers when stored in the local cache. In general it is advisable and possible to store both, the encrypted TGT (resp. ST) and the information on which TPM keys must be used to decrypt it, in the ticket cache. Thus, the session key will only be revealed when needed. During the ST retrieval process the secret K needed to decrypt the session key S_2 is encrypted for the targeted platform using the TPM_DataBind operation. This mechanism ensures that the ST is bound to the platform containing the TPM.

Another idea would be to use a Data_Sealing operation for this binding. The key K is then not only bound to the platform but to predefined PCR values, too. Actually, the PCR values could be the ones included in the quote. According to the specification of the Data_Sealing operation, the client will then only be able to decrypt the key K if the system's PCR values still match the values from the time the quote was retrieved. The previously discussed problem of software that gets started between the measurement and the use of the ST would thus be addressed. Only if the system's configuration, represented by the PCR values, does not change, the client can decrypt the session key S_2. Although this might add a little security, using the TPM_DataSeal operation also adds new problems. If a program or library is loaded with delay and thus is not included in the quote, the PCR values will change and the client wouldn't be able to decrypt the key K. This is very likely to happen, e.g. if the process used to send the measurement modifies the PCR values while sending the SML.

In the current implementation the SP returns the result of the service unencrypted to the client. This exposes no security risk, if the service is executed locally at the SP. An example would be a print server with a quota per user. The user redeems his ST for

the print of a document. The SP (print server) prints the pages and reduces the quota accordingly. In this case, only a notifier about successful execution is returned and thus encryption of the notifier message is not necessary.

Other applications can be imagined, where the SP has to communicate with the client (e.g. file download, video streaming, etc.). In this case, the communication can be encrypted using the either the CSK or a newly TPM generated key. Using the session key S_2 the two communication partners can decide on a new communication key. The use of the session key is discouraged as it is already known to the TGS. If the service is provided unencrypted to the client, an attacker could be able to capture the response from the SP and thus gain access to the service without prior authentication.

7.3 Enhancing privacy

The second goal, the protection of the client's privacy, is achieved by separation of duties between the TGS and the AS. In the present concept, a user has to register with one AS, revealing his complete identity to the AS. Then, the user can register multiple partial identities, referred to as claimed identities. The user claims to be in possession of the identity and the AS can certify this by providing a certificate cert(AIK,AS). The certificate cert(AIK,AS) is stored inside the TGT and encrypted with the TGS key. Thus only the targeted TGS will be able to decrypt the certificate. An eavesdropper will not be able to get any information about the claimed identity.

The AS will be the only instance being able to map the claimed identities to the real identity. By the means of legal contracts and AS' privacy policy, the user can ensure that the AS will not reveal its real identity. Service providers only get the needed information to provide the service to the user. The SP relies on the cert(AIK,TGS) issued by the TGS to attest the client's platform integrity. The trust relation between SP and TGS has to be established by external contracts.

To further enhance the privacy, no communication shall reveal information to other parties than the current communications partner. An eavesdropper on the network should not be able to get any valuable data from the network streams when listening to any of the ticket retrieval or redemption processes.
During initial TGT request, the communication between the client and the AS is encrypted using the AS public key, protecting the client's real identity information. The response from the AS is encrypted using the public EK of the TPM and the private key of the TGS. Only the client and the TGS will be able to decrypt the identity related information.
When the client uses a TGT to request a ST from the TGS, the communication is encrypted using the session key from the TGT. The TGT can only be decrypted by the TGS, revealing the session key, allowing only the TGS to examine the data provided by the client. The response is cryptographically bound to the TPM by encrypting the session key with a one-time key bound to the CSK from the TPM. Only the user in possession of

the TPM and the credentials for the CSK can decrypt the ST. No privacy sensitive information is disclosed.
The request to the service provider reveals no information concerning the user's identity to a third party as it is in turn encrypted with the session key from the ST. Only the targeted service provider will be able to decrypt the given partial identity.

Due to the separation of duties between AS, TGS and SP, the service provider will not gain information about the real identity of the user. As he is associated to a TGS that issued the ST, the service provider can contact the TGS operator in the case of misbehaving users. The TGS can then forward the message to the corresponding AS for this identity. As the AS has a mapping of the claimed, partial identities and the real identities, the user can be made liable.

In addition, the service provider is not required to keep a database of existing users. The user authentication and attestation is handled by the AS and the TGS. This concept enables the establishment of trust domains with multi-service single-sign-on experience for users wanting to access multiple services. Service providers can still build a local user database based on the pseudonymous identities, e.g. to provide distinguished services to select customers.

The design also allows to charge users for accessing a service. In accordance with the description in use case 2 (p. 40), the SP issues a charging request when the client redeems the ST. This request is sent to the TGS which removes the statement about which service was used and forwards the charging request to the AS. This step is important to maintain the user's privacy. Otherwise the AS would be able to link the real identity to the actual services that were used. As the AS is able to link the claimed identity to the real one, the AS can charge the user for the service. A rogue or compromised TGS would be able to send forged charging requests, charging the user to his own benefit. Using a zero-knowledge proof between the SP and the AS could assure that a ticket has indeed been redeemed at the SP without allowing to link the real identity and the service request.

Chapter 8

Conclusions

This chapter summarizes the results and findings of the thesis and presents an outlook providing aspects for further research.

8.1 Results

As a result of this thesis, a concept for the integration of Trusted Computing technologies into an existing ticket system, namely Kerberos, has been successfully developed. The establishment of an infrastructure for Trusted Tickets enables the establishment of trust in IDM systems which are based on the usage of tickets.

The presented approach provides increased security to existing solutions whilst enhancing their privacy. The concept is strengthened by the association of the tickets to the TPM and the assessment of the system's integrity.
Additionally, this thesis adds an academic view to the establishment of trust. Enhancing trust in network based services while maintaining privacy remains an important task. The lack of trust is a key inhibitor to the growth of network service architecture, including e-commerce scenarios. Bringing the notion of trust into an existing ticket system allows to build trusted identities, and thus trusted IDM solutions.

With the adaption of ideas from AIK certification with a PCA and the verification mechanism of remote attestation, the concept aligns with the TCG specifications. The issued certificates are valid X.509 certificates that can be integrated into existing certificate infrastructures. This allows for an easy adaption of the presented concept into established environments. The given reference implementation rebuilds all major parts of a Trusted Kerberos environment, including PCA and attestation functionality.
Although a complete integration on protocol level was left out, the concept can be further integrated into the Kerberos protocols and data structures. The generated tickets are not compliant to Kerberos ASN.1 formatted tickets. Investigation shows that an appropriate field in the ASN.1 tickets for the inclusion of the needed information exists and can be used for this purpose.

Further, with the establishment of the ethemba framework, a base for future research and development in the field of Trusted Computing applications has been built with

success. The implemented network protocols and data structures facilitate the usage and adaption of TCG concepts. It enables the rapid development of TPM based demonstrators and an experimentation environment by providing high-level functions for TPM access. Furthermore, with the developed network protocols and entities for AIK certification and remote attestation, the ethemba framework an eleveated infrastructure to build upon. The ethemba software implementation is complemented with a QEMU virtual machine which has access to a software TPM emulator. Inside the virtual machine, all TPM functions can be accessed and the necessary modifications to provide system integrity measurements are applied. As this framework is built upon a software TPM emulator, it can be used for research and development without the need for a TPM in hardware. While the ethemba framework gives a reference implementation of a PCA and the remote attestation process, both mechanisms have to be studied in more depth, providing a field for further development.

The benefits of Trusted Computing are shown by presenting two use cases in chapter 4. Generally, more and more applications involving Trusted Computing technology emerge. Nearly all modern operating systems provide support for the TPM, but there are still some topics that have to be covered by future research. It is a key feature of trusted computing to enable system integrity measurements and their validation by third parties. Future use cases and applications for which software or hardware integrity are key requirements can build upon trusted computing concepts.

8.2 Outlook

Dealing with the establishment of a base technology, which brings Trusted Computing into IDM, this thesis provides multiple entry points to further research.

As the reference implementation is not fully compliant to the Kerberos message format, one topic will be the integration of the data structures into the ASN.1 specifications. This can possibly be done by exploiting the authentication_data field in the Kerberos messages. According to the Kerberos specification this field can be used to transport additional authentication data. In this case, additional modifications have to be applied to all participating entities, to fully evaluate the new message field correctly.

The integration into other IDM systems remains for further research. With the widespread use of software based tickets in information technology, the applications of trusted tickets remain not limited to IDM. An integration into the concepts for Trusted Watermarks in peer-to-peer networks, which are discussed by Andreas Brett in [5, 6], seems to be promising. In a peer-to-peer network in which content is to be distributed, trusted tickets could be used in analogy to physical tickets which allow the access to specific content, e.g. video or music. This concept would require every partner in the network with the possibility to verify the trusted tickets. To provide scalability it could be imagined that, similar to the presented use case, the peer-to-peer partners are able to issue tickets themselves. Additional use cases that require elevated security and validation of system integrity can be developed.

In the present concept, the client authenticates to the AS, TGS and the SP. By introducing mutual authentication, an elevated level of privacy and security can be achieved. One idea involves the authentication of the SP to the client. This enables to client to assess the trustworthiness of the SP and allows him to sort out rogue SPs. Another idea would be to let the TGS mediate the trust between the SP and the client in such a way that the TGS verifies the integrity of the SP before issuing the ticket. Trust models and protocols have to be established to enable a trusted mutual authentication.

As discussed in chapter 7, the establishment of an infrastructure to provide certified measurement values of trusted programs remains to be clarified. This infrastructure must establish policies to define which software to consider trustworthy and the measurement values always have to be integrity protected. These policies should answer the question of how such measurements should take place and which entity should provide them. This enables new business models for the certification of software, as such an entity can charge software vendors for the certification of their products. If a software is not certified, it will not be considered trustworthy and as a consequence can not be used on systems that require attestation. This shows that the entity which issues the certified measurements has to be impartial towards the different software distributors. The protection and maintenance of a database for reference integrity measurements will be a main target for future research as it is a key enabler for trusted applications that involve system validation.

The ethemba framework provides a rather pragmatical approach by letting the administrator of the database import a measurement file. It can be imagined that this reference measurement is provided by a clean-room assessment of the device. While being feasible for embedded systems, concepts have to be developed for the assessment of complex systems that react to user interactions.
Although ethemba tries to introduce a complete trusted system, the implementation of trusted boot is still missing. Currently, there is no BIOS code for QEMU available, which integrates TPM drivers such that the TPM can be accessed at boot time and be used to measure the BIOS. The chain of trust is thus lacking an integral part, namely the CRTM in the system's BIOS. Options to include TPM drivers in the Bochs BIOS used by QEMU, or the integration into open BIOS concepts such as coreboot [12] have to be explored by additional research.

Furthermore a formal verification of the protocols used in the ethemba framework and the Trusted Kerberos concept is not given in the present thesis and can be done by the use of tools such as SHVT from Fraunhofer-Institute for Secure Information Technology [22]. This includes the development of strategies to secure the local ticket cache, where solutions involving the TPM might come up.

As the tickets are bound to the TPM, a backup strategy or at least a migration concept must be developed to securely transfer the tickets in case of a system or TPM failure. TCG provides concepts for the migration of TPM keys and data. The applicability and integration of these concepts is to be discussed. The concept of a backup

and migration service presented by Gökhan Bal in [25] might be exploited for this purpose.

The analysis of open topics shows that Trusted Computing and its applications remain an interesting field of activity for future research. With the rise of ubiquitous computing and the “internet of things”, new methods have to be implemented to establish trust between networked devices and entities. Trusted applications, paired with reliable integrity validation, can become fundamental parts of forthcoming developments.

List of Figures

List of Tables

Listings

Glossary

AES ..... the *Advanced Encryption Standard* is a symmetric block cipher. It has been selected by the US Government as a replacement to DES.

Asymmetric Cryptography ..... also known as *Public key cryptography*, called asymmetric due to the asymmetry in keys and en-/decoding operations.

Authentication ..... describes the process in which a claimed identity is verified.

Authorization ..... describes the process to determine if a client is allowed to use a service or access a specific resource.

Brute-Force Attack ..... similar to a dictionary attack, a Brute-Force attack aims to guess passwords using (random) generated passwords. Brute-Force attacks require even more tries than dictionary attacks. Using longer passwords and increasing the number of possible different characters in a password requires an attacker to spend more time on guessing.

CA ..... an entity who signs certificates and thus binds public keys to a statement about the key holder is called a *certification authority*. Normally a CA is considered to issue X.509 certificates.

Ciphertext ..... output of a cryptographic encryption function.

DDoS ..... a special form of DoS is the *Distributed Denial of Service* which is driven by coordinating a large amount of systems to issue a DoS against a single target.

DES ..... *data encryption standard*, a symmetric bloick cipher. It is outdated by *AES*, as its key length of 56 bits is to be considered unsafe.

Dictionary Attack ..... a *dictionary attack* aims at guessing passwords by using a dictionary or a wordlist. The attack relies on the fact that most passwords are chosen to be words

that appear in natural language. A large amount of passwords must be tried in subsequent tries.

Digital signature ..... in *digital signature* schemes, a person A is able to sign a message m such that only A is able to produce this special signature but anyone else is able to verify the signature. An example of a signature scheme is the *RSA* signature.

DoS ..... *Denial of Service* (DoS) is an attack which aims at rendering a host and some or even all of its services unavailable. A DoS attack can be driven by flooding a service with a large amount of data or malformed requests.

DRM ..... *DRM* refers to techniques allowing for access control to limit the use of digital media. The goal is to technically protect media from unauthorized use.

Hash ..... a hash function is a mathematical function to convert a large amount of input data to a small datum. The return values are referred to as *hash values*, *hash sums* or *hashes*. Cryptographic hash functions are assumed to have two main properties: being irreversible (it is infeasible to find an input that maps to the given hash value) and collision-resistant (it is infeasible to find two messages that map to the same hash value).

HMAC ..... a *keyed-Hash message authentication code* is a MAC calculated using a cryptographic hash function together with a secret key. It can be used to verify data integrity and data authenticity. If *SHA-1* is used in the calculation of HMAC, the resulting MAC algorithm will be called HMAC-SHA-1.

jTSS ..... an open-source *TSS* written in java, published in the course of the EU OpenTC initiative by IAIK Graz (`http://trustedjava.sourceforge.net/`)

Kerberos ..... The name given to the Project Athena's authentication service, the protocol used by that service, or the code used to implement the authentication service. The name is adopted from the three-headed dog that guards Hades [51].

LDAP ..... the *lightweight directory access protocol* is an application protocol to query and access directory services such as Microsoft Active Directory.

MAC ..... a *message authentication code* refers to a short piece of information used to authenticate a message. They are usually built based on a secret key using fast symmetric ciphers.

MITM ..... a *Man-In-The-Middle* (MITM) attack is an active eavesdropping attack on a network. The attacker relays messages between two parties, making them both believe to be communicating with each other directly. For the attack to work, all network traffic must be captured and the attacker must be able to inject messages into the network. If the attacker achieves to impersonate the communication endpoints for both parties, he is able to control the entire conversation.

PCR ..... a *platform configuration register*, part of the *TPM* that stores system measurements.

PKI ..... *Public key infrastructure*, an environment which is necessary to provide public key cryptography and enable trust in large environments. The main objective of a *PKI* is to learn and verify certificates for other parties.

Plaintext ..... the message that gets encrypted or the output of the decryption of a ciphertext.

PRNG ..... a *pseudorandom number generator* generates sequences of numbers that approximate properties of real random numbers. The generated numbers cannot be considered true random, but are computationally indistinguishable from true random numbers. Often a *PRNG* is used to amplify a small number of random bits to a larger sequence.

Public Key ..... *Public key cryptography* systems generate a keypair, which consists of a public and a private part. It is infeasible to compute the private key if only the public key is given. Thus, the public key can be published and then be used to encrypt data for the person holding the private key or verify signatures. As no shared secret is needed prior to communication, *public key cryptography* enables many useful scenarios.

RNG ..... a *random number generator* generates random bits that are unpredictable by others.

RSA ..... is a *public key cryptography* algorithm for signing and encryption. It is widely used in electronic commerce and communication and believed to be secure using sufficient long keys. It was first described by Ron Rivest, Adi Shamir and Leonard Adleman in 1977.

SHA-1 ..... also written *SHA1* is the widely accepted choice of cryptographic hash algorithm. It has been designed by the NSA and published by NIST. Its name stands for *secure hash algorithm*.

Symmetric Key Cryptography .... *Symmetric key cryptography* requires both parties to share a secret. It is more limited than *public key cryptography* but it is still used due to its higher performance.

TCG ..... The *Trusted Computing Group*, the successor of the *TCPA* is a consortium of industry leading businesses and research institutes to implement Trusted Computing. The main goal was the development and specification of the TPM.

TOCTOU ..... If throughout the duration between when a system measures and tests a certain condition and when it acts on the result of the test, the condition might change, the *time-of-check* / *time-of-use* vulnerability occurs. This can actually happen during the remote attestation, when the client's system configuration changes during the time the certifier verifies the given measurements.

TPM ..... the *Trusted Platform Module* is at the heart of the TCG specifications. It is a separate, passive chip that provides a secure storage and cryptographical functions such as hash calculations and key generation. It has several registers, the *PCRs* to store platform configuration information.

Trousers ..... an open-source *TSS* written in C++ (`http://trousers.sourceforge.net/`)

TSS ..... As defined by the TCG, a *TCG software stack* is specified to use the TPM. Several implementations exist, most famous are *trousers* (a TSS written in C++) and *jTSS* (written in Java).

X.509 ..... *X.509* is a standard for a *PKI* to handle public key certificates.

Bibliography

[1] W.A. Arbaugh, D.J. Farber, and J.M. Smith. A secure and reliable bootstrap architecture. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 65–71, 1997.

[2] M. Benantar. *Access Control Systems: Security, Identity Management and Trust Models*. Springer, 2006.

[3] A. Bottcher, B. Kauer, and H. Hartig. Trusted Computing Serving an Anonymity Service. *Lecture Notes in Computer Science*, 4968:143–154, 2008.

[4] S. Bratus, N. D'Cunha, E. Sparks, and S. Smith. TOCTOU, Traps, and Trusted Computing.

[5] Andreas Brett. Trusted Watermarks. Diplomarbeit, TU Darmstadt, to be published 2009.

[6] Andreas Brett, Nicolai Kuntze, and Dr. Andreas U. Schmidt. Trusted Watermarks. In *To appear in: Proceedings of 2009 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB 2009)*, 13 -15 May 2009 - Bilbao, Spain.

[7] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM New York, NY, USA, 2004.

[8] Broadcom. Integration of TPM in Ethernet Controllers. `http://www.broadcom.com/press/release.php?id=700509`, april 2005.

[9] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[10] D. Bruschi, L. Cavallaro, A. Lanzi, and M. Monga. Replay Attack in TCG Specification and Solution. In *Computer Security Applications Conference, 21st Annual*, pages 127–137, 2005.

[11] L. Chen and M. Ryan. Offline dictionary attack on TCG TPM weak authorisation data, and solution. 2008.

[12] Coreboot. open source BIOS. `http://www.coreboot.org/`.

[13] Nihal A. D'Cunha. *Exploring the Integration of Memory Management and Trusted Computing. Dartmouth Computer Science Technical Report TR2007-594*. PhD thesis, MSc thesis. DARTMOUTH COLLEGE Hanover, New Hampshire May 31st, 2007, 2007.

[14] QEMU devel mailing list. Patch to Add TPM support. `http://www.mail-archive.com/qemu-devel@nongnu.org/msg13408.html`.

[15] Ronald Marx (ed.) et al. Specification of general identity-centric security model that supports user control of privacy. *SWIFT project deliverable D302*, January 2009.

[16] P. England. Practical Techniques for Operating System Attestation. In *Trusted Computing Challenges and Applications: First International Conference on Trusted Computing and Trust in Information Technologies, Trust 2008 Villach, Austria, March 11-12, 2008 Proceedings*, page 1. Springer, 2008.

[17] Luis Sarmenta (MIT) et. al. TPM/J Java-based API for the Trusted Platform Module. `http://projects.csail.mit.edu/tc/tpmj/`.

[18] Mario Strasser et. al. Software-based TPM Emulator. `http://tpm-emulator.berlios.de/`.

[19] Sparks et. al. TPM Reset Attack. `http://www.cs.dartmouth.edu/~pkilab/sparks/`.

[20] Barbara Fichtinger. Trusted Infrastructures for Identities. Diplomarbeit, Fachhochschule Hagenberg, Austria, may 2007.

[21] 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.

[22] Fraunhofer Institue for Secure Telecooperation SIT. Simple Homomorphism Verification Tool - Manual, 2007.

[23] J. Garman. *Kerberos: The Definitive Guide*. O'Reilly Media, Inc., 2003.

[24] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The Digital Distributed System Security Architecture. In *Proc. 12th NIST-NCSC National Computer Security Conference*, pages 305–319, 1989.

[25] Dr. Andreas U. Schmidt Gökhan Bal, Nicolai Kuntze. Injecting Trust to Cryptographic Key Management. In *To appear in: Proceedings of the The 11th International Conference on Advanced Communication Technology (ICACT 2009)*, Feb. 15-18, 2009, Phoenix Park, Korea.

[26] Sigrid Gürgens and Carsten Rudolph. AIK certification. 2006.

[27] S. Gurgens, C. Rudolph, D. Scheuermann, M. Atts, and R. Plaga. Security Evaluation of Scenarios Based on the TCG's TPM Specification. *LECTURE NOTES IN COMPUTER SCIENCE*, 4734:438, 2007.

[28] Sam Hartman. Distributed identity for the web. 2006.

[29] Reiner Sailer et. al IBM Research. Integrity Measurement Architecture (IMA). `http://sourceforge.net/projects/linux-ima`.

[30] M.W. Inc. *Merriam-Webster's collegiate dictionary*. Merriam-Webster, 2003.

[31] ITAA-Whitepaper. Identity management: Building trust, mitigating risks, balancing rights. 2005.

[32] B. Kauer. Oslo: Improving the security of trusted computing. In *Proceedings of the 16th USENIX Security Symposium*, pages 229–237, 2007.

[33] J. Kay. *Cryptanalysis Techniques: An Example Using Kerberos*. School of Computer Science, Carnegie Mellon University, 1995.

[34] J.T. Kohl, B.C. Neuman, and T.Y. Ts'o. The Evolution of the Kerberos Authentication Service.

[35] Nicolai Kuntze, Dominique Mähler, and Andreas U. Schmidt. Employing Trusted Computing for the forward pricing of pseudonyms in reputation systems. In Kia Ng, Atta Badii, and Pierfrancesco Bellini, editors, *Axmedis 2006 : proceedings of the 2nd international Conference on automated production of cross media content for multi-channel distribution; Workshops Tutorials Applications and Industrial*, Atti del Convegno, Leeds (UK), pages 145–149. Firenze University Press, 2006.

[36] Nicolai Kuntze and Andreas U. Schmidt. Transitive trust in mobile scenarios. In Günter Müller, editor, *Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006)*, volume 3995 of *Lecture Notes in Computer Science*, pages 73–85. Springer-Verlag, 2006.

[37] Nicolai Kuntze and Andreas U. Schmidt. Trusted computing in mobile action. In H. S. Venter, J. H. P. Eloff, L. Labuschagne, and M. M. Eloff, editors, *Peer-reviewed Proceedings of the ISSA 2006 From Insight to Foresight Conference*. Information Security South Africa (ISSA), 2006.

[38] Nicolai Kuntze and Andreas U. Schmidt. Trusted ticket systems and applications. In H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, editors, *New Approaches for Security, Privacy and Trust in Complex Systems*, volume 232 of *IFIP International Federation for Information Processing*, pages 49–60, Boston, 2007. Springer.

[39] K. Kursawe, D. Schellekens, and B. Preneel. Analyzing trusted platform communication. In *ECRYPT Workshop, CRASH-CRyptographic Advances in Secure Hardware, September*, 2005.

[40] Andreas Leicher and Andreas Brett. *Ethemba - Ethemba Trusted Host Environment Mainly Based on Attestation.* `http://ethemba.info`, 2008.

[41] A. Leung, L. Chen, and C.J. Mitchell. On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). Technical report.

[42] Liberty Alliance. Whitepaper: Identity Theft Primer. `http://www.projectliberty.org/liberty/content/download/376/2687/file/id_Theft_Primer_Final.pdf`, December 2005.

[43] Liberty Alliance. Whitepaper: Personal Identity. `http://www.projectliberty.org/liberty/content/download/395/2744/file/Personal_Identity.pdf`, March 2006.

[44] H. Maruyama, T. Nakamura, S. Munetoh, Y. Funaki, and Y. Yamashita. Linux with TCPA Integrity Measurement. *IBM Research Report (January 28, 2003)*, 2003.

[45] D.H. McKnight and N.L. Chervany. The Meanings of Trust. *Trust in Cyber-Societies-LNAI*, 2246:27–54, 2001.

[46] Sun Microsystems. JDK 6 Security-related APIs & Developer Guides. `http://java.sun.com/javase/6/docs/technotes/guides/security/`.

[47] T. Müller. *Trusted Computing Systeme: Konzepte und Anforderungen.* Springer, 2008.

[48] S. Munetoh. Practical integrity measurement and remote verification for linux platform. 2006.

[49] Nanodatacenters. Results - Security Experimentation Environment. `http://nanodatacenters.eu/index.php?option=com_content\&view=section\\&layout=blog\&id=9\&Itemid=63`.

[50] B.C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[51] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021.

[52] Department of Defense. Trusted Computer System Evaluation Criteria; December 1985, DOD 5200.28-STD. Technical report, Supersedes CSC-STD-001-83, dtd 15 Aug 83, Library, 1985.

[53] N.L. Petroni Jr, T. Fraser, J. Molina, and W.A. Arbaugh. Copilot-a coprocessor-based kernel runtime integrity monitor. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13 table of contents*, pages 13–13. USENIX Association Berkeley, CA, USA, 2004.

[54] Trousers Project. Trusted GRUB. `http://trousers.sourceforge.net/grub.html`.

[55] QEMU. open source processor emulator. `http://bellard.org/qemu/`.

[56] Reiner Sailer, Trent Jaeger, Xiaolan Zhang, and Leendert van Doorn. Attestation-based policy enforcement for remote access. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 308–317, New York, NY, USA, 2004. ACM.

[57] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.

[58] Dr. Andreas U. Schmidt. Trusted Computing: Introduction and Applications, Lecture notes.

[59] Marcel Selhorst, Christian Stüble, Felix Teerkorn, Florian v. Samson, and Daniel Nowack. TSS study - introduction and analysis of the open source tcg software stack trousers and tools in its environment. Technical report, BSI and Sirrix AG, `http://www.bsi.bund.de/literat/studien/TSS/TSS-Study_en.pdf`, 2008.

[60] S.W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2004.

[61] E.R. Sparks. A Security Assessment of Trusted Platform Modules. 2007.

[62] F. Stumpf, O. Tafreschi, P. Roder, and C. Eckert. A Robust Integrity Reporting Protocol for Remote Attestation. In *Proceedings of the Workshop on Advances in Trusted Computing (WATC)*, 2006.

[63] Sun. VirtualBox. `http://www.virtualbox.org/`.

[64] Trusted Computing Group. TCG PC Client Specific TPM Interface Secification, v1.2final, 2005.

[65] Trusted Computing Group. TCG Architecture Overview, revision 1.4, August 2007, 2007. Available at www.trustedcomputinggroup.org.

[66] Trusted Computing Group. TCG Credential Profiles, Specification v1.1, rev1.014, May 2007, 2007.

[67] Trusted Computing Group. TCG Software Stack (TSS) Specification Version 1.2, Level 1, Errata A, 2007.

[68] Trusted Computing Group. TPM Main, Part 1, Design Principles, spec v1.2 rev103, 2007.

[69] P.J. Windley. *Digital Identity*. O'Reilly Media, Inc., 2005.

Appendix A

Implementation of a Demo Environment

A demo environment was designed to develop and test the ideas to form a trusted ticket system based on Kerberos. Its design principles are based on the requirements to provide a platform for current and future work in the field of trusted computing. Several components are required and have been evaluated and properly configured to form a stable platform for development and demonstration.

A.1 Introduction to the Demo Environment

The demo environment consists of multiple components. To not be dependant on a hardware TPM, the TPM software emulator was used. This allows for easy debugging of TPM output and allows for convenient management of the TPM (reset, save state, etc.).

In order to be able to assess a trusted boot process, the emulated TPM is connected to a virtualized system. The TPM emulator runs on the host system and its functions are provided to the virtualized (guest) system. Therefore a patch was customized to work with the current QEMU source code. The guest system consists of a standard debian linux installation. In order to access the TPM inside the guest, the kernel has to be compiled with TPM driver support. Using the IMA kernel patch, the guest system was able to measure loaded programs at load time (see section 2.7.1).
The virtualization enables to assess the whole system from the boot process and allows to run multiple instances each with an own TPM. The following sections provide an overview of used software and how to set up the demo environment from scratch.

A.2 Requirements

Following hardware and software is required:

- a host system running any flavour of linux and decent hardware to provide enough resources for virtualization
- development tools and compilers (gcc v.3) to compile the software
- qemu v0.9.1 source [55]
- patch to enable the TPM in QEMU [49]

- TPM emulator v0.5.1 [18]
- linux kernel version 2.6.24-3 (available from `http://www.kernel.org/`)
- IBM IMA patch for the linux kernel [29]
- jTSS distribution and its required libraries (available from `http://trustedjava.sourceforge.net/`, see also section 2.8)
- jTPM for the parameter parsing functions of ethemba [17]
- ethemba software framework to run, test and develop applications [40]

A.3 Setup of the Demo Environment

A.3.1 Setup of the TPM Emulator

The TPM emulator enables to access and review the internal operations in the TPM, which made it a very powerful tool for analysis, testing and debugging.

The emulator consists of three parts: an implementation of the TPM Device Driver Library (TDDL), a kernel module (tpmd_dev) and the TPM emulator daemon. As specified by the TCG, TDDL provides a convenient way to access the TPM from applications. By substituting this library, applications that use the TDDL are forced to use the TPM emulator instead of a hardware TPM.
For those applications and libraries that access the TPM directly, the kernel module tpmd_dev simulates a hardware TPM by forwarding all messages directly to the TPM emulator daemon, which is the main component of the emulator. The tpmd listens on a Unix socket and waits for incoming commands. At current, most of the commands specified by the TCG are supported by the emulator.
The installation of the TPM emulator is quite straightforward, compiling from source version 0.5.1. To prevent that QEMU gets disconnected from the emulator before the guest OS is up, as long as no TPM commands are issued during the boot process, the value of TPM_COMMAND_TIMEOUT in tpmd.c is changed to a higher value (3000, default: 30).

A.3.2 Setup of QEMU

To establish the connection between the TPM emulator and QEMU to gain a virtualized client environment a patch from the QEMU mailing lists [14] was modified to work with the current QEMU source version 0.9.1 [49, 55]. The patch allows QEMU to connect to the Unix socket created by tpmd via command line option,and registers a new I/O port inside QEMU and forwards all commands to the socket and thus to the TPM emulator.

A.3.3 Setup of Virtual Machine Client

The virtual machine was set up by installing a minimal debian distribution with the aim of a small footprint in storage and memory usage. To communicate with the TPM the

kernel (version 2.6.24-3) was recompiled, adding the IMA patch from IBM [29]. To configure TPM support in the kernel the options CONFIG_TCG_TPM=y and CONFIG_TCG_ATMEL=y have to be set. In addition the IMA patch [29] was applied with the kernel configuration parameters CONFIG_IMA_MEASURE = y, CONFIG_IMA_TEST_MODE = y and CONFIG_IMA_MEASURE_PCR_IDX = 10).
There are several ways to access the virtualized system, such as VNC and SSH. The system can be customized to meet further requirements (graphical user interface, etc.).

A.3.4 Setup of Virtual Machine Server

To present the concepts of a PCA and remote attestation, a second virtual system is needed to provide the server side functionality. Essentially this was achieved by copying the client system and running it without TPM support.

A.3.5 Setup of Virtual Machine Network

To enable network communication between the virtual machines, a system based on a bootable ISO image, acts as router in the virtual network. This allows the virtual machines to communicate with each other and the outside world. It is configured according to the QEMU manual for virtual networks. By adding and configuring a DHCP and a DNS server in the router system, client and server VMs can easily be integrated in the network.

A.4 Starting the demo environment

A.4.1 Starting the TPM emulator

The location and the socket the emulator listens on are specified via command line paramters.

```
tpmd -s ~ale/tpm\_emu/tpmd\_storage -u ~ale/tpm\_emu/tpmd\_socket\:0 \$\@
```

Listing A.1: Start Script for the TPM Emulator

By using the '-f' paramter, the emulator is run in foreground mode. Adding the '-d' parameter enters debug mode for verbose output.

The emulator can be reset by using the additional parameters 'deactivate' and 'clear'.

A.4.2 Starting the client VM

The login credentials for the QEMU-VM are:

Username/Password: root / tpm

```
#
# QEMU Run File for the ClientTPM VM
#

#!/bin/bash
DISKFILE=/vm/ale/client_20081130.dsk

MAC=52:54:00:12:34:10
VNC_DISPLAY=10
MEM=512
TPMSOCKET=/home/ale/tpm_emu/tpmd_socket\:0
NAME=client1
#TPMSOCKET=/var/run/tpm/tpmd_socket\:0

#should not be changed, opens the socket for the VLAN,
#so that the VM can connect to the virtual network
LOCALSOCKET=1234


#if we get a parameter we load a different ImageFile
if [ $# -eq 1 ]; then
   DISKFILE=$1
fi

qemu \
-tpm $TPMSOCKET \
-hda $DISKFILE \
-name $NAME \
-m $MEM \
-vnc :$VNC_DISPLAY -monitor stdio \
-k de \
-usbdevice tablet \
-net nic,vlan=2,macaddr=$MAC -net socket,vlan=2,connect=localhost:$LOCALSOCKET
```

Listing A.2: Start script for the client VM

A.4.3 Starting the server VM

```
#
# QEMU Run File for the Server VM
#

#!/bin/bash
DISKFILE=/vm/ale/server_20081130.dsk
```

```
MAC=52:54:00:12:34:11
VNC_DISPLAY=11
MEM=512
NAME=server

#should not be changed, opens the socket for the VLAN,
#so that the VM can connect to the virtual network
LOCALSOCKET=1234


#if we get a parameter we load a different ImageFile
if [ $# -eq 1 ]; then
   DISKFILE=$1
fi

qemu \
-hda $DISKFILE \
-m $MEM \
-name $NAME \
-vnc :$VNC_DISPLAY -monitor stdio \
-k de \
-net nic,vlan=2,macaddr=$MAC -net socket,vlan=2,connect=localhost:$LOCALSOCKET
```

Listing A.3: Start script for the server VM

A.4.4 Starting the router VM

```
#
# QEMU Run File for the Router VM
#

#!/bin/bash
PATH=$PATH:/usr/local/bin
CDFILE=/vm/router/plop.iso
DISKFILE=/vm/router/plop.img

VNC_DISPLAY=0
MEM=80

#should not be changed, opens the socket for the VLAN,
#so that the other VMs can connect
LOCALSOCKET=1234


/usr/local/bin/qemu \
-hda $DISKFILE \
-cdrom $CDFILE \
-boot d \
-m $MEM \
-vnc :$VNC_DISPLAY -monitor stdio \
-k de \
-net nic,vlan=1 -net user,vlan=1 -net nic,vlan=2,macaddr=52:54:00:12:34:57 \
```

```
-net socket,vlan=2,listen=localhost:$LOCALSOCKET &
```

Listing A.4: Start script for the router VM

A.5 Ethemba Live Virtual-Machine

Another option to run the Ethemba demonstration framework is the *Ethemba Live VM*, included on the DVD. The Ethemba Live Virtual-Machine can be run in Sun Virtualbox [63] as a virtual machine and includes the QEMU guest VM and the TPM emulator.

To run the Ethemba Live VM, set up a new virtual machine with the included ethemba disk-image as first hard drive. The login credentials are (*username* / *password*): *root* / *root*.

Inside the virtual machine, the TPM emulator can be started with the script from section A.4.1. The script can be found in the */root/ethemba/* folder.

To start the QEMU guest VM, use the provided *start_client.sh* start script in the */root/ethemba/* folder. For the QEMU guest VM use the username *root*, password *tpm*.

Appendix B

Implemented Software Solutions

B.1 Ethemba

The implemented framework ethemba mainly contains applications for TPM maintenance on the one hand and client- and server-applications that implement AIK-Certification and Remote-Attestation on the other hand [40].

It is meant to facilitate the development and usage of TPM driven applications. Ethemba is based on jTSS for TPM functions and is written in Java.

For further implementation details and a technical description, the ethemba documentation is provided.

B.2 Trusted Kerberos Implementation

Details and a more technical description of the Trusted Kerberos implementation are provided in the appended documentation of Trusted Kerberos.

Appendix C

DVD Contents

C.1 Thesis

The diploma thesis can be found in the DVD root folder.

C.2 Ethemba Live VM and Demo Implementation

The *Ethemba Live VM* disk image is located in the `/vm` folder; login: *root* , password: *root*.

The folder also contains the setup files for Sun VirtualBox (Linux and Windows) [63].

The Ethemba Live VM already contains the QEMU guest VM with the username *root* and password *tpm*.

The demonstration implementation files are already included inside the VM. They are located in the */root/Tethemba/jkrb_ethemba/* folder of both the host and the guest machine.

To run the programs, the *ethemba.sh* shell script can be used to set all required options, such as the classpath. See the included video for detailed instructions.

C.3 Trusted Kerberos

The `/Tkerberos` folder contains the demonstration video of the Trusted Kerberos implementation.

The source code can be found in the `/Tkerberos/src` subfolder.

The `/Tkerberos/doc` subfolder contains the technical description of Trusted Kerberos.

C.4 Ethemba

The `/ethemba` folder contains the source code of ethemba. The ethemba documentation can be found in the `/ethemba/doc` subfolder.

C.5 Bibliography

If electronically available, the used bibliography items can be found in the `/bib` folder.

C.6 Software

The used software is provided in the `/software` folder.