Trusted Infrastructures for Identities

Barbara Fichtinger, BSc.

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Sichere Informationssysteme

in Hagenberg

im Mai 2007

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 24. Mai 2007

Barbara Fichtinger, BSc.

Vorwort

Die Grundidee für diese Arbeit, die Verwendung von Trusted Computing im Zusammenhang mit Identitätsmanagement, stammt vom Fraunhofer Institut für Sichere Informationstechnologie in Darmstadt. Im Rahmen der Arbeit habe ich mich dabei mit dem grundsätzlichen Problem zur Herstellung von Vertrauensbeziehungen zwischen verschiedenen Identifier Domänen, gestützt auf Trusted Computing, beschäftigt. Ein konkretes Konzept für die Herstellung der Vertrauensbeziehungen wurde entwickelt und schwerpunktmäßig im Rahmen einer Referenzimplementierung umgesetzt.

Am Beginn war das Einarbeiten in die Themen *Trusted Computing* und *Identitätsmanagement* eine wesentliche Aufgabe. Aufbauend auf meine Grundkenntnisse musste ich mir detailliertes Wissen in diesen Bereichen aneignen, um anschließend die geeigneten Konzepte ausarbeiten zu können. Dabei war es vor allem im Zusammenhang mit Trusted Computing schwierig, außer den Standards der Trusted Computing Group, informative Lektüren zu finden. Mittlerweile existieren allerdings bereits einige Bücher zu diesem Thema. Im Bereich des Identitätsmanagements hingegen ist das Literaturangebot äußerst umfangreich, weshalb es hier nicht so leicht war, einen geeigneten Einstiegspunkt in die umfassende Thematik zu finden.

Ich habe mich dazu entschlossen, die Arbeit in englischer Sprache zu verfassen. Dadurch wird es im Vergleich zur deutschen Sprache möglich, eine weitaus größere Zielgruppe mit den Inhalten zu erreichen. Vor allem in der Informatik hat sich Englisch bereits weitgehend durchgesetzt. Ein weiterer Grund ist vor allem mein eigenes Interesse an der englischen Sprache, das vor allem durch mein Auslandssemester während meines Studiums stark gestiegen ist.

Rückblickend möchte ich mich vor allem bei Nicolai Kuntze und Andreas U. Schmidt vom Fraunhofer Institut für Sichere Informationstechnologie in Darmstadt für die fachliche Unterstützung sowie für die mir gewährten Einblicke in ihre Forschungstätigkeit bedanken. Weiters danke ich noch meinem Professor an der Fachhochschule in Hagenberg, Eckehard Hermann, für die kompetente Betreuung in fachlichen und organisatorischen Angelegenheiten. Thomas Winkler vom Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie der Technischen Universität Graz sowie Mario Strasser von der Eidgenössischen Technischen Hochschule

Zürich waren bei Fragen zu Trusted Java und zum TPM Emulator stets hilfreich. Für das Korrekturlesen der Arbeit möchte ich mich außerdem noch bei Jeremiah Diephuis von der Fachhochschule in Hagenberg sowie bei Petra Obritzberger bedanken.

Ein besonderer Dank gilt aber vor allem meinen Eltern, Christine und Heinrich, die mir mein Studium ermöglicht haben, sowie bei meiner restlichen Familie für die Unterstützung während des Studiums und der Zeit, an der ich an der Diplomarbeit gearbeitet habe.

Bei Fragen oder Anmerkungen zu meiner Arbeit wenden Sie sich am besten per E-Mail an mich:

barbara_fichtinger@gmx.at
Barbara Fichtinger, BSc.
Sichere Informationssysteme
FH Oberösterreich, Campus Hagenberg, Austria

Kurzfassung

Die Herstellung von Vertrauensbeziehungen zwischen verschiedenen Identifier Domänen ist ein wesentliches Forschungsgebiet im Zusammenhang mit Identitätsmanagement. Konventionelle Ansätze, wie etwa Cross Certification, übergreifende Certificate Authorities oder das Spiegeln von Benutzerdatenbanken rufen meist einen hohen technischen Mehraufwand hervor. Ein möglicher Ansatz ist die Verwendung von bereits bestehenden Architekturen, wie beispielsweise jener, die von der Trusted Computing Technologie zur Verfügung gestellt wird.

Ziel dieser Arbeit ist die Entwicklung eines Konzeptes, das einem Benutzer einer bestimmten Identifier Domäne den Zugriff auf einen Dienst eines Providers in einer fremden Identifier Domäne, unter der Verwendung eines Tickets, das von einem Identity Provider aus der primären Domäne ausgestellt wurde, ermöglicht. Um zu entscheiden, ob dem fremden Ticket vertraut werden kann, muss der Service Provider verifizieren, dass der Identity Provider autorisiert ist, für die jeweilige Domäne Tickets auszustellen. Zusätzlich muss noch der Konfigurations- und Systemstatus des Identity Providers zum Zeitpunkt der Ausstellung des Tickets überprüft werden.

Im Rahmen der Arbeit wird die Realisierung dieser Anforderungen durch die Verwendung der Trusted Computing Architektur analysiert. Für die Autorisierung des Identity Providers wird der Prozess der Ausstellung von Attestation Identity Zertifikaten durch die Privacy-CA adaptiert. Falls der Identity Provider zur Ausstellung vertrauenswürdiger Tickets autorisiert ist, wird ein entsprechendes Key Usage Attribut im Zertifikat gesetzt. Um die Tickets zu erstellen, werden SAML Assertions zum Transport der Statusinformationen vom Identity Provider zum Service Provider verwendet. Das flexible SAML Framework erlaubt die Integration der mit dem Attestation Identity Schlüssel signierten Statusinformationen in die Attribute der Assertion. Zum Erzeugen der Informationen, die den aktuellen Konfigurations- und Systemstatus des Identity Providers beschreiben, werden die Attestierungsmechanismen der Trusted Computing Technologie genutzt.

Das entwickelte Konzept sowie die Referenzimplementierung zeigen, dass die Verwendung der Trusted Computing Architektur zum Herstellen von Vertrauensbeziehungen zwischen verschiedener Identifier Domänen im Zusammenhang mit Identitätsmanagement realisierbar ist.

Abstract

In the context of identity management, a major research topic is the establishment of trust relationships between multiple identifier domains. Conventional approaches such as cross certification, spanning certificate authorities or mirroring of user databases result in substantial technical overhead. A possible approach is the usage of already existing architectures such as the infrastructure provided by trusted computing technology.

The goal of this thesis is the development of a concept that allows a user from a certain identifier domain to access a service at a service provider belonging to a foreign identifier domain by using a ticket issued by an identity provider located in the primary identifier domain. To decide whether the foreign ticket can be accepted, the service provider first of all has to verify that the identity provider is authorized to issue tickets for the particular identifier domain. Moreover, the service provider has to validate the identity provider's configuration and system status at the moment of the ticket issuing in order to decide whether the identity provider's decision can be trusted.

In the thesis, the realization of these requirements by using trusted computing technology is analyzed. For the authorization of the identity provider, the process of issuing attestation identifier credentials by the privacy-CA has to be adapted. If the identity provider is authorized to issue trusted tickets, the privacy-CA includes a certain key usage attribute in the certificate. In order to create the trusted tickets, SAML assertions are used to transport the status information from the identity provider to the foreign service provider. The flexible SAML framework allows the integration of the identity provider's status information signed with the attestation identifier key in the attributes of the assertion. For the creation of information describing the platform's current configuration and system status, the trusted computing technology offers the necessary attestation mechanisms.

The developed concept and the reference implementation show that it is possible to use the trusted computing architecture for the establishment of trust relationships across multiple identifier domains in the context of identity management.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Problem Description

Due to the recent shift of the business processes to the virtual world and the increasing importance of service-oriented architectures, the significance of identity management has been steadily increasing. Creation, usage, management and possible destruction of digital identities are tasks which require sophisticated identity management architectures. There are different approaches for the realization of an identity management architecture which have one thing in common: they always deal with users who want to access a certain service in their particular identifier domains. In this context, an identifier domain is defined as the area of the network where the user can access services by using a certain identifier and the respective credentials. As the management of an individual pair of identifier and credentials for each service provider leads to a high management overhead, identity management architectures offer ways to access multiple services in a domain with the same identifier and credentials.

If the user aims to access a service in another identifier domain, the conventional solution would be the creation of a new identifier and credential pair for the service. However, it would be more convenient to use the already established identifier and credential pair from the other domain. This means that the foreign identifier domain would have to trust the decisions made in the other domain. For the establishment of this trust relationship, several conventional approaches—such as cross certification, spanning certificate authorities or mirroring of user databases—already exist. But unfortunately, all of them lead to substantial technical overhead. Therefore, alternative solutions need to be developed. An idea for the establishment of trust relationships is the usage of already existing architectures, such as the infrastructure offered by the Trusted Computing Group based on its Trusted Platform Module.

1.2 Motivation and Goals

Basically, the thesis examines the question of the establishment of trust relationships between multiple identifier domains in the context of identity management which arose during the research work of the Fraunhofer Institute for Secure Information Technology. In particular the foundations are laid in several research papers published by Nicolai Kuntze and Andreas U. Schmidt. One of these papers [31] deals with the establishment of transitive trust in mobile scenarios. In the second paper [32], a usage scenario for trusted computing in the context of user authentication for converging mobile access technologies is presented. The third paper which has to be mentioned [33] develops a trusted ticketing system based on the trusted computing infrastructure. This research project is already situated in the area of identity management and in particular analyzes ticket-based systems.

The reasons for choosing the trusted computing architecture for the establishment of the trust relationships are quite widespread. First of all, the trusted computing architecture (platforms with TPMs, privacy-CA etc.) does not have to be especially deployed because of this use case; it can already be taken as granted. In the context of trust, the trusted computing technology offers special techniques (transitive trust) which allow external parties to trust applications running on a certain machine equipped with a Trusted Platform Module. Another reason is the fact that more and more systems are deployed with such a Trusted Platform Module by default. Since the public architecture of the module allows a usage in various applications, the usage in the context of identity management seems promising.

Apart from the introduction to identity management and trusted computing, the major goal of the thesis is the enumeration of possible solutions to enable the establishment of trust relationships across multiple identifier domains by using the infrastructure provided by trusted computing. Depending on the evaluation of these possibilities, one of the approaches is chosen for the development of a concrete realization concept. The main parts of this concept are subsequently implemented as a proof of concept.

1.3 Outline

The second chapter of this thesis gives an introduction to identity management technologies. In addition to the description of identity management architectures and digital identities, different identity-related aspects—such as trust, access-control policies, privacy, authentication, authorization etc.—are discussed. Furthermore, important standards and frameworks (e.g. SOAP, SAML) are presented.

Trusted computing is the main topic of the third chapter. It presents the development of trusted computing and gives an introduction to the funda-

mental features such as roots of trust and transitive trust. Moreover, the Trusted Platform Module is described in detail and the cryptographic components (keys, certificates, attestation mechanisms, protected message exchange) are analyzed. The end of the chapter focusses on the development of applications using the Trusted Platform Module.

In the beginning of the fourth chapter, the problem of establishing mutual trust relationships in identity management architectures is discussed. As a part of this discussion, the basic requirements for the establishment of trust relationships by using trusted computing technology are identified. Possible solutions for the realization are presented in order to select one of the scenarios. In addition, the selected scenario is analyzed with regard to the previously described identity-related aspects. Finally, possible usage scenarios and application fields (apart from the establishment of trust relationships) are presented.

Chapter five presents the concrete realization concept for the selected scenario. First, the communication architecture with its components is described and afterwards, those protocol messages which have to be adapted or newly created are presented. Moreover, the functionality required by the Trusted Platform Module (e.g. key generation, signature calculation) is presented and discussed.

Implementation-related aspects are summarized in chapter six. After the presentation of the implementation scope, the application architecture and the infrastructure used (base system, Trusted Platform Module, additional libraries etc.) are described. Furthermore, the two main components (key generator and identity provider) are discussed with regard to their tasks and design. In addition, special implementation issues for the two components are mentioned.

An analysis of the implementation is presented in chapter seven, which basically consists of an enumeration of all the relevant issues, which have to be mentioned in conjunction with the practical part of the thesis. These issues are critically discussed and in case of arising problems, possible solutions are suggested.

The conclusion in chapter eight summarizes the major parts of the thesis and identifies possible tasks for further research.

Chapter 2

Identity Management

This chapter gives an introduction to identity management technologies. In addition to the description of identity management architectures and digital identities, different identity-related aspects—such as trust, access-control policies, privacy, authentication, authorization etc.—are discussed. Furthermore, important standards and frameworks are presented.

2.1 Introduction

2.1.1 The Need for Identity Management

During the last few decades, business models have significantly changed, mostly due to the rise of the Internet. Enterprises no longer establish physical trust relationships with their customers. Instead, the customer-business relationship has to work via the World Wide Web. Customers do not purchase goods at physical shops, but they use electronic means. Additionally, relationships between enterprises in the form of business partners, suppliers etc. have changed as well. This transfer of business models into the network-based world makes it necessary to develop strategies for the establishment of trust relationships between the parties.

Another key player in this development is the increasing importance of service-oriented architecture, which can be described as an application architecture based on functions or services defined using a description language that can be accessed via predefined interfaces [3, pp. 4f.]. The business processes are all based on these functions and services. As all the interactions have to be independent of each other, it leads to the requirement that they have to be connected with digital identities in order to successfully implement the processes. In this context, digital identity does not only stand for customers but also for employees, resources and services. And as these digital identities play such an important role in the whole process, it is necessary to establish an identity management process in order to manage them appropriately.

2.1.2 Identity Management

Digital Identity Management (IDM) deals with the creation, usage, management and possible destruction of digital identities [67, pp. 8ff.]. A digital identity (see section 2.2) can be defined as a set of permanent or long-lasting attributes associated with a subject or an entity [9, p. 36]. The definition of subject or identity is quite widespread as it includes people, organisations, software programs, machines etc.

Basically there are three different approaches for Identity Management Architectures (IMA) which are described in [28]. Figure 2.1 gives an overview of the different systems.

- In the **isolated IMA** each service provider acts as identifier and credential provider for the user. Even though this approach is rather simple, it is complicated for the user as it is necessary to maintain different identifiers and credentials for each requested service.
- The **federated IMA** creates an identifier domain (circle of trust) consisting of different service providers that accept the identifiers and entitlements issued by other service providers within the domain. Different agreements, standards and technologies are necessary in order to enable identity federation.
- In a **centralized IMA** there is only one identifier and credential provider in each identifier domain (circle of trust) which has to be used by all the different service providers in the domain.

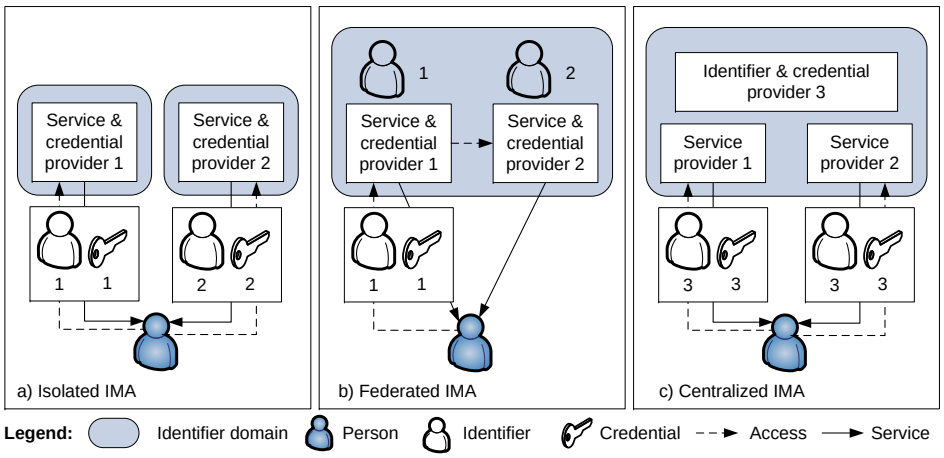

Figure 2.1: Identity management architectures. [28]

2.2 Digital Identity

2.2.1 Characteristics

Usually, a subject or entity possesses multiple digital identities. Depending on the particular situation the appropriate identity is used. Each identity stores the suitable attributes for the use case. According to Andre Durand [10] there are three different types of digital identities:

- **T1 Assumed (Personal)**: Identities belonging to T1 are timeless and unconditional; they are the true digital identities owned and controlled by the physical person. Permanent attributes such as name, eye color or birth date describe these identities.
- **T2 Assigned (Corporate)**: These identities are conditional and temporary; they are assigned by others especially in the context of a business relationship but with the consent of the physical person. Possible attributes are job title, library card or health insurance card.
- **T3 Abstracted (Marketing)**: Attributes that companies associate with customers are stored in T3 identities. Mostly, these group attributes are forced upon the customer without a consent in order to be used for marketing issues. A negative usage scenario of T3 identities is spam. As these identities do not offer a concrete benefit for the people, they are generally considered as rather bothersome. *Frequent buyer* or *one-time customer* are examples for T3 attributes.

Even though the purpose of the identity layers is different and they are controlled by diverse parties, they are in constant interaction. It is possible that identities of a certain layer contain links to identities from other layers. T2 and T3 identities in particular are converging because it is very easy for the companies to store additional marketing attributes with the T2 identities. This aspect could lead to severe privacy problems as the customers could be annoyed by too specific marketing campaigns or even spam.

As it is possible to use multiple T2 identities depending on the particular use case, another problem arises. On the one hand, each identity used in communication with a company only contains the attributes absolutely necessary for the particular purpose, which allows people to minimize the attributes the business knows about them. But on the other hand, customers tend to aggregate their identities because they do not want to administrate a vast number of identities at the same time. This means that a company might be able to link multiple T2 identities of a user.

2.2.2 Life Cycle

Regardless of the IDM system's complexity, the life cycle of the digital identities, as described in [67, pp. 30ff.], is the same. As a good design of a digital

identity life cycle is very important for a company, it should be planned at a very detailed level during the conception of the IMA. Figure 2.2 illustrates the process.

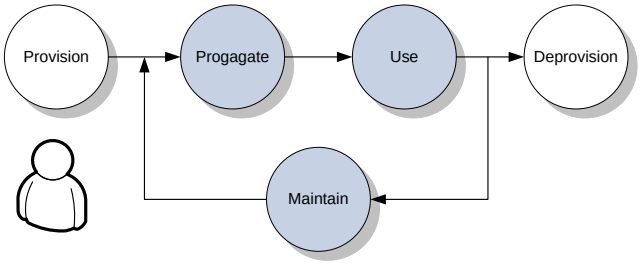

Figure 2.2: Digital identity life cycle. [67, p. 29]

During the provisioning phase, the digital identity is created and its attributes are specified. These tasks are either performed by an administrator or by the individual user as a self-service. Especially in cases in which only a few credentials have to be verified, the self-service approach saves a lot of work for a company. After the provisioning, the identity record may need to be propagated to other systems. One possible way is to copy the record directly to the other system; another method is to use a shared identity directory (e.g. LDAP repository). Each time an adoption of an identity record is performed, a new propagation is necessary. The next state in the life cycle is the usage of the identity. As it is inevitable that identity attributes change from time to time, the information has to be steadily maintained. Most often the IT help desk is responsible for these maintenance activities. The last step in the life cycle is the deprovisioning, which becomes relevant if a certain identity record is not needed any longer because for example an employee leaves the company. Above all, it is important that the deprovisioning tasks are correctly conducted, as a forgotten identity record which is still active can cause serious security problems in a company.

2.3 Identity-related Aspects

2.3.1 Trust

Trust is one of the fundamental aspects of identity management. In general, trust can be defined as

> "... 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." [37]

As analyzed in [67, pp. 15ff.], trust is always linked to a set of identity credentials and the associated attributes. In order to prove that an entity is authorized to use a claimed identity, credentials (passwords, digital certificates, biometrics etc.) should provide evidence. Another important issue is that trust is not something that can be taken for granted. A trust relationship can be established and abrogated due to particular reasons at any point of time. There are several trust properties that play an important role in this context:

- Trust is transitive only in very specific circumstances. If A trusts B and B trusts C, A may trust C.
- Trust cannot be shared. If A trusts B and A trusts C, it does not mean that B trusts C.
- Trust is not symmetric. If A trusts B it does not follow that B trusts A.
- Trustworthiness cannot be self-declared. Just because A says that the other parties should trust him does not mean that they have to.

Once a trust relationship is established, the participating entities form a trust domain or circle of trust. As described in section 2.1.2, the federated and centralized IMA try to create a certain trust domain in which the service providers trust each other or a central identity provider. In addition to the trust relationships inside one of these trust domains, it is also an issue to establish relationships between multiple domains that may be separated by technical boundaries (e.g. different access control technologies).

2.3.2 Access-Control Policies

During the access-control process, access-control policies form an integral part as they specify the entitlements and permissions of a certain subject for a particular resource [67, pp. 9f.]. An entitlement is a service or a resource an entity is allowed to access (e.g. credit limit, disk space, bandwidth allocation). The actions the subject is allowed to perform are called permissions (e.g. updating a record, completing a purchase). Examples of such a policy could be:

- All authenticated users from the IT department can write in a certain folder.
- Sales employees who are account managers in Europe can modify the customer database during working hours from the corporate LAN only.

These examples show that it is quite simple to formulate the requirements but the technical realization can be rather difficult.

During the implementation of access-control policies, the first step is to write them down. Thereby it is important to consider the principle of least-privilege which means that every subject should only have the minimal amount of necessary privileges. The next step is the policy implementation on the various systems [67, pp. 111f.]. A problem that arises in this context is that changes in the policies lead to adjustments on the implemented policies on different systems. Therefore, a single policy server can be used to manage all the access-policies in an enterprise and to send them on demand directly to the appropriate systems. One possible standard to realize this concept is the eXtensible Access Control Markup Language (XACML) [49], which is an XML-based language that can be used for storing and sharing access control policies.

2.3.3 Privacy

During the design of identity management architectures, privacy laws and regulations have to be considered as well [67, pp. 23f.]. As identity management systems often span over various countries, the differences of the regulations can lead to problems. The European Data Protection Directive, for example, is applicable for all the organizations operating in the EU, whereas the Health Information Portability and Accountability Act in the U.S. only deals with organizations that manage health care data.

Anonymity and pseudonymity play an important role in context with privacy issues. It is important to consider the level of identity that is needed for every particular relationship in an identity management architecture. Privacy policies can be used to inform the users what exactly is happening with their identity data. As defined in [13, p. 37], in an anonymous system it is possible that a user uses a resource without disclosing its identity. In pseudonymous systems users can also access a resource without allowing an inference on their real identities but still it is possible to conduct accounting procedures.

2.3.4 Authentication

Authentication deals with the establishment of an identity and is an essential prerequisite for access control [67, pp. 50ff.]. In order to prove that a claimed identity really belongs to a certain subject, it has to present credentials that can be authenticated. An important aspect is that authentication is also dependent on trust because it is not enough to present valid credentials; they need to be issued by a trustworthy entity. There are several authentication factors that can be used to establish credentials: (1) something you know, (2) have or (3) are; and of course combinations of these three factors are possible as well. Examples are ID and password, challenge-response systems, digital certificates, biometric devices or smart cards.

2.3.5 Authorization

One of the most fundamental aspects of IDM is authorization, the process of granting or denying a subject access to a particular resource [67, pp. 63ff.]. The identity established during the authentication phase is used to control the authorization actions. Access policies (see section 2.3.2) define the entitlements and permissions of a certain subject for a particular resource. Figure 2.3 shows the involved parties. The Policy Enforcement Point (PEP) is the entity from which a user requests access to a resource (e.g. web server). After the authentication of the credentials with the optional help of an authentication server, the Policy Decision Point (PDP) has to decide according to the access-policy whether the user is allowed to access the resource. The PDP's decision is stored in an Authorization Decision Assertion (ADA) which is sent to the PEP that finally denies or allows the user's request. Basically it is possible that the various entities are located on the same physical machine; nevertheless they should always be treated individually.

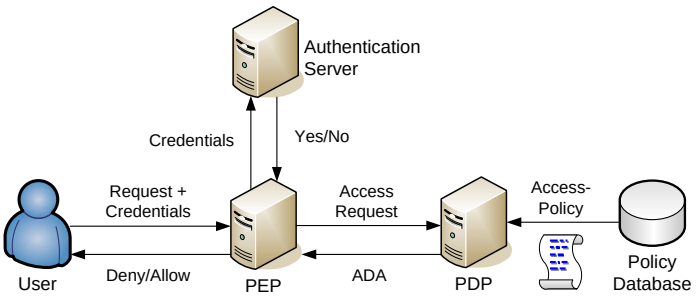

Figure 2.3: IDM authentication and authorization.

2.3.6 Integrity, Non-Repudiation and Confidentiality

In order to realize a secure identity management infrastructure, integrity, non-repudiation and confidentiality play important roles [67, pp. 63ff.]. Integrity ensures that a message or transaction cannot be altered once sent. A possible way to ensure this requirement is the usage of digital signatures.

The evidence for the existence of a message or transaction is called non-repudiation. It would be a problem if entities could pretend that they had not sent a message even though it arrived at the receiver. The proof that the message has indeed been sent is called non-repudiation of origin. The analogue case in which a receiver cannot deny the receipt of a message is non-repudiation of receipt.

Furthermore, confidentiality deals with the fact that only authorized

people or processes have access to the contents of a message or transaction. Especially cryptography is an important means in realizing this requirement.

2.4 Standards and Frameworks

In the IDM area, there are several consortia that are steadily developing IDM frameworks such as the Liberty Alliance, the WS-Roadmap or Shibboleth. All these complex technologies are based on standards such as XML, SOAP or SAML. The next sections give an overview about the most relevant standards and frameworks.

2.4.1 Simple Object Access Protocol (SOAP)

Overview

The Simple Object Access Protocol (SOAP) (currently available in version 1.2) is a lightweight protocol for the exchange of structured information in a decentralized, distributed environment (especially for web services) [16]. With the help of the underlying protocols, messages defined in XML format can be exchanged independently from the programming model that is used. Another major purpose of the SOAP is the possibility for other protocols to use it as a transport mechanism [48, pp. 111ff.].

SOAP Messages

A SOAP message (see figure 2.4) consists of three major blocks [45]:

- The **envelope** is required and marks the start and end of the SOAP message.
- An optional **header** consisting of one or more header blocks can be used as an extension mechanism to transport information in SOAP messages besides the payload stored in the body. This control information specifies for example details concerning the processing of the message for intermediary nodes or the receiver.
- In the mandatory **body** consisting of one or more blocks, the actual message is specified.

In order to transport message content with the help of SOAP, application-defined data structures have to be mapped to XML data and further on into SOAP messages. In addition to the exchange of regular XML-based content it is also possible to encapsulate remote procedure call functionality in SOAP messages [17].

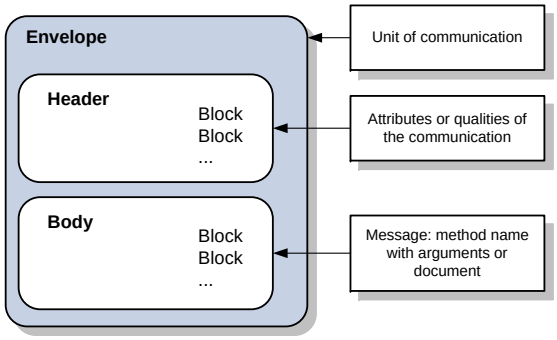

Figure 2.4: SOAP message format. [48, p. 113]

SOAP Message Exchange

The SOAP defines a messaging framework for exchanging information between a SOAP sender and a SOAP receiver optionally passing multiple SOAP intermediaries. A special processing model exists which specifies the actions of intermediary and receiving nodes to perform on the SOAP messages according to the header information in the messages.

As the underlying protocols are used for the transport of SOAP messages, a formal set of rules for the integration of the SOAP messages in the other protocol messages has to be specified. This specification is called protocol binding framework. Possible binding mechanisms for SOAP are HTTP [17] or Email [46]. The following issues have to be addressed by the binding mechanism [16]:

- specification of certain features provided by the binding (e.g. correlation of request and response messages, encrypted channel),
- usage of the services of the underlying protocol to transmit the SOAP messages,
- implementation of the features by using the underlying protocol,
- error handling and
- implementation guidelines.

In addition to the protocol bindings, SOAP message exchange patterns (templates for the exchange of messages between SOAP nodes) are required. In [17], the following three patterns are enumerated:

- **Request-Response Message Exchange Pattern**: A SOAP request message is first transferred from the requesting SOAP node to the

responding SOAP node. After the successful processing of the message, the responding node creates a SOAP response message and sends it back to the requesting SOAP node.

- **Response Message Exchange Pattern**: In this case, a non-SOAP message is sent as a request and a SOAP message is created as a response.
- **Web Method Feature**: Bindings supporting web methods (e.g. GET, PUT, POST, DELETE) give the applications the control over the web methods in order to use them for the transport of the SOAP messages.

2.4.2 Security Assertion Markup Language (SAML)

Overview

The Security Assertion Markup Language (SAML) [50] originates in the two standards Security Services Markup Language (developed by Netegrity, Verisign and further partner companies) and Authentication and Authorization in XML (developed by Securant). Both standards had been individually handed in to the W3C and the OASIS almost at the same time. In order to guarantee the compatibility in the web service sector, the result was a common specification for the SAML. Currently, the version 2.0 of the standard is available but as most of the products currently available on the market do not support the latest version of SAML, version 1.1 is the basis for this thesis.

SAML is an XML-based security standard for the transport of authentication and authorization credentials across identity and service providers in the same or across different identifier domains on the Internet [14, pp. 303ff.]. If an entity in an identifier domain requests resources in another domain, a trust relationship (see section 2.3.1) has to be established between the domains in order to accept the ticket in the foreign domain. Quite often, the identity and service providers in the identifier domains use different access-control systems. SAML addresses this compatibility problem by enabling a user to authenticate at the identity provider and accessing a service with the received ticket without having to authenticate at the service provider again.

Components

The main components of SAML and their relations are analyzed in [19, p. 6]:

- The **assertions** are statements about a subject issued by SAML authorities. They are classified in three different kinds:
  - After a subject's identity has already been proven to the authentication authority, the *authentication assertion* contains authentication information.

  - Specific information about the subject (credit limit, access level etc.) is stored in the *attribute assertion*.
  - Details about the activities a subject is authorized to perform on a specific resource are communicated in the *authorization decision assertion*.

- Further, SAML defines various generalized request and response **protocols** mainly for obtaining various assertion statements.
- **Binding** describes the communication mechanism of SAML messages over standard transport and messaging frameworks or protocols. SAML v1.1 defines the binding of SAML messages to SOAP messages [36]. Figure 2.5 shows a typical example of a SAML message encapsulated in a SOAP envelope.
- **Profiles** describe the communication and protocol messages for particular use cases.

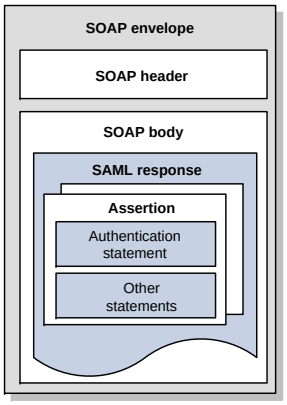

Figure 2.5: SAML message encapsulated in a SOAP envelope. [19, p. 7]

Profiles

The SAML specification [36, pp. 12ff.] defines two web browser-based profiles to support single sign-on (see figure 2.6). In both scenarios the subject represents the user and its web browser. Besides, the OASIS Web Services Security Technical Commitee (in the WSS specification) and the Liberty Alliance Project are working on usage scenarios for the SAML (see sections 2.4.3 and 2.4.4).

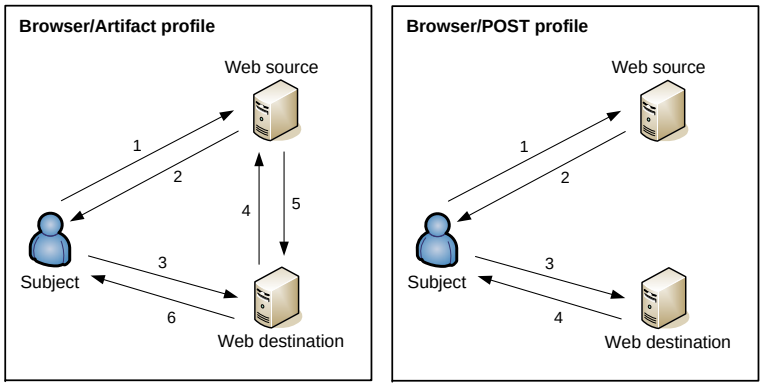

Figure 2.6: Web browser-based SAML profiles to support single sign-on.

- **Browser/Artifact Profile**: In the first step (1), the user requests the web source. After the successful authentication process, a session is created for the user. Next, the user has to indicate the desired access to another destination. The web source creates an assertion for the user and issues a SAML artifact containing the source ID of the SAML responder and a reference to the assertion. A redirection response is sent to the browser which includes the SAML artifact (2). If the browser processes the redirect it sends the artifact together with the service request to the web destination (3), which analyzes the artifact and sends a SAML request to the web source containing the artifact (4). Upon receipt of the request, the SAML responder sends the original assertion back to the web destination (5) which uses it to decide whether the user can be granted access to the desired service (6).
- **Browser/POST Profile**: After the request of the web source and the authentication (1), a session for the user is established. Again, the user has to indicate the desired access to another destination. The web source creates an assertion for the user and sends the full SAML response containing the assertion (e.g. as an HTML form) back to the user (2). If the user wants to access the web destination, the SAML response is sent to the destination (3), which validates the assertion and either denies or grants the access to the desired service (4).

Assertions

As SAML is an XML-based security standard, all the message types are defined in XML syntax. The most important messages in SAML are assertions. They consist of a generic envelope assertion element and a series of

inner elements that describe the authentication, attribute or authorization decisions. In addition, it is possible to define user-specific assertion types or extensions to already existing ones which allows applications to customize the assertions for their particular purpose. The XML syntax for assertions is specified in [35, pp. 11ff.].

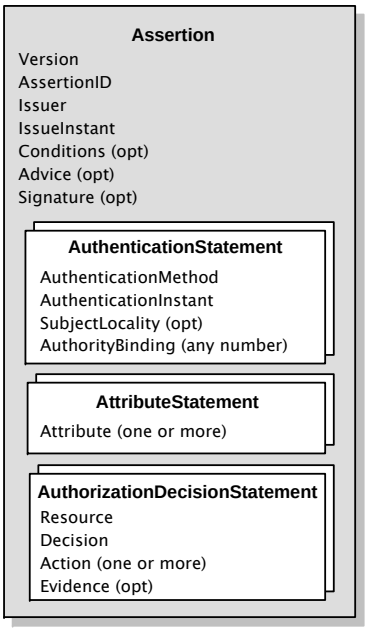

Figure 2.7: Structure of SAML assertions.

The content of an assertion element is illustrated in figure 2.7. The basic attributes of the assertion are version, identifier, issuer (SAML authority making the claims in the assertion) and issue instant (time when the issue took place in Universal Coordinated Time). Optionally, elements such as conditions which have to be evaluated if the assertion is used (e.g. time slot when the assertion is valid), advice which might be evaluated during the usage and a digital signature can be added to the assertion. The main content in the assertion are the statements. In addition to authentication, authorization decision and attribute statement, general statement types (Statement and SubjectStatement), which allow other assertion-based applications to reuse the SAML assertion framework, are available. In order to guarantee appropriate security, it is possible to digitally sign assertions via XML signatures.

In the authentication statement, authentication method (the type of authentication that took place) and authentication instant (the time when the authentication took place) are specified. An optional element is the sub-

ject locality which describes the DNS domain name and IP address of the authentication authority. Additionally, any arbitrary number of authority bindings which indicate whether additional information about the subject is available can be included in the statement.

For the transport of attributes, the attribute assertion has to be used. It stores one or more key and value pairs.

Information about the authorization process is stored in the authorization decision assertion. Resource (URI identifying the resource the subject wants to access), decision (permit, deny or indeterminate) and one or more actions the subject is allowed to perform on the specific resource are obligatory elements. One or more assertions that the SAML authority relied on in issuing the authorization decision can optionally be included in the evidence element.

If a party receives an authentication assertion of another identity provider, it might require additional information about the authentication procedures. There is a special attribute in the authentication assertions (AuthenticationMethod) for carrying this information. As it is likely that different authentication authorities will use different technologies, the SAML supports the specification of different authentication schemes such as password, Kerberos, hardware token, SSL/TLS certificate based client authentication, X.509 public key, PGP public key or XML digital signatures, to name only a few of them [30].

2.4.3 Liberty Alliance

The Liberty Alliance [34] was formed in 2001 and has grown to almost 150 members. Its first release was the *Liberty Federation*, an industry standard addressing authentication, privacy and security challenges surrounding online IDM. Another important release is the open framework *Liberty Web Services* for deploying and managing identity-based web services.

Technically, the Liberty protocols are extensions of the SAML protocol (see chapter 2.4.2). A Liberty architecture has three actors [14, p. 351]:

- Principal: A consumer or an entity that possesses an identity provided by an identity provider,
- Identity provider: The SAML asserting party or authentication authority and
- Service provider: The SAML relying party that offers services to the principal.

After the authentication of a principal, the identity provider can issue an authentication assertion to the principal which can be used during a service request at a service provider. If the service provider trusts the assertion

issued by the identity provider, an identity federation exists between the two parties.

2.4.4 WS-Roadmap

In 2002, Microsoft and IBM submitted a group of Web Services Security (WSS) specifications [66] to the Organization for the Advancement of Structured Information Standards (OASIS) [20]. In 2006, Microsoft made the WS protocols available in the Windows Communication Foundation [43] as a part of the .NET Framework 3.0.

The first layer of the web services security specifications [51] is the WS-Security layer which specifies a message security model used by all the other security specifications in the WS-Roadmap. It specifies a SOAP-aware mechanism (see section 2.4.1) for signing and sealing parts of SOAP messages [14, pp. 391f.]. Furthermore, it describes how to include security tokens such as Kerberos tickets or X.509 certificates in messages. The current version of the WS-Security standard is 1.1.

Building upon this initial specification, there are policy, trust and privacy layers which again form the basis for the three layers secure conversation, federation and authorization.

Chapter 3

Trusted Computing

The main topic of this chapter is trusted computing. First, it presents the development of trusted computing and gives an introduction to the fundamental features such as roots of trust and transitive trust. Moreover, the Trusted Platform Module is described in detail and the cryptographic components (keys, certificates, attestation mechanisms, protected message exchange) are analyzed. The end of the chapter focusses on the development of applications using the Trusted Platform Module.

3.1 Introduction

3.1.1 Trusted Computing

Nowadays, computers play an important role in enterprises and private households. Even though perimeter security has significantly improved over the last years, every single platform is exposed to threats arising from security vulnerabilities, malware etc. In the networked world it is important to determine whether a system can be trusted. In the broader sense it is also possible to link certain actions (e.g. the execution of an application) to a preconditioned system state. The trusted computing initiative has been established to develop a concept for attesting the trustworthiness of a platform. In order to guarantee this trustworthiness, there are three different requirements which have to be addressed in the desired architecture [65, p. 9]:

- **Unambiguous identity:** Every component (hardware and software) in a trusted platform must be identifiable.
- **Unhindered operations:** Something is trusted if it behaves in an expected manner for a particular purpose.
- **Attestation:** A technique to verify consistent good behavior is required.

In order to establish this confidence in the platform, there has to be a special component which forms the foundation of trust and validates the trustworthiness of all the other components. Realizing this foundation of trust with the help of a software component is not possible as it can easily be influenced. Therefore, the trusted computing group has designed a hardware chip—the Trusted Platform Module (TPM)—that is mounted on the motherboard of a computer to form the foundation of trust. With the help of this module it is possible to verify whether a platform can be trusted.

The significance of trusted computing is steadily increasing; the first devices equipped with a TPM are already available on the market and more and more application scenarios for the TPM are being developed. As a consequence, there are a lot of discussions about the conveniences and risks of trusted computing. There are many different use-cases for the trusted computing technology besides its original purpose. In [62, p. 3] the Trusted Computing Group (TCG) enumerates several fields of application for its technology which are e-commerce, risk management, asset management, security monitoring and emergency response. In addition to these fields of application, scientists are currently developing new usage scenarios such as digital rights management[1], digital signatures, P2P networks, video broadcasting systems or identity management, to name only a few of them. And it is already clear that the usage scenarios for trusted computing will tremendously increase over the next few years.

3.1.2 Development

The development of the trusted computing platform specifications was initiated by the Trusted Computing Platform Alliance (TCPA) that had been founded in 1999. As described in [44, p. 33], the initial key players consisted of HP, IBM, Intel and Microsoft. In the same year, they published the first draft for the specification and invited other companies to join their project. By the year 2002, more than 150 companies had participated in the development process and made the specification an open industry standard. Because of the fact that each decision had to be unanimously, the TCPA came to a standstill [11, p. 619]. As a result, the initiating companies founded the TCG in 2003 in order to replace the TCPA and to take over the responsibilities for the further development of the standards. The current version of the TPM standard is 1.2, which was released in 2003.

Currently, there are 13 different working groups that primarily deal with technical issues. The document [59] gives a brief description of the main topics of each working group. The most important issues are: specification

[1] Even though the assumption that digital rights management has been the initial purpose for the development of trusted computing is wide-spread, the trusted computing group denies working on the development of a digital rights management architecture. However, the TPM can be used in such a scenario.

of the TPM; usage of the TPM in PC clients, storage systems or servers; development of APIs to use the TPM in applications; and adoption of the TCG concepts for mobile devices.

In addition to the development in the TCG, there are further developments that deal with the integration of the TPM in operating systems. In this area, the key players are Intel with its Intel Trusted Execution Technology [24] (the former LaGrande technology) and Microsoft with its Next Generation Secure Computing Base (NGSCB) [41] (the former Palladium technology). Microsoft founded the system integrity team which is currently working on the tasks originally performed by the NGSCB [42]. For the trusted platform services in Windows Longhorn, Microsoft set their sights quite high. The original approach was the creation of a secure computing base running parallel to the regular Windows environment [38, pp. 5ff.]. However, mainly due to severe criticism the first delivery is only a small part of the original concept. In Windows Vista, the BitLocker Drive Enryption uses the functions provided by the TPM. The main functions of the hardware-enabled data protection tool are explained in [40] and [39] as:

- encryption of the Windows volume and
- integrity measurement during the boot process.

3.2 The Trusted Platform

3.2.1 Fundamental Features

A trusted platform has to implement at least the following three basic security features that are described in [62, pp. 5f.] and [44, pp. 31f.] as:

- **Protected capabilities and shielded locations:** A shielded location is a protected storage area designed for sensitive information (e.g. integrity metrics or cryptographic keys). These locations can only be managed by commands called protected capabilities. The main functionality provided by protected capabilities consist of integrity reporting, key management, random number generation and sealing of data.
- **Integrity measurement, storage and reporting:** These activities include the calculation, storage and reporting of metrics out of the status or the characteristics of a certain platform.
- **Attestation mechanisms:** A trusted platform has to provide different forms of attestation mechanisms that allow an external party to verify the accuracy of a certain piece of information known to the TPM.

- Attestation by the TPM: The TPM proves the possession of particular data by applying digital signatures with keys only known to the TPM.
- Attestation to the platform: This operation assures that a platform can be trusted for reporting integrity measurements (metrics describing the state or the characteristics of the platform).
- Attestation of the platform: The platform proves a set of its current integrity measurements.
- Authentication of the platform: This is the process of confirming the platform's identity.

3.2.2 Roots of Trust

In order to realize the fundamental features described in section 3.2.1, the TPM has to have three roots of trust, which are discussed in [11, p. 624]. These roots of trust

> "... are components that must be trusted because misbehavior might not be detected. A complete set of Roots of Trust has at least the minimum functionality necessary to describe the platform characteristics that affect the trustworthiness of the platform." [62, p. 6]

The *Root of Trust for Measurement (RTM)* is a computing engine which offers functions to determine the integrity of a platform's configuration during the boot process. The calculations are performed by the Core Root of Trust for Measurement (CRTM) which is basically a BIOS extension that is executed first during the boot process. The TPM itself offers the necessary functions such as calculation of hash values and storage of the integrity values in the Platform Configuration Registers (PCR). Second, there is the *Root of Trust for Storage (RTS)*, a computing engine which is responsible for the protection of keys and data objects. The whole key management is also situated in its field of duty. The reporting of data protected by the RTS is a task performed by the computing engine *Root of Trust for Reporting (RTR)*. The necessary activity in order to perform trustworthy reporting tasks is attestation.

In addition to the roots of trust, the *Trusted Building Blocks (TBB)* play an important role. As explained in [62, pp. 6f.], a TBB is a part of a root of trust that has neither shielded locations nor protected capabilities, which means that it has to be trusted. Examples for TBBs are the instructions for the RTM and TPM initialization functions. Together, the roots of trust and the trusted building blocks form a trust boundary which facilitates measurement, storage and reporting.

3.2.3 Transitive Trust

Transitive trust plays an important role in answering the question whether a certain application on a platform can be trusted. If a trusted component of the platform measures the trustworthiness of another component, a transitive trust relationship is established between the first trustworthy component and the second. In other words, this means that the second component is accepted as trustworthy because the first component attests its trustworthiness. As a consequence, the trust boundary is extended and further on includes not only the first but also the second component [65, p. 9].

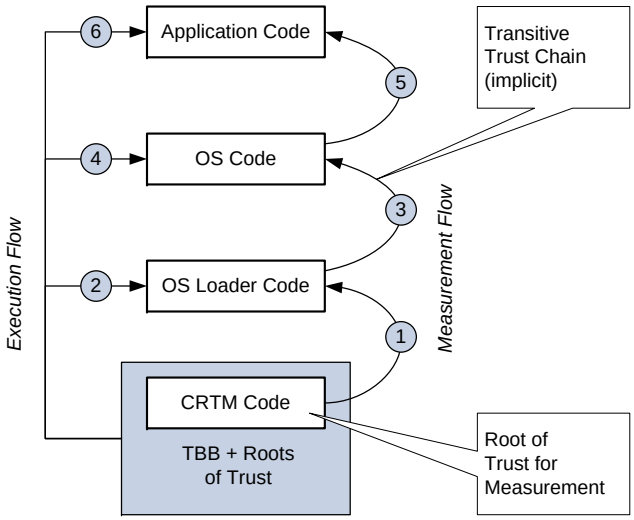

Figure 3.1: Transitive trust enables the extension of the trust boundary. [65, p. 10]

As we see in figure 3.1, the basis of the process for establishing transitive trust is the hardware TPM with its TBB and roots of trust [62, pp. 7f.]. These components form the foundation of trust wherefore they have to be accepted as trustworthy due to practical purposes. The main reasons for accepting these components as trustworthy are the following:

- Technical trust in the design and implementation.
- Social trust in the developers and manufacturers due to their reputation.

In the first step, the roots of trust attempt to determine whether the target code of the upper layer (in our example the OS Loader code) can be trusted. If the result of the measurement is positive, the underlying layer

gives a trustworthy description of the upper layer which allows the external entity to decide whether the trust level is acceptable. In the positive case, the trust boundary is extended and includes TBB, roots of trust and the target code of the upper layer. The next steps are the execution of the target code and the iteration of the whole process until the application layer is reached. By extending the trust boundary from TBB and the roots of trust up to the application, an external entity can be sure that it can trust the application running on the trusted platform as well.

3.2.4 Trusted Platform Module

Basically, the TPM is the chip that covers the hardware part of the trusted computing architecture. In [11, p. 626], the following basic functions of the TPM are enumerated:

- generation of asymmetric and symmetric keys,
- calculation of signatures and hash values,
- asymmetric and symmetric encryption,
- encryption of cryptographic keys (binding),
- secure storage (shielded locations) and processing (protected capabilities) of small objects and hash values (of measurement values from the platform configuration),
- creation of signed reports about these measurement values,
- key management (endorsement and attestation identity keys),
- offering functions for the owner of the platform to take possession of the platform and (de)activate the TPM,
- offering a trustworthy timer.

The logical layout of the TPM according to [62, pp. 19ff.] and [44, pp. 41ff.] is illustrated in figure 3.2. The *input and output* component is responsible for the management of the communication over the bus in addition to encoding/decoding of information and the routing of messages to the other components inside the TPM. Endorsement, storage root keys, owner authorization data and persistent flags are stored in *non-volatile storage*. The *platform configuration registers* (PCRs) can either be implemented in non-volatile or volatile storage. The PCRs are used to store 160-bit hash values which are calculated, among other things, during the secure boot process out of parts of the system configuration. In total there have to be at least 16 such registers inside each TPM. *Attestation identity keys* (AIK) must be

stored in a persistent location. It is recommended to store the AIKs as blobs in persistent storage outside the TPM and to only load them into the TPM if they are needed during execution. The *program code* area stores firmware for measuring platform devices (CRTM). Ideally, this means that the CRTM is stored inside the TPM. Nevertheless, sometimes the CRTM has to be located elsewhere due to technical issues. *Random number generator*, *HMAC engine* and *SHA-1 engine* take over some of the cryptographic work. The *key generation* module is responsible for the creation of signing and storage keys up to a 2048 bit modulus. Signing, encryption and decryption operations are performed in the *RSA engine*. The *Opt-In* component stores the current state of the TPM that ranges from disabled and deactivated to fully enabled. Finally, the *execution engine* is responsible for running the program code.

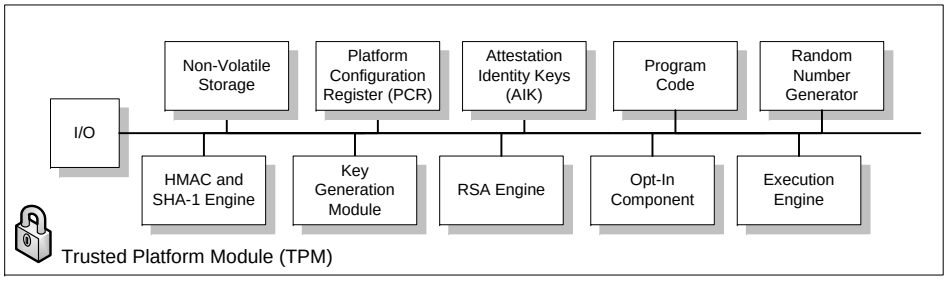

Figure 3.2: TPM component architecture. [62, p. 19]

3.2.5 Operational States

A TPM can enter three different discrete states: *enabled/disabled*, *activated/deactivated* and *owned/unowned*. In the disabled state, a TPM is only able to execute simple commands such as hash calculations, whereas the real functions of the TPM are not available. The inactive state is quite similar except that it is possible for a user to execute the command to take the ownership of the TPM.

As remarked in [63, pp. 45f.], these states can be combined to eight possible operational states for the whole TPM. They range from enabled-active-owned which describes a TPM in the fully operational state in which all TPM functions are available to disabled-deactivated-unowned, which is the default deliverable state. Due to these combination possibilities, the TPM can be modified in a suitable way for each usage scenario.

During the process of taking ownership of the TPM, physical presence at the device is necessary. In [11, p. 632] it is explained that the owner has to

present a password of which an 160-bit authentication value (e.g. SHA-1 hash value[2]) is used as the owner authentication key. This authentication value is encrypted with the public part of the endorsement key pair and stored in the non-volatile memory of the TPM. In case the owner wants to execute commands which require owner authorization, this particular password has to be presented.

3.3 Cryptographic Components

3.3.1 Keys

It is the responsibility of the RTS to protect keys and data objects inside the TPM. In [62, pp. 16f.] it is shown that the RTS manages a small amount of volatile memory inside the TPM in which keys are stored during cryptographic operations. Inactive keys and further data are stored in an external memory in a certain storage hierarchy. As illustrated by figure 3.3, the Storage Root Key (SRK) is used to protect the first layer of the hierarchy. Further down, additional storage keys can be created and used. The key cache manager is responsible for the management of the key slot cache and interfaces with the external storage device. SRK and Endorsement Key (EK) are stored inside the TPM and cannot be removed. A new SRK can only be created if the platform gets a new owner, which unfortunately means that all the data protected by the previous SRK is lost because it is still encrypted with the old key.

Keys managed by the RTS can have two different attributes: migratable or non-migratable. These attributes describe the possibility to transfer the keys from one TPM to another. The attribute is established during the creation of the key and cannot be changed. However, they are not applicable to opaque data stored by the RTS. In order to make opaque data non-migratable, it can be stored using a non-migratable storage key. As long as the particular piece of data is under the control of the TPM and not decrypted externally, it is well protected.

A description of the different key types used by the TPM is given in [11, pp. 632ff.]. Basically they differ in restrictions and application scenarios.

- **Signing keys** are either migratable or non-migratable asymmetric keys which are used for signing of application data and messages.
- **Storage keys** are 2048 bit RSA keys which are used for the encryption of data and other keys stored externally of the TPM.

[2]It has to be remarked that several collision attacks for the SHA-1 hash algorithm currently exist. However, they are not so far developed that the SHA-1 cannot be used any longer. Depending on future developments, another hash algorithm may have to be used in the TPM.

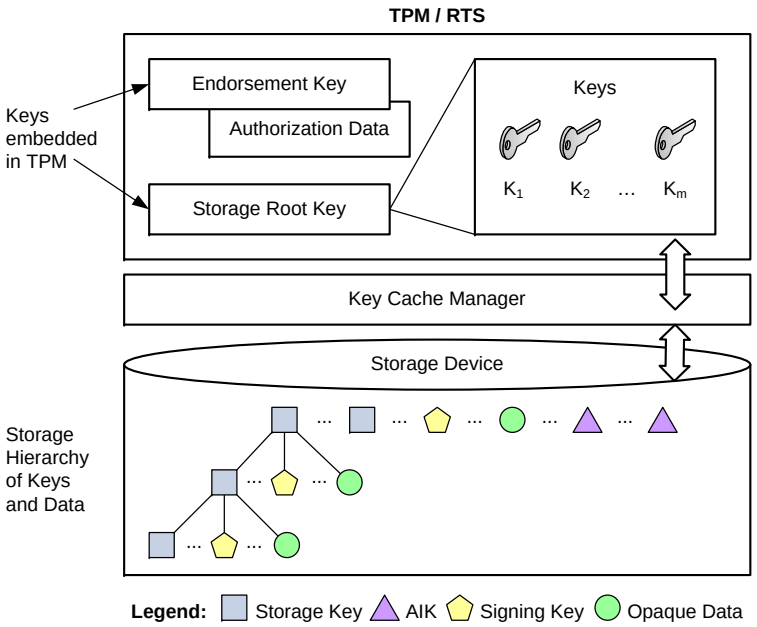

Figure 3.3: The special architecture of the RTS enables the protection of keys and small data objects. [62, p. 17]

- The **endorsement key** is a non-migratable 2048 bit RSA key pair which is created during chip manufacturing. As it is uniquely associated with a TPM, it can be used to verify its authenticity. Furthermore, the EK is used in conjunction with the owner authorization data during the owner establishment process as well as for signing attestation identity keys during their creation.

- **Attestation Identity Keys (AIK)** are non-migratable 2048 bit RSA key pairs which are used for signing only data originated by the TPM (e.g. TPM capabilities, PCR values). It is possible to create an arbitrary number of attestation identity keys after the process of taking ownership of the TPM. Mainly AIKs are used during the attestation process (see chapter 3.3.3) where it is necessary to authenticate the platform or the TPM without revealing its true identity (in particular the EK).

- **Bind keys** encrypt small amounts of data (e.g. symmetric keys) on one platform in order to be decrypted on another one.

- **Legacy keys** are created outside the TPM and are imported for signature and encryption operations. They are by default migratable keys.

- **Authentication keys** are symmetric keys which are used to protect transport sessions between the TPM and an operating system or application process.

3.3.2 Certificates

In order to attest the identity of a TCG platform or the AIK, different certificate types (credentials) are required. Each credential is defined in ASN.1 notation based on X.509 certificate structures [11, pp. 635f.].

- The **endorsement credential** issued by the TPM manufacturer attests that the creation and transmission of the EK to the TPM was performed according to the specification. Among other information, it includes the name of the TPM manufacturer, TPM version number and the public part of the EK. Even though the credential is public information, it should be handled carefully as it allows the direct identification of the TPM.

- Every accredited instance (e.g. manufacturer, distributor) that is able to evaluate a TPM can issue a **conformance credential** for the platform. This credential attests that the design and implementation of the TBB fulfill the specification.

- **Platform (endorsement) (PE) credentials** can be issued by manufacturers, distributors or other trustworthy instances. They identify the manufacturer of the platform as well as some other attributes and include links to the endorsement and conformance credentials of the platform. The platform credential claims that a certain platform has exactly the TPM which is claimed by the endorsement credential.

- In order to attest that certain security critical components (e.g. video-, network adapter, keyboard, mouse or software) are trustworthy (no trojans, backdoors etc.), the **validation credentials** can be issued after the evaluation of the component in a secure environment by the evaluation body.

- The **attestation identity credential** attests and identifies the private part of the AIK. During the creation of the AIK, the TPM sends a request to a trusted third party, the privacy-CA, which includes the public part of the AIK as well as endorsement, platform and conformance credentials. The privacy-CA checks the credentials and issues the attestation identity credential which attests that the certain AIK belongs to a TPM which is conformable to the specification. It has to be remarked that the privacy-CA is the only one (except the TPM) that is able to link EK and AIK. As a TPM can create an arbitrary

number of AIKs, a TPM can use multiple pseudonymous identities that cannot be linked by an external party except the privacy-CA.

3.3.3 Attestation

The attestation mechanism is an integral part of the TPM specification as it allows a TPM to prove its integrity to an external party and as a consequence allows the establishment of a trust relationship between the two parties. Two different attestation mechanisms exist which assure a different extent of privacy. A detailed analysis of the attestation mechanisms can be found in [8] and [4].

Attestation with Privacy-CA

Attestation with privacy-CA is a mechanism used in version 1.1 of the TPM specification that requires a trusted third party, the privacy-CA. Figure 3.4 illustrates the protocol. If the user of a platform wants to attest its integrity to a certain verifier, an AIK (AIK_i) for the specific verifier has to be generated and a certificate (attestation identity credential $Sig_{PrCA}(AIK_i)$) from the privacy-CA has to be obtained by sending EK and AIK_i to the privacy-CA. The appropriate AIK is used to sign the data which describes the integrity state of the configuration (PCR values). Moreover, the AIK_i, the attestation identity credential $Sig_{PrCA}(AIK_i)$ and the attestation value $Sig_{AIK_i}(PCR)$ are sent to the verifier which checks the trustworthiness of the privacy-CA and the attestation identity credential. After this verification, the verifier has to validate the configuration state of the platform (PCR values) and to decide whether or not the platform can be trusted.

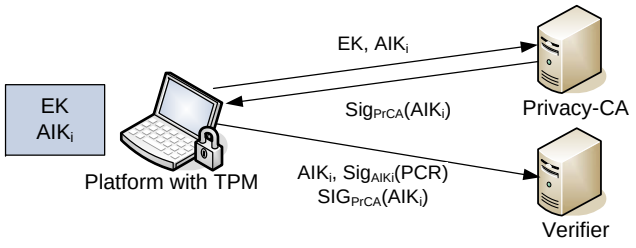

Figure 3.4: Remote attestation with privacy-CA. [8, p. 10]

Direct Anonymous Attestation

Because the privacy-CA is able to link EK and AIK and as a consequence all the different AIKs of a trusted platform, the remote attestation with the

privacy-CA was often criticised. As a result, Direct Anonymous Attestation (DAA) has been introduced in version 1.2 of the TPM specification [4]. Besides increasing the level of privacy, it eliminates the danger of having a single point of failure as it does not require a privacy-CA. Basically, DAA is a group signature scheme which does not allow the opening of signatures (or identifying the signer) but which offers a mechanism for the detection of rogue members (TPMs) [6, p. 2]. In every group signature scheme there has to be a group manager (TPM manufacturer) whose task it is to control the membership for the group (the TPMs produced by this manufacturer). By the usage of group signatures, a member of the group can prove its membership without revealing its true identity. A simplified illustration of the protocol is given by figure 3.5.

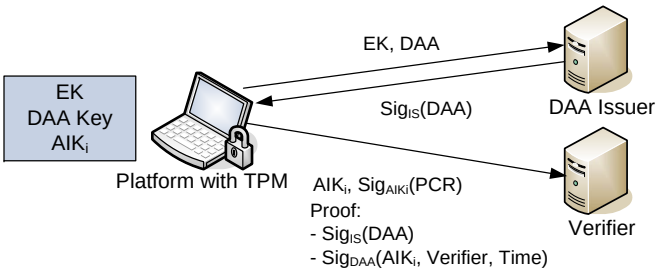

Figure 3.5: Direct anonymous attestation. [8, p. 13]

If a platform wants to perform an attestation operation, it has to generate a DAA key pair and obtain a signature from a DAA issuer who can be for example a manufacturer or a distributor of the TPM. The DAA issuer validates the identity of the platform by using its EK. After a successful validation, the DAA issuer signs the platform's DAA key and returns the DAA credentials $Sig_{IS}(DAA)$ to the platform. During the attestation process, the platform generates an AIK (AIK_i) and signs it with its DAA key ($Sig_{DAA}(AIK_i, Verifier, Time)$). Furthermore, the AIK is used to sign the PCR values. The relevant information is sent to the verifier which has to check whether the platform generated the signature of the AIK by using the DAA key and whether the platform really possesses the DAA credentials. This requirement is accomplished by the usage of a zero-knowledge algorithm which means that the trusted platform proves the knowledge of the DAA keys without revealing them to the DAA verifier. Moreover, the configuration state of the platform (PCR values) has to be validated as well in order to know if the platform can be trusted.

The advantage of using the DAA protocol is the increased level of privacy as the DAA issuer is only able to link the identity of the platform with the DAA key and the verifier only knows the AIK [8, p. 14]. First of all this means

that—even if DAA issuer and verifier work together—during the usage of two AIKs of a single platform, it is not possible to link these keys and second it is not possible to identify the platform if an AIK is used.

3.3.4 Protected Message Exchange

Classic techniques such as asymmetric encryption and signing exist for the protection of the message exchange. Additionally it is important to assure a secure key management and proper configuration at the end points of the communication. A TPM especially supports all of these necessary aspects and offers four different classes of protected message exchange which are enumerated in [62, pp. 15f.].

- **Binding** is the process of asymmetric encryption using a public key.
- **Signing** includes the creation of signatures in the traditional way.
- **Sealing** can be described as an extension of the binding mechanism which allows that the decryption of a message is bound to a specific configuration of the platform. This is achieved by encrypting the symmetric key (used for the encryption of the message) and some specific PCR values with an asymmetric public key.
- **Sealed-signing** allows a signing platform to include its current PCR values in a certain signature. By using this mechanism, a verifier can be informed about the configuration state of the platform during signature generation.

3.4 Applications

3.4.1 Software Layers

In order to use the TPM in applications and to ensure the trustworthiness of the applications, the TCG defines the three software layers and interfaces building on the TPM as shown in figure 3.6. These layers are necessary to encapsulate critical functions so that users of the TPM do not have to programme them over and over again. This leads to a minimization of error sources as the developers can concentrate on the development of the actual program [5].

- **TPM Device Driver and TPM Device Driver Library**: The TPM device drivers are not part of the Trusted Software Stack (TSS). They are provided by the TPM manufacturers for the direct control of the TPM via the TPM Device Driver Library (TDDL) and the TPM Device Driver Library Interface (TDDLI) which is used as an interface for the TSS components.

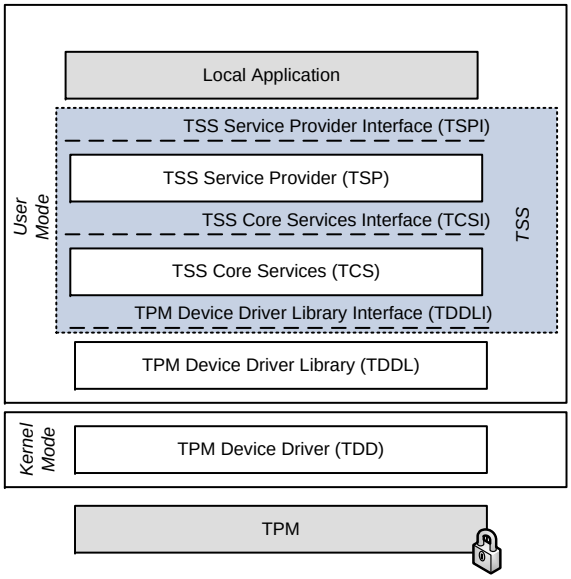

Figure 3.6: TCG software layering. [5]

- **TSS Core Services**: The TSS Core Services (TCS) are an intermediate stage used to control the execution of the critical hardware-near commands. Mainly, its tasks are management functions such as synchronization, memory management or key management. Depending on the operating system, they are either implemented as a service or a daemon.
- **TSS Service Provider**: The TCG Service Provider (TSP) is implemented as a service or a daemon. Usually, multiple TSPs are provided for each TPM. According to an object-oriented approach, for each application a unique module is loaded. The TCS is responsible for the coordination of the modules whereas the TSP offers multiple functions which partially exceed TPM functionality (e.g. formatting of TPM data in a better readable format). The TSS Service Provider Interface (TSPI) is the interface used by the local applications to integrate the TPM.

3.4.2 Command Validation

For all the commands which affect security, privacy or reveal platform secrets, authorization is required [44, pp. 52ff.]. In a TPM, there are different types of the 160 bit authorization data: unique owner authorization data (commands which require TPM owner authorization), TPM object usage

authorization data and TPM object migration authorization data. Two different protocols can be used in order to prove knowledge of the adequate authorization data:

- **Object-Independent Authorization Protocol (OIAP)**: The OIAP is a challenge-response protocol which can be used to prove knowledge of authorization data associated with multiple TPM objects once the OIAP session has been established.
- **Object-Specific Authorization Protocol (OSAP)**: In contrast to the OIAP, the OSAP can be used to establish an authorized session to exchange authorization data for only one specific TPM object.

In addition to the protocols to establish secure sessions, protocols to manage the authorization data are also required. The TCG specifies three mechanisms: Authorization Data Insertion Protocol (ADIP), Authorization Data Change Protocol (ADCP) and Asymmetric Authorization Change Protocol (AACP) (which is deprecated in version 1.2 of the TPM specifications).

Chapter 4

Trusted Infrastructures for Identities

This chapter discusses the problem of establishing mutual trust relationships between different identifier domains in identity management architectures. As a part of this discussion, the basic requirements for the establishment of trust relationships by using trusted computing technology are identified. Possible solutions for the realization are presented in order to select one of the scenarios. In addition, the selected scenario is analyzed with regard to the previously described identity-related aspects. Finally, possible usage scenarios and application fields (apart from the establishment of trust relationships) are presented.

4.1 Problem Overview

One goal of identity management is the establishment of identifier or trust domains where participants trust each other. Traditionally, the trust relationship is based on the fact that the participants know each other because they, for example, belong to the same company, department or university. Furthermore, multiple business-related enterprises with mutual agreements can establish trust relationships. Another example for an identifier domain are all the e-government solutions in a country where it is essential that an identity and the belonging credentials can be used to access multiple services.

In such a traditional architecture, a domain consists of multiple service providers and one or more identity providers depending on whether a federated or centralized approach is chosen (see section 2.1.2). Figure 4.1 shows two identifier domains with a centralized IMA. Even though only the centralized approach is presented, the same problems exist if a federated IMA is used instead. *Identifier and credential provider 1* is responsible for the access to *service provider 1 and 2* and *identifier and credential provider A*

is responsible for *service provider A and B*. It is only possible to use an acquired ticket internally in the particular domain.

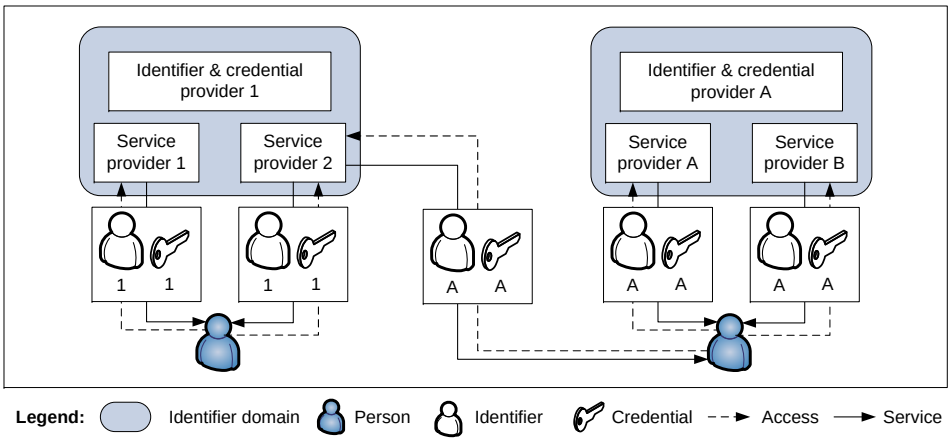

Figure 4.1: Identity management with multiple identifier (trust) domains.

In order to use tickets across multiple identifier domains, trust relationships between the domains have to be established. Traditionally, in PKI-based environments, cross certification (one certificate authority signs the public key of the other certificate authority) or the usage of a spanning certificate authority (the spanning certificate authority signs the public key of the underlying certificate authorities) are means to establish mutual trust between the domains. Furthermore, the mirroring of user databases enables an identifier domain to accept identities of another domain. Even though all these techniques lead to the establishment of mutual trust, the technical overhead is especially high the more identifier domains are involved. Therefore, another approach is the usage of the architecture provided by the trusted computing technology to establish trust relationships (see figure 4.1). A user belonging to *identifier domain A* can use a ticket issued from *identifier and credential provider A* to access a service at *service provider 2* from another identifier domain. To decide whether the foreign ticket can be accepted, the following two requirements have to be fulfilled and verified by the service provider:

1. The identifier and credential provider is authorized to issue tickets for the identifier domain the service provider belongs to.

2. At the moment of ticket issuing, the identifier and credential provider (in particular its configuration and system status) was trustworthy.

With the help of trusted computing technology these requirements can be fulfilled and as a consequence, a trust relationship between the identifier domains can be established. As the significance of trusted computing is steadily rising, the usage of this technology enables a flexible integration of the concept in current identity management architectures because the already established infrastructure for trusted computing can be used. The next section presents different approaches for realizing the requirements.

4.2 Basic Scenario

The basic scenario without the integration of status information and the establishment of trust relationships between multiple identifier domains is illustrated in figure 4.2. It shows a user acquiring a ticket from an identity provider (PEP and PDP in identifier domain A) in order to use it during a service request at an external service provider (PEP in identifier domain B).

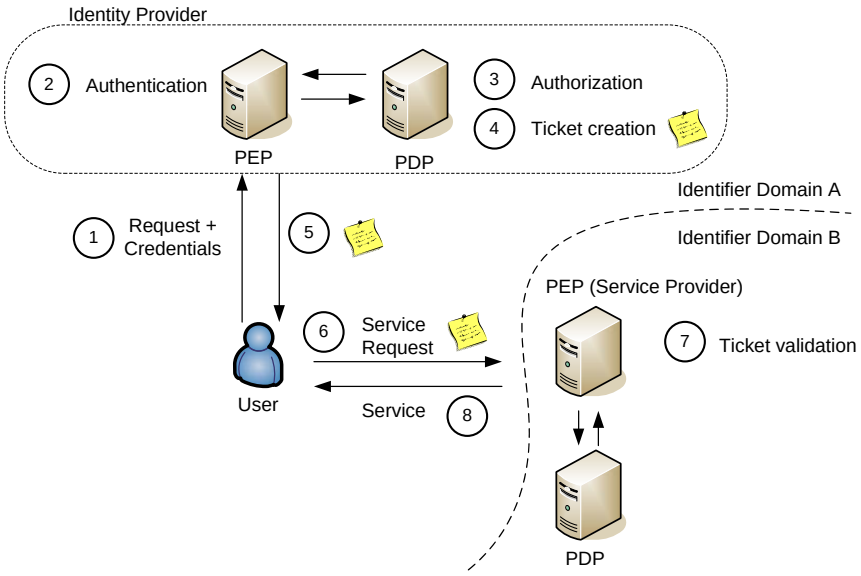

Figure 4.2: Basic IDM scenario.

1. User authentication consisting of request and presentation of the credentials at the Policy Enforcement Point (PEP).
2. The PEP authenticates the user possibly with the help of an authentication server.
3. After the authentication, the Policy Decision Point (PDP) is responsible for the authorization decision according to a predefined policy.

4. Next, the ticket for the user is created.
5. The ticket is sent back to the user via the PEP.
6. In order to request a service from a service provider (PEP) potentially located in another identifier domain, the ticket has to be presented by the user.
7. The PEP validates the ticket with the help of a PDP and the predefined policy in order to decide whether or not the desired service can be granted.
8. Depending on the validation process, the user is granted access to the service or not.

It is important to note that the ticket validation of the PEP in identifier domain B is based on the policies valid in this domain. Theoretically, it is possible that the PEP contacts the issuing PDP in domain A in order to take the authorization decision. However, in this case the PEP would not be located in a foreign identifier domain anymore but instead in the issuing identifier domain A. Therefore, the policy of domain B has to be used to take the decision. In further scenarios, a simplified architecture is used. PEP and PDP from identifier domain A are summarized as identity provider and the PEP in identifier domain B is the service provider. The PDP from identifier domain B is not part of the diagrams.

4.3 Integration of Status Information

A method to prove the trustworthy status of the identity provider is the integration of status information in the tickets. All in all, multiple techniques exist to measure the status of a platform. Two interesting approaches for integrity measurement are based on the running software and applied configurations in a system, or on application independent system properties.

4.3.1 Component-aware Integrity Measurement

The TCG defines integrity information management as the process of providing reliable and authentic integrity data about the components of a trusted platform that can later be verified. A trusted platform consists not only of hardware components but also of software. Consequently, integrity measurement has to include both, hardware and software. Figure 4.3 illustrates the process of integrity measurement. The foundation of the whole process are the measurement values:

- **Runtime/Loadtime Measurements**: These types of measurements describe the status of a running system. Loadtime measurements are

taken during the boot-up sequence which means that they describe the pre-operating system state. On the contrary, runtime measurements generally refer to the integrity of the components during the operation of the platform. In addition to this general distinction, the loadtime measurements are taken by the RTM whereas an operating system part is responsible for the creation of runtime measurements.

- **Reference Measurements**: In order to verify the trustworthiness of a platform, there has to be a way to validate the runtime and loadtime measurements of a platform. The solution for this problem is the creation of reference measurements which describe the desired state of each component.

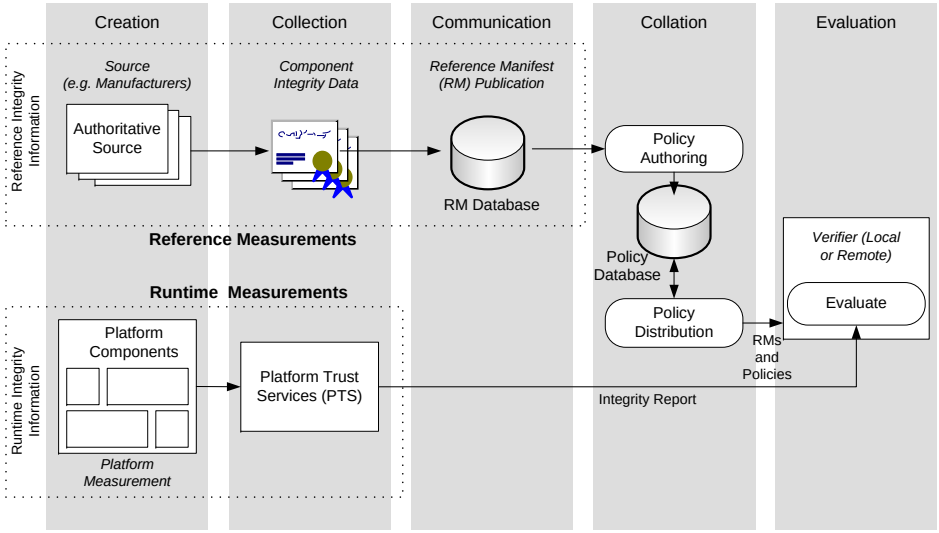

Figure 4.3: Runtime/Loadtime measurements and reference measurement processes. [65, p. 16]

As illustrated by figure 4.3, authoritative sources (e.g. the manufacturers of the specific component) are responsible for the creation of reference measurements. They collect and process (usually in XML format) the component integrity data and publish it in a Reference Manifest (RM) database. As the next step, the relevant integrity data for a target platform is gathered. The goal of this collation phase is also the eventual verification of the integrity information. During the evaluation phase, the reference measurements are compared with the runtime and loadtime measurements. The attesting platform sends an integrity report consisting of its actual measurement values

collected by the platform trust services (integrated within the trusted operating system) to the local or remote verifier who is responsible for the evaluation.

Basically, the measurement value of a component is its SHA-1 hash value which is stored in a particular PCR. According to [61, pp. 31ff.], the measurement values are stored in the PCRs of the TPM. Totally there have to be at least 16 PCRs available which are reserved for loadtime (PCR0-7) and runtime (PCR8-15) measurements. As there are more than 16 components demanding integrity measurement, the TCG specifies a special process to calculate the PCR values [65, pp. 26f.]. The old value of the PCR (initial value is zero) is concatenated with the new measurement value and hashed in order to get the new value for the PCR. Since the entries of the PCR change with every new measurement value, a mechanism to keep track of the measurement values of each component is required in order to evaluate the PCR values. These tasks are performed by the integrity measurement log which stores all the hash values together with additional information describing the event. Generally, it has to be stated that the calculation of the loadtime measurements is already implemented by the current TPMs available on the market whereas the operating system support for the creation of the runtime measurements is not yet available. In the area of runtime measurements, research is ongoing in the TCG.

During the attestation process, the values of the PCRs and the integrity measurement log have to be delivered to the verifying platform. As the values are signed with the AIK, the verifier has to check the signature as well as the correlation of the PCR values with the integrity measurement log. Furthermore, the integrity measurement value of each component has to be validated based on the reference measurements.

A similar approach as described by the TCG is the integrity measurement architecture presented in [54] and [53] which enables the attestation of system runtime properties. In contrast to the TCG specification, the project team not only developed the specification but also implemented a reference architecture in the linux kernel. The operating system is responsible for taking the integrity measurements before the execution of content. As support, the TPM is used to create, store and protect the measurement values. As the data is measured before the execution, problems occurring during the runtime (buffer overflow exploits, SQL injection etc.) cannot be detected in the measurement values. However, an external verifier can use public databases with known vulnerabilities (e.g. CERT) to distinguish whether a component is vulnerable. In the presented architecture, kernel modules, executables and shared libraries, configuration files, and other important files (Bash command files, Java servlets, Java libraries etc.) are included in the integrity measurement process.

4.3.2 Property-based Integrity Measurement

The problem with the integrity measurement presented by the TCG is that it is platform specific and extremely complex due to the possibility of multiple configuration files or software updates. Consequently, problems such as discrimination of certain vendors, information disclosure (as the verifier knows the specific platform configuration), or a colossal measurement database could arise. A different approach is presented in [52] which is independent of the software configuration because it only focuses on the properties a platform offers. Normally, an external party is not really interested in the specific system configuration but in the question whether the platform properties fulfill certain (security) requirements. A platform property is defined as

> "... a quantity that describes an aspect of the behavior of that platform with respect to certain requirements." [52, p. 2]

In order to define the platform properties, an approach similar to the Common Criteria Protection Profiles can be used. Properties are summarized in a property profile which includes the specific requirements according to the particular purpose (e.g. DRM platform, identity provider). For the transformation of the platform configuration into properties and vice versa, either a trusted third party or the platform itself is responsible. Even though the version without the trusted third party offers more security, it is not currently feasible. However, in the long-term, developments in areas such as proof-carrying code [47] could make improvements.

4.4 Authorization of the Identity Provider

In order to realize the authorization of the identity provider to issue trusted tickets, several approaches are feasible. The following sections present and discuss different strategies.

4.4.1 Digital Rights Management

One possible strategy to assure that only authorized identity providers are able to issue trusted tickets is the utilization of a digital rights management solution. Its task is the protection of the adopted identity management application which implements functionality to issue trusted tickets. If a service provider receives one of the trusted tickets, it can be sure that it originates from a trusted identity provider as the ticket issuing application itself is protected and cannot be copied to another machine. An additional requirement is the embedding of information about the application in the ticket which can be realized with the help of cryptography (e.g asymmetric signatures or encryption).

The major advantage of this concept is that it requires very few adoptions in the identity management protocol messages and moreover it is not necessary to change the trusted computing architecture. However, an additional technology has to be used which significantly increases the maintenance overhead.

4.4.2 AIK Credentials and Privacy-CA

Another possibility is to use the AIK credentials and the privacy-CA to establish a trust relationship between the identifier domains. This scenario (see figure 4.4) allows the optimal usage of the trusted computing infrastructure. The privacy-CA plays the major role in authorizing the identity providers to issue trusted tickets based on the vendor certificates in the EK credentials. Alternatively, a more subtle authorization is possible depending on the public parts or serial numbers of the EK.

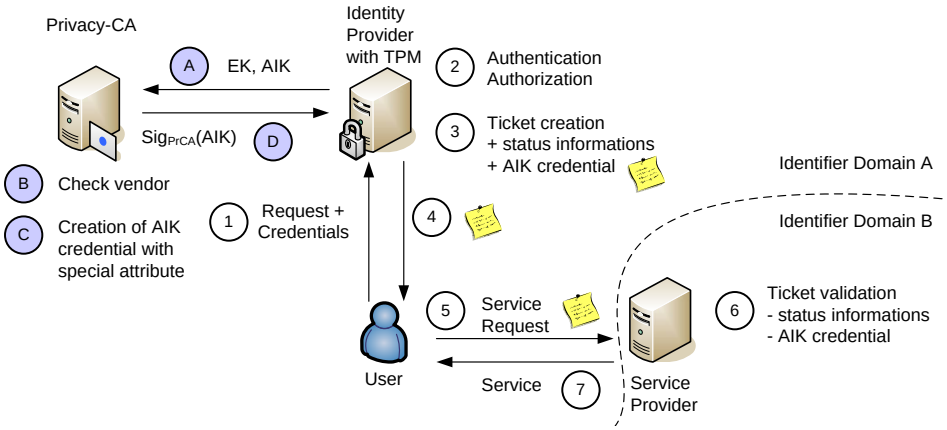

Figure 4.4: Trust establishment with AIK credentials and the privacy-CA.

As a basic requirement, each identity provider has to be equipped with a TPM that forms the base for the trust establishment. Further, the process of the establishment of the AIK credentials and the tasks of the privacy-CA have to be slightly adapted.

A. The identity provider sends a request consisting of EK credential and the public part of the AIK to the privacy-CA.

B. The privacy-CA checks the vendor certificate in the EK credential and decides whether TPMs issued by this vendor are allowed to issue trusted certificates.

C. In the case of a positive decision, the privacy-CA issues an AIK credential containing a special attribute attesting that the identity provider owning the certificate is authorized to issue trusted tickets.

D. Finally, the AIK credential is sent back to the identity provider and can be used during the identity management process.

In the identity management architecture, there are only some minor adaptions necessary. The main task is the integration of authorization and status information in the created ticket.

1. - 2. Request, authentication and authorization.

3. In order to embed the authorization information in the ticket, the AIK credential is used. Additionally, the status information describing the trustworthy status of the identity provider has to be included in the ticket.

4. The ticket is sent back to the user.

5. The user requests a service and includes the issued ticket.

6. During the validation of a ticket, the service provider can check whether the AIK credential in the ticket contains the necessary attribute and is issued by a trustworthy privacy-CA. Furthermore, the status of the identity provider during the issuing of the ticket has to be checked.

7. The service is either granted or refused.

One of the advantages of this approach is that the infrastructure provided by the trusted computing architecture is directly used to establish trust relationships between multiple identifier domains. The authorization of the identity providers is centrally controlled by the privacy-CA and it is only necessary to adapt the AIK credentials whereas the EK credentials remain untouched. However, the syntax of AIK credentials has to be extended which means that some changes in the original design by the TCG are inevitable.

4.4.3 DAA Credentials and DAA

In comparison to the establishment of mutual trust with AIK credentials and the privacy-CA (see section 4.4.2), the realization with DAA credentials and DAA (section 3.3.3 gives a detailed description of the attestation mechanisms) is more complicated. That is because during the DAA the TPM does not simply reveal the certificate obtained by the DAA issuer but instead only confirms the possession during a zero knowledge protocol. Therefore, the identity provider has to prove that the DAA credential contains the necessary attribute without publicly showing the credential.

A protocol to protect anonymous credentials with the Trusted Computing Group's TPM is discussed in [7]. The authors propose a mechanism which, among other things, allows the extension of DAA credentials to include specific attributes which can be anonymously verified during the DAA. During the creation of the DAA credentials, the TPM has to include the additional attributes in the request. Depending on the use case, it is either possible to send them directly to the DAA issuer or to keep them as a secret and only prove the possession with a zero knowledge protocol. After verifying that the TPM created all the attributes and parameters correctly, the DAA issuer signs the additional attributes in the credential. During the attestation, an external party can verify whether the TPM obtained a signature over those attributes from the DAA issuer. Again, it is either possible to let the verifier learn the attributes or to only prove the possession with the help of the zero knowledge protocol.

The proposed realization architecture from figure 4.4 only has to be slightly adapted. Instead of the privacy-CA, the DAA issuer issues the DAA credentials for the identity provider. The identity provider includes the necessary attributes in the request sent to the DAA issuer. After the authorization validation, the DAA issuer creates the DAA credentials for the identity provider which include the signed attributes attesting that the identity provider is allowed to issue trusted tickets. During the creation of the trusted ticket, these attributes are included with the help of the mechanisms presented in [7].

4.4.4 Additional Certificates

The usage of additional certificates allows the most accurate authorization of identity providers. Figure 4.5 illustrates the scenario. Instead of the already existing trusted computing architecture and the AIK credentials, an additional public key infrastructure is established. Each identity provider contains a TPM, which is used to integrate the status information in the tickets, and an additional certificate indicating the right of the identity provider to issue trusted tickets. In order to guarantee the security, the certificate as well as the corresponding public and private keys have to be protected by the TPM. It is especially important that the certificate and the keys are not migratable. Either these additional certificates are already installed in the TPM during the manufacturing process or later, when the platform is already in use.

1. - 2. Request, authentication and authorization.

3. During the ticket creation, the status information and the certificate testifying the right of the identity provider to issue tickets are embedded.

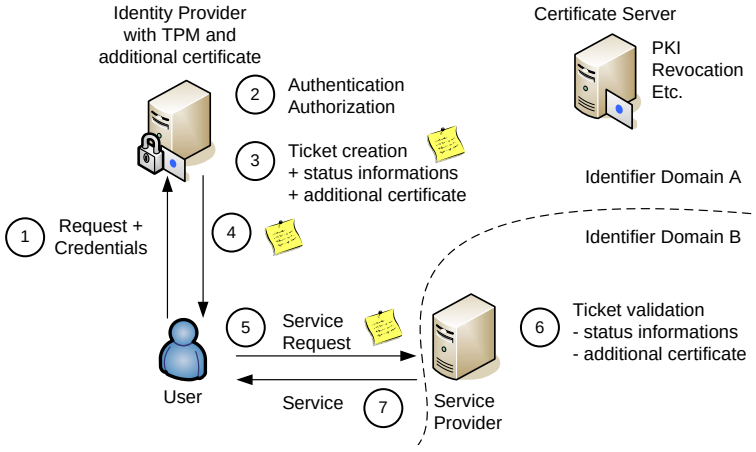

Figure 4.5: Trust establishment with additional certificates.

4. - 5. The ticket is sent back to the user who requests a service with the help of the ticket.

6. The service provider has to validate the status information and the additional certificate included in the ticket. In order to validate the ticket, the certificate server which issued the certificate has to be consulted.

7. The service is either granted or refused.

A benefit of using this scenario is that it can be used independently from the trusted computing architecture. On the contrary, an additional public key infrastructure has to be established and most of all maintained. This means that the administrative overhead is significantly increasing. Another advantage is that the authorization is not based on vendor certificates but individually on each TPM possessing such a certificate. However, this authorization scheme can also be depicted by the scheme described in section 4.4.2 if the privacy-CA uses the values of the public part or the serial number of the EK to decide whether or not to authorize an identity provider.

4.5 Selected Scenario

After the analysis of several possible realization scenarios, a strategy for the integration of status information and the establishment of mutual trust has to be chosen in order to develop the detailed concept. Due to the following reasons, the component-aware integrity measurement (see section 4.3) is selected for the integration of status information, even though the property-based approach offers more vendor independence:

- The development of the component-aware architecture is significantly driven by the TCG and the operating system vendors.
- It is more likely that an operating system supporting the component-aware integrity measurement will be available on the market in the near future.

However, the operating system supported calculation of measurement values will not be part of the realization concept. Instead, only the PCR values and a virtually available integrity measurement log will be taken into account and transmitted from the identity to the service provider.

The establishment of mutual trust relationships will be based on the AIK credentials and the privacy-CA (see section 4.4.2). The reasons are the following:

- In contrast to some of the other approaches (Digital Rights Management and additional certificates), it allows an optimal usage of the trusted computing architecture which is used for the realization of both requirements: authorization of identity providers and integration of status information.
- At the moment, it is wiser to use attestation with the privacy-CA instead of DAA particularly with respect to the reference implementation. The vendors are currently working on products for the privacy-CA which implies that they will focus on DAA afterwards.

Theoretically, it is possible to adapt the concept in order to use DAA instead of attestation with the help of the privacy-CA. In this case, a DAA issuer would take over the tasks of the privacy-CA. A detailed analysis of this approach is presented in section 4.4.3.

4.6 Identity-related Aspects

As the selected concept extends the conventional identity management architecture, in some cases special adaptions are necessary to fulfill the identity-related aspects defined in section 2.3. Hereafter, the aspects are analyzed.

4.6.1 Trust

The establishment of the trust relationship is based on the two assumptions that the identity provider is authorized to issue trusted tickets and that its state during the issuing of the ticket is trustworthy. During the identity management process, a transitive trust relationship is established. This is because the service provider trusts the decision of the privacy-CA which however trusts the identity provider. As a consequence, the service provider trusts the identity provider because of the privacy-CA. Additionally, the

status information integrated in the ticket has to be valid in order to amplify the trust relationship.

4.6.2 Access-Control Policies

As an identity provider cannot predict exactly which service provider the user needs the ticket for, the access-control policies have to be defined with respect to this requirement. This means that it is not possible to specify the access rights for a particular server; instead they have to be generic so that they can be used for multiple servers offering certain services.

4.6.3 Privacy

First, the privacy of the identity provider with regard to the service provider has to be considered. The question arises which amount of privacy is required for the identity provider in this scenario. If pseudonymity is necessary, it is possible to realize it due to the usage of the AIK credentials. A service provider only knows the AIK credential of the identity provider, which can only be linked to its true identity by the privacy-CA. It is not important for the service provider to know who the identity provider is but only whether it is authorized to issue the tickets. Depending on the standard used for the transmission of the tickets (e.g. SAML), the issuer (SAML authority) potentially has to be declared. If necessary, a pseudonym can be used instead of the true identity in the ticket.

Secondly, the privacy of the user during the usage of a ticket plays an important role. During the communication with the identity provider (authentication and authorization), only pseudonymity is possible due to the authentication with the credentials. Depending on the use case, even pseudonymity might be impossible if, for example, credit transactions are involved. However, during the usage of the ticket at a service provider, the true identity of the user does not matter unless the service provider is responsible for accounting tasks. In other words, the amount of privacy again depends on the use case.

In general, privacy enhancing techniques can be additionally used during the data transmission between the parties. Even if the identity of a particular party does not have to be revealed it might be possible to deduce its identity by analyzing the network traffic.

4.6.4 Authentication

For the authentication process, no special changes in the regular architecture are necessary. The user has to authenticate at the identity provider before a ticket is issued. It is irrelevant which authentication factors are used as long as they are strong enough to provide a sufficient amount of security.

4.6.5 Authorization

Several parties are involved in the authorization process. One of these places is the identity provider that includes authorization information in the tickets based on the access-control policies. The other party is the service provider that validates the authorization information stored in the ticket and then decides whether a user is allowed to access a certain service. Furthermore, it is the service provider's responsibility to check whether the identity provider's decision can be trusted, which is generally the prerequisite for accepting the ticket.

4.6.6 Integrity, Non-Repudiation and Confidentiality

Mainly, the transmission of the ticket from the identity provider to the user and further on to the service provider are the parts where integrity, non-repudiation and confidentiality have to be especially considered.

A main requirement is the integrity of the ticket, which means that it cannot be altered once issued by the service provider. Digital signatures are a technique that can be used in this context. A possible solution would be the usage of the AIK to sign the data, but unfortunately these key pairs can only be used to sign data originated by the TPM. However, it is possible to use signing keys created in the TPM and certified by the AIK (the private AIK is used to sign statements about the properties of the asymmetric non-migratable signing key) [44, p. 49].

Additionally, the status information stored in the ticket is protected with a signature with the TPM's AIK. This mechanism assures that the identity management application or any other application or entity cannot alter these measurement values.

In addition to the integrity protection of the ticket, there has to be a prerequisite which assures that no unauthorized entity can create AIK certificates with the additional attribute. Therefore, a service provider validating a ticket must not forget to validate the certificate issuer and its signature in the AIK credential as well.

Depending on the usage scenario, non-repudiation may require special consideration. If the ticket authorizes the user to access a certain service (e.g. in a certain time slot) it is not problematic if the user presents it several times even at different servers. However, if the user is only allowed to use the ticket once, the problem is more difficult. As the identity provider does not know for which service provider the user needs the ticket, the service provider cannot be specified in the assertion. This means that the user can send a ticket to multiple service providers and it is not enough if the service provider only checks whether a certain ticket has already been received. Instead, the service provider must also verify whether the ticket has been used at another service provider so far. A possible solution for this

problem is the usage of the privacy-CA acting as the central authority for the devaluation of the tickets.

Should ticket transmission need to be carried out confidentially, encryption techniques are employed. As the identity provider does not know for which service provider the user needs the ticket, endpoint-to-endpoint encryption is not possible. Instead, the ticket can be encrypted partially, on the way from the identity provider to the user and from the user to the service provider. If the ticket is specified in XML language, either the whole ticket or several parts can be protected with the help of XML encryption.

4.7 Usage Scenarios

In the identity management area, the purpose of this concept is the establishment of trust relationships between multiple trust domains. Without significant overhead, a service provider belonging to a certain identifier domain can trust the decisions of an identity provider in another domain. Above all, the service provider can be sure that the identity provider is trustworthy, which means that its applications and configurations are secure. Especially in situations where it is not possible to install overall protection mechanisms for the identity provider, the integration of status information allows a foreign service provider to estimate the status of the machine.

In addition to the primary purpose, the scenario can be adapted for the usage in other areas. An advantage of the scenario is that after the issuing of a ticket, it can be used anonymously for accessing a service at a certain provider. Theoretically, this concept can be combined with a payment system. Each ticket could have a particular monetary value or allow the access to a chargeable service. Unfortunately the problem arises that a ticket could be used multiple times at different service providers. In this case, non-repudiation has to be considered. As discussed in section 4.6.6, the privacy-CA could take over the tasks of ticket devaluation. Especially in this scenario it is important to trust the authorization decisions of a foreign identity provider because money is involved. Moreover, the service provider has to perform accounting tasks with the identity provider that originally got the money from the user. In this case, the identity provider would be able to link the identity of the user with the accessed service at the service provider which could lead to privacy problems in case of a compromised identity provider. As a consequence, the authorization to issue tickets would have to be revoked.

Chapter 5

Realization Concept

In this chapter, the concrete realization concept for the selected scenario is presented. First, the communication architecture with its components is described and afterwards, those protocol messages which have to be adapted or newly created are presented. Moreover, the functionality required by the Trusted Platform Module (e.g. key generation, signature calculation) is discussed.

5.1 Architecture

The analysis of possible architecture scenarios in chapter 4 resulted in the selection of the component-aware integrity measurement and the establishment of mutual trust relationships based on the AIK credentials and the privacy-CA (see section 4.5 for details concerning the decision). Basically, the scenario consists of two main parts:

- Authorization of the identity provider with the help of the privacy-CA during the creation of AIK credentials.
- Creation of trustworthy tickets during the identity management process.

Before the identity provider can issue trusted tickets, the necessary AIK credentials have to be acquired. The authorization of the identity provider with the help of the privacy-CA during the creation of AIK credentials is illustrated in figure 5.1.

1. First, the identity provider creates an AIK pair and a request for the AIK credential consisting of the endorsement (includes the public endorsement key), platform and conformance credential and of the signature with the private AIK over the public AIK. This request is encrypted with a symmetric key G, which is encrypted with the public key of the privacy-CA and additionally included in the request.

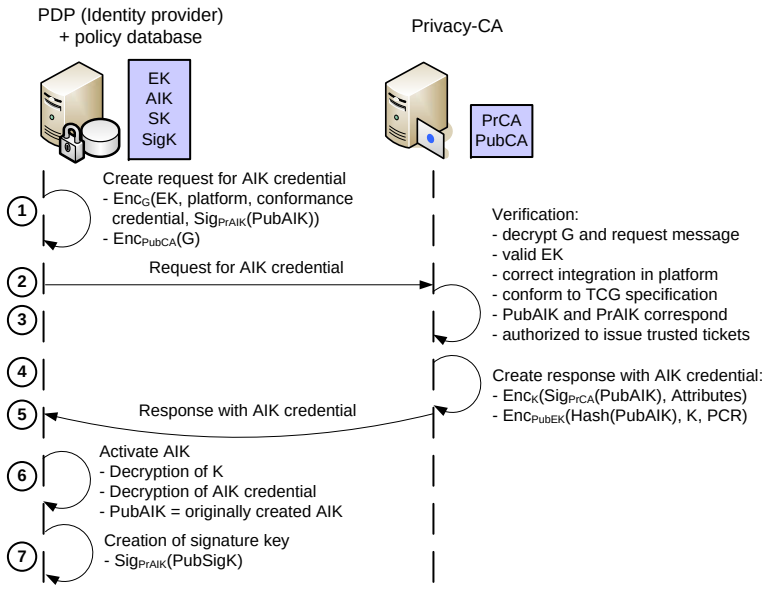

Figure 5.1: Implementation architecture for the creation of AIK credentials with the help of the privacy-CA.

2. The finished request $Enc_G(EK, PE, CE\ cred, Sig_{PrAIK}(PubAIK))$ and $Enc_{PubCA}(G)$ is sent to the privacy-CA using a suitable credential management protocol. In [64, p. 43], the TCG proposes the usage of the Certificate Management Messages over CMS protocol for X.509 certificates and the XML Key Management Services protocol for XML certificates.

3. Upon receipt of the request for the AIK credential, the privacy-CA has to perform the necessary decryption operations and verify that the EK in the request is valid. With the help of the platform and the conformance credentials, it can additionally verify that the TPM is correctly integrated in the platform and working according to the TCG specification. Furthermore, the correspondence of the public and private AIK is verified with the help of the TPM's signature with the private AIK over the public AIK. In other words, the privacy-CA verifies the TPM's signature over the public AIK. The last step of the verification is the decision whether the identity provider is authorized to issue trusted tickets, which is taken either based on the vendor certificate in the EK credential or on the public EK value. Using the public EK value allows a more specific decision because TPMs from the same vendor can be treated differently.

4. After the verification, the privacy-CA creates the response with the AIK credential, which—in addition to the regular attributes—contains a special certificate attribute indicating the authorization to issue trusted tickets (see section 5.3.2 for details) and the public AIK signed with the privacy-CA's private key. As a protection, the credential is encrypted with a newly generated symmetric key K. Additionally, a hash of the public AIK, the symmetric key K and optionally some PCR values are encrypted using the public EK. If PCR values are included in the encrypted value, the receiver (identity provider) has to be exactly in this specified platform status in order to be able to decrypt the message.

5. Next, the two response values are sent back to the identity provider which has to activate the AIK.

6. After the decryption of the symmetric key K, the TPM can decrypt the AIK credential. The identity provider has to verify that the public AIK in the credential is indeed the originally created one. Finally, the AIK can be registered in the persistent storage of the TPM and used by the TPM.

7. During the identity management process, several signature keys are needed to protect the messages. As the AIK cannot be used to encrypt or sign arbitrary messages, an additional signature key is created and attested by the AIK with the help of a signature with the private AIK over the public part of the newly generated signing key. This attestation states that the signing key has been created by a TPM and that it is being protected by the TPM. For further usage, the signing key is stored in the persistent database of the TPM wrapped with the storage key (SK).

A problem during the described process of generating AIKs is that it is not possible for the privacy-CA to verify that the received AIKs have been created by a TPM. This means that an attacker could intercept an AIK credential request and replace the AIK in the request with an externally created key pair and let the privacy-CA issue a credential for these faked AIKs. Even though it is not possible for the attacker to activate these AIKs created outside the TPM, the mere possibility is undesirable. A simple adaption of the protocol which prevents this problem is presented in [15].

The creation of trustworthy tickets during the identity management process is shown in figure 5.2.

1. A user accesses a source site at the policy enforcement point (PEP) which in this case is a typical web server.

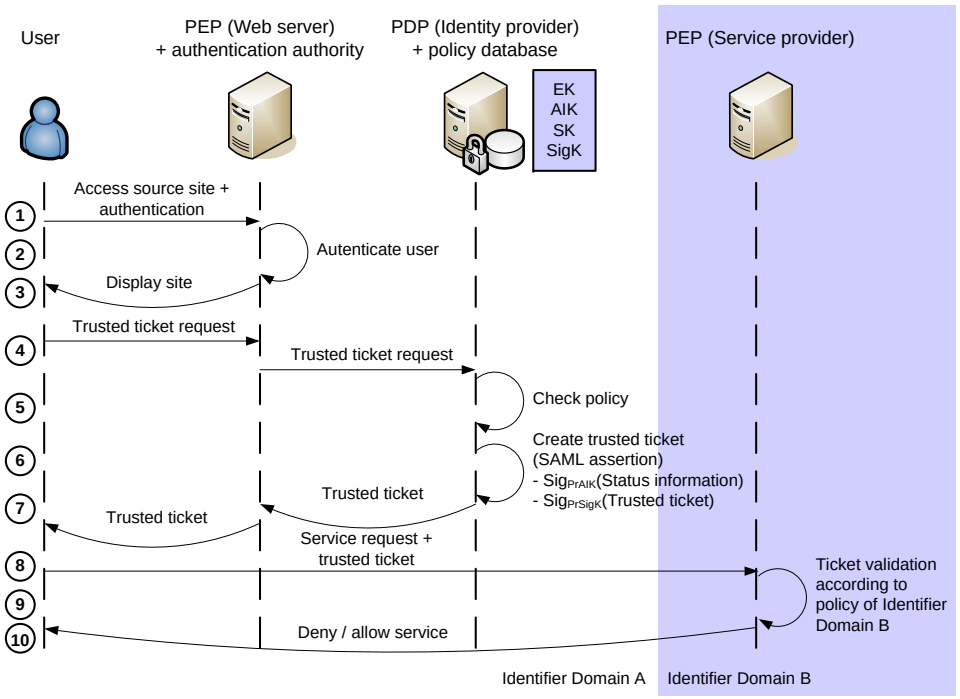

Figure 5.2: Implementation architecture for the trusted identity management infrastructure with the help of the TPM.

2. - 4. After the successful authentication (optionally with the help of an authentication authority), the web server displays the site for the user, who, as the next step, can request a trusted ticket for a service provider in a possibly foreign identifier domain. The trusted ticket request is forwarded by the web server to the policy decision point (PDP), the identity provider.

5. In order to decide whether the user is authorized to access the desired service, the policy (stored in the policy database) is checked by the identity provider.

6. Next, the trusted ticket, a SAML assertion (see section 5.3.3 for details), is created by the identity provider. The ticket has to contain status information about the identity provider which is signed with the identity provider's AIK. Due to integrity issues, the whole ticket has to be signed with the previously created signature key.

7. Furthermore, the trusted ticket is forwarded to the user via the web server (Browser/POST profile of SAML, see section 2.4.2 for details).

8. Upon receipt of the trusted ticket, the user can request a service from the foreign service provider showing the trusted ticket.

9. - 10. In order to deny or allow the service, the service provider (PEP) has to validate the ticket according to the policy of its own identifier domain. In addition to the validation of the authorization information in the ticket, the signatures and certificates in the ticket have to be validated. This means that, optionally, the privacy-CA has to be consulted to verify the AIK credentials.

5.2 Components

5.2.1 Privacy-CA

In addition to the regular tasks of creating AIK credentials, the privacy-CA is responsible for verifying whether the identity provider (the particular TPM) is authorized to issue trusted tickets. Depending on the public EK or the vendor certificate, an AIK credential containing the authorizing attribute is issued. Another task of the privacy-CA is the certificate revocation if the private EK or AIK gets compromised. Moreover, revocation is necessary if the assertion changes, is no longer valid or due to the loss of the privacy-CA's private signing key. For the realization of the privacy-CA, either a publicly available product can be adapted or an individual implementation according to the specification of the TCG can be used. Currently, the Institute of Applied Information Processing and Communication (IAIK) of the Graz University of Technology is working on a reference implementation for the privacy-CA [23].

5.2.2 Ticket-issuing Web Server

The main tasks of the ticket-issuing web server are the authentication of the users and the provision of a mechanism to request tickets for foreign services. Furthermore, it is responsible for the forwarding of messages to and from the identity provider. During the authentication process, different authentication methods ranging from username and password to biometrics or X.509 certificates are supported. The application can be implemented either as a traditional web application or as a SOAP-based web service.

5.2.3 Identity Provider

In addition to the conventional identity management tasks, the identity provider creates the customized trusted tickets (SAML assertions containing status and authorization information). Standard identity management software such as Shibboleth [27] has to be extended to create these assertions. The identity provider can either be located on an individual machine

or on the same platform with the ticket-issuing web server. Regardless of the chosen version, identity provider and ticket-issuing web server have to be treated independently. The access-control policies stored in the policy database specify whether the user is authorized to request the desired service. During the specification of the policies it is important to formulate them as generic as possible so that they are suitable for multiple servers in foreign domains offering the same or similar services. In addition to the generic entries, several server-specific entries can occur. For the specification of the policies, certain description languages such as XACML are available depending on the selected identity provider.

Another task of the identity provider is the communication with the privacy-CA to create the necessary AIK credentials. For this purpose, the client application for the communication with the privacy-CA is used. On the side of the identity provider, the client application includes an additional attribute in the request message. This attribute indicates the request for an AIK credential which authorizes the identity provider to issue trusted tickets. Alternatively, it is possible to abandon the adaption of the request message for the AIK credential and to let the privacy-CA issue AIK credentials with the additional attribute by default.

5.2.4 User

Depending on whether the ticket-issuing web server and the ticket-receiving service provider are implemented as web applications or SOAP-based web services, either a browser is sufficient or a special application capable of handling the SOAP messages has to be implemented. Regardless of the chosen version, the user is responsible for the authentication at the ticket-issuing web server, for requesting the trusted tickets, for receiving and forwarding them in conjunction with the request for the service offered by the ticket-receiving service provider.

5.2.5 Ticket-receiving Service Provider

The ticket-receiving service provider has to be capable of handling the service request containing the trusted ticket from the user. It can either be implemented as a regular web application or a SOAP-based web service. In addition to the part of the application that is responsible for offering the service, a module is required for the validation of the trusted ticket. This task consists of validation of the certificates, the status information and the authorization and authentication information contained in the SAML assertion, according to the policy specified in the identifier domain of the ticket-receiving service provider.

5.3 Protocol Messages

5.3.1 AIK Credential Request

The demand for the authorization to issue trusted tickets is included in the identity provider's request for AIK credentials at the privacy-CA. The regular ASN.1 format for this request message is specified in [57, p. 279]. Figure 5.3 illustrates the adapted format of the *Tcpa Identity Request* message.

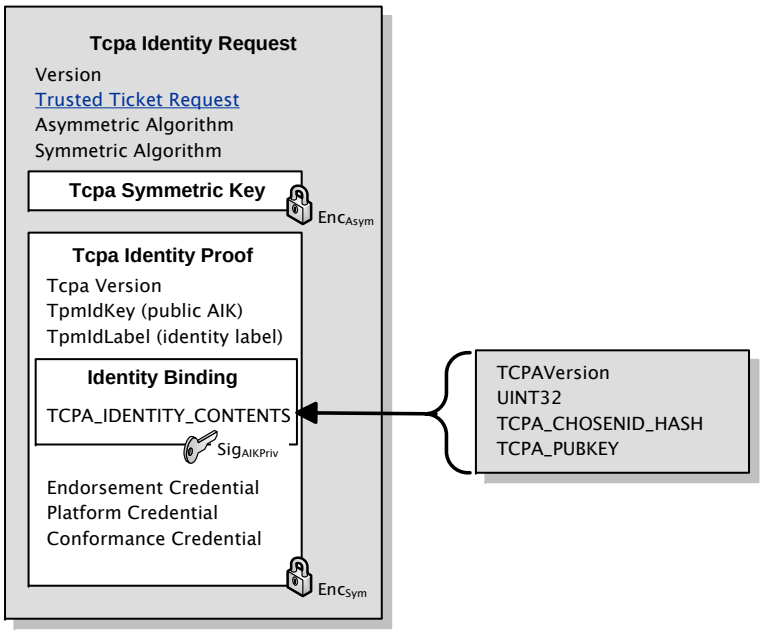

Figure 5.3: AIK credential request indicating the demand for the authorization to issue trusted tickets.

The message starts with some basic information such as version and used asymmetric and symmetric algorithms. In this basic information, the request for the authorization to issue trusted tickets is included as a boolean value. In order to protect the message content, a symmetric key is used, which is asymmetrically encrypted with the public key of the privacy-CA and stored in the *Tcpa Symmetric Key* field. This symmetric key is used to encrypt the content of the *Tcpa Identity Proof* which consists of version, public AIK, the chosen identity label for the AIK and the endorsement, platform and conformance credentials. Additionally, the identity binding is included in the *Tcpa Identity Proof*. This identity binding is calculated as the signature value using the private AIK over the data TCPA_IDENTITY_CONTENTS of the TPM which consists of TCPA version, the ordinal of the MAKE_IDENTITY command of the TPM (UINT32), the hash of the concatenation of the chosen

identity label, the public key of the privacy-CA (TCPA_CHOSENID_HASH) and the public AIK (TCPA_PUBKEY).

For the creation of this adapted request message, the identity provider's client application for the creation of AIK credentials is responsible. It has to assure that the boolean flag—indicating the request for an adapted AIK credential—is included in the *Tcpa Identity Request* whenever it is required. The reason for the integration of the flag in the unprotected part of the request is that it logically does not fit the content of the *Tcpa Identity Proof*. Furthermore, the fact that the identity provider requests the authorization to issue trusted tickets is not sensitive. If an attacker alters the request message and removes or adds the request flag, no security problems arise for the identity provider because the attacker cannot benefit from this attack.

5.3.2 AIK Credential

In the AIK credential issued by the privacy-CA, the authorization attribute for the identity provider, which allows the issuance of trusted tickets, has to be included. The TCG specifies the format for the endorsement, platform and attestation identity credentials in [60]. In all of the three credential types, the following fields must be included [60, p. 11]:

- Credential type label: The field specifies the type of the credential, which allows the issuer to sign the credential with a key not exclusively reserved for signing this specific type of credential.
- Issuer: Identifier of the entity that signed and issued the credential.
- TCG specification version: Identifies the version of the specifications of the TPM or the platform's TBB.
- Signature value: The issuer's signature over the other fields in the credential.

In addition to these mandatory fields, which have to be included in all the different credential types, the following information fields have to be additionally included in AIK credentials: Identity (AIK) public key, TPM model/manufacturer/version and Platform model/manufacturer/type. Moreover, the credential can contain further information as specified by the requester and issuer policies. Example elements that might be used in an AIK credential are identity label or validity period. A full list can be found in [60, p. 19].

The implementation of the AIK credentials is achieved by reusing the message format of X.509 certificates and X.509 attribute certificates as specified in RFC 3280 [18] and RFC 3281 [12]. Table 5.1 gives an overview of the necessary fields in the AIK certificate and their usage [60, pp. 33ff.]. The field for the alternative names is used for the specification of the TPM and platform manufacturer, model and version numbers as well as for the TPM

Field Name	Description	Stat
Version	Certificate syntax version number	1
Serial Number	Unique relative to the issuer	1
Signature Algorithm	Used by the issuer to sign this certificate	1
Issuer	Distinguished name of the privacy-CA	1
Validity	Time interval during which the certificate is valid	1
Subject	Distinguished name of the certificate, must be empty	1
Public Key Info	Algorithm identifier for the public key	1
Certificate Policies	Policy terms under which the certificate was issued	2
Alternative Names	Name forms other than directory distinguished names	2
Basic Constraints	CA certificate indicator and path constraints	2
Subject Directory Attributes	Various device characteristics	2
Authority Key ID	Identifies the subject public key of the certificate issuer	3
Authority Info Access	Indicates how to access CA information	4
CRL Distribution	Indicates how to access CRL information	4
Key Usage	Indicates the intended use of the subject public key	5
Extended Key Usage	Indicates the intended use of the subject public key	5
Subject Key ID	Identifies the subject public key of the certificate	5
Subject Unique ID	Unique value when using a shared subject name	5
Issuer Unique ID	Unique value when using a shared issuer name	5

Table 5.1: AIK certificate fields. The value "Standard" in the status column means that the field is an inherent component of the standard certificate syntax and is not optional. [60, p. 34] **Key:** Status (Stat) (1) Standard, (2) Must, (3) Should, (4) May, (5) Should not

identity label originally provided by the TPM owner in the *Tcpa Identity Request*. In the subject directory attributes, the TPM and TCG platform specifications have to be included. Optionally, the supported algorithms implemented by the TPM as well as the TPM and TBB security assertions should be included in this field.

Even though the extended key usage field should not be used according to the TCG, this field will be used for the integration of the authorization information in the certificate. It has to be stated that according to the TCG, the certificate has to be rejected during the path validation if the certificate is marked as critical and contains an extended key usage attribute which is not understood [60, p. 37]. The extended key usage is defined by unique object

identifiers assigned in accordance with IANA or ITU-T recommendation X.660. According to the object identifier repository [1], the key purpose with the object identifier 1.3.6.1.5.5.7.3 is already defined for the usage scenarios one to 16. The next free number in the tree is 1.3.6.1.5.5.7.3.17 which has to be newly registered for the issuing of trusted tickets (trustedTicketIssuing).

5.3.3 Trusted Ticket

The trusted ticket issued by the identity provider is specified as a SAML assertion including authentication, attribute and authorization decision statements (see figure 5.4). The authentication statement specifies details about the authentication process at the ticket-issuing web server and the authorization decision statement contains information about the actions, the user is allowed to perform at a certain service provider.

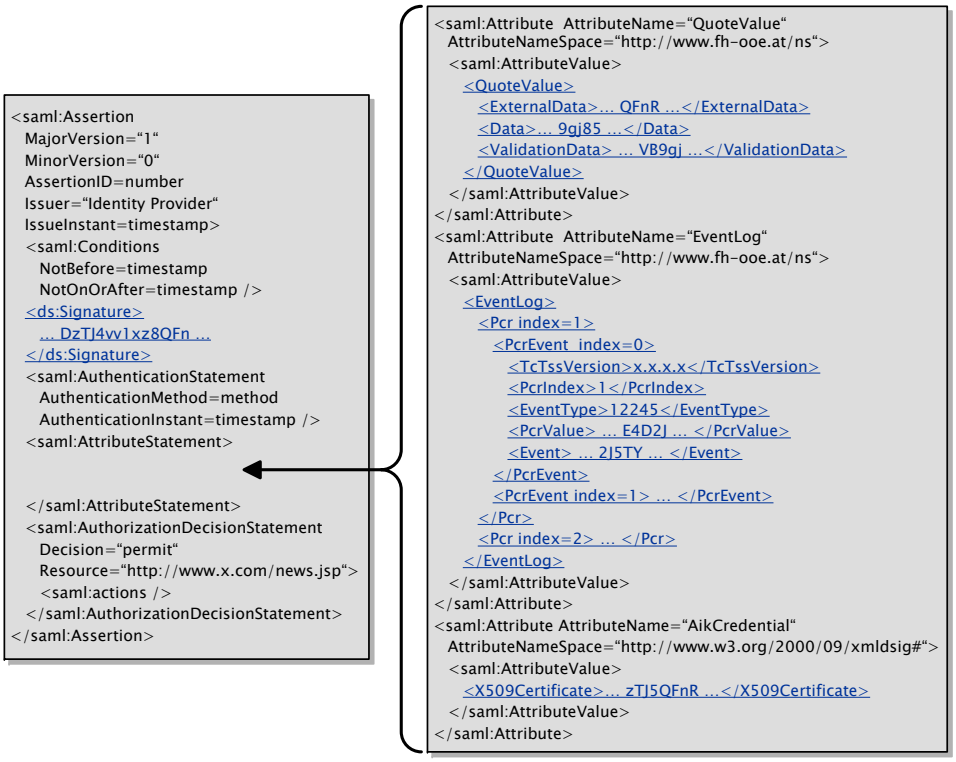

Figure 5.4: Modified SAML assertion used as a trusted ticket.

The necessary information which allows a ticket-receiving service provider to validate the ticket (authorization of the identity provider to issue trusted tickets and status information about the time of the ticket issuance) has to

be included in the attribute statement. A signature with the private AIK over a hash value of the PCRs is stored in the QuoteValue field which internally consists of three different data blocks. In order to verify the PCR hash value included in the quoted data field and the current configuration of the platform, the integrity measurement log, which includes the measurement values as well as the PCR values (see section 4.3.1), has to be transmitted to the ticket-receiving service provider in the EventLog field. The event log consists of multiple event values (PcrEvent) for each PCR (Pcr). For each event value, several information fields, such as TcTss version, PCR index, event type, PCR value and event value, have to be specified. Finally, the AikCredential field is used to store the AIK credential, which is required to validate the signature in the quoted value. For the encoding of the data in the XML elements, the base 64 encoding scheme according to RFC 4648 [29] is used.

In order to protect the integrity of the whole assertion, a digital signature according to the XML Digital Signature Standard [2] is required. A signing key created by the TPM and asserted by the private AIK is used to calculate the signature over the content of the assertion. The digital signature is stored in the field ds:Signature, which is already defined in the SAML specification, together with the public signing key.

5.4 TPM Functionality

5.4.1 Basic Operations

One of the basic operations, which is not within the scope of the application design, is the activation of the TPM and the process of taking ownership of the TPM. During this process, the user authorization data is specified in order to make the TPM ready for its usage. For the activation, the trusted software stack (TSS, see section 3.4.1 for details) offers the following function [58, p. 110]:

- Take ownership of the TPM (Tspi_TPM_TakeOwnership): The owner of the TPM specifies a secret which allows the usage of certain functions offered by the TPM.

Additionally, a context object has to be generated in each application which represents a connection to the TSS Core Service running on a local or remote TCG system. This context can further on be used to access the TPM functionality. The necessary functions to manage this connection are [58, pp. 74ff.]:

- Create context (Tspi_Context_Create): Creates a context object.
- Close context (Tspi_Context_Close): Destroys the context.

- Connect (Tspi_Context_Connect): Establishes a connection to a local or remote TSS system.

5.4.2 Extraction of Status Information

For the extraction of the status information, the application can use the functionality offered by the trusted software stack. As specified in [58, pp. 83,181], the following objects and commands are required:

- Create PCR composite object (Tspi_Context_CreateObject): This object offers a convenient way to manage the PCR values of the TPM. Operations such as select, read or write the PCRs are available.
- Select PCR indices (Tspi_PcrComposite_SelectPcrIndex): With the help of this function, several PCR indices in a previously created PCR composite object can be selected. The selected PCRs will later on be used in the calculation of the signature over the PCR values (quote process).

In order to extract the current event log, the TSS provides the following functions [58, pp. 135f.]:

- Get events (Tspi_TPM_GetEvents): This method returns a specific number of PCR events for a particular PCR index.
- Get event log (Tspi_TPM_GetEventLog): With the help of this function, the whole event log can be accessed at once.

5.4.3 Key Generation

The applications require several cryptographic keys which have to be created by the application of the identity provider. These are namely the AIK and the signing key (2048 bit RSA keys), which is required to assure the integrity of the trusted ticket. The trusted software stack offers the following interfaces to create and certify AIKs [58, pp. 83,87,111,113]:

- Create RSA key object (Tspi_Context_CreateObject): With the help of this method, an RSA key object specifying the initialization flags (key size, usage) can be created.
- Get TPM object (Tspi_Context_GetTPMObject): This method returns the TPM object belonging to a certain context. For each context, only one instance of this object exists.
- Create AIK and identity request (Tspi_TPM_CollateIdentityRequest): This method creates an identity key pair, binds it to a specified identifier label and returns a certificate request package, which can be sent

to the privacy-CA to acquire the AIK credential. As an additional parameter, a symmetric session key has to be specified, which is used to encrypt the data structure sent to the privacy-CA.

- Activate AIK (Tspi_TPM_ActivateIdentity): The response of the privacy-CA is processed with the help of this function. It verifies that the credential in the response message belongs to the originally created AIK and returns the decrypted credential created by the privacy-CA. After the activation and the registration of the AIK in the persistent storage of the TPM, the AIK and the corresponding credential can be used in any arbitrary application according to the usage policy.

Moreover, a 2048 bit non-migratable RSA key pair has to be created for the calculation of the necessary signatures over the trusted tickets. To attest that the signing key belongs to a valid TPM, the public key of the signing key has to be certified with a private AIK. It is important that in a trusted ticket, the same AIK is used to quote the status information and to certify the signing key because otherwise two AIKs belonging to one and the same TPM can be linked by external entities. For the implementation, the following TSS functions are required [58, pp. 83,90,153,154]:

- Create RSA key object (Tspi_Context_CreateObject): With the help of this method, an RSA key object specifying the initialization flags (key size, usage) can be created.
- Create key pair (Tspi_Key_CreateKey): This method creates a key pair within the TPM and wraps it with the storage key specified as a parameter. This means that it is encrypted with the storage key and deposited in the key hierarchy of the external storage.
- Certify key (Tspi_Key_CertifyKey): By using this method, the public key of a key pair can be signed with the private key of another—the attesting—key pair.
- Register key (Tspi_Context_RegisterKey): Every key which should be internally managed by the key management services of the TSS has to be registered in the persistent storage database of the system or the user. To register the signing key in the TSS persistent storage databases, this function has to be used. Once registered in a database, a key will be referenced by its Universal Unique Identifier (UUID) (TSS_UUID). As long as the keys are not unregistered, they are persistently stored in the database even after a platform reset.

5.4.4 Cryptographic Operations

To quote the status information in the PCRs of the TPM, the TSS offers the following functionality [58, pp. 131,137]:

- Quote (Tspi_TPM_Quote): This method quotes a TPM system, which means that it calculates a signature with the private AIK over the PCR values of the system. AIK and PCR values are specified as parameters. The indices of the PCRs to be quoted are specified in the PCR composite object (Tspi_PcrComposite_SelectPcrIndex). It is important to note that during the quote process, a composite hash value is calculated based on the selected PCR indices in the PCR composite object. The original values of the PCRs are not included in the quoted value. Additionally to the already mentioned parameters, an object of the type TSS_VALIDATION has to be specified. On the one hand this validation object acts as an input for a random number and on the other hand, the return value of the quote process is stored in this object after the command has been successfully processed.

- Generate random number (Tspi_TPM_GetRandom): With the help of this method, a random number of arbitrary length can be created by using the TPM's random number generator.

Additionally, the mechanisms to calculate digital signatures over arbitrary content with a key managed by the TPM are required to sign the contents of the trusted ticket. The necessary functions are [58, pp. 83,92,162,167]:

- Get key by UUID (Tspi_Context_GetKeyByUUID): The persistent storage is searched for a registered key with the provided UUID. Upon success, a key object is initialized according to the data.

- Create hash object (Tspi_Context_CreateObject): An object to manage the hash calculations has to be created.

- Calculate the hash value (Tspi_Hash_UpdateHashValue): By using this method, the hash value of the previously created hash object is updated. The supported algorithm is SHA-1.

- Sign the hash value (Tspi_Hash_Sign): The current hash value is signed with the signing key specified as a parameter. The signature value can be accessed via the return value of the method.

Chapter 6

Implementation

This chapter summarizes all the implementation-related aspects. After the presentation of the implementation scope, the application architecture and the infrastructure used (base system, Trusted Platform Module, additional libraries etc.) are described. Furthermore, the two main components (key generator and identity provider) are discussed with regard to their tasks and design. In addition, special implementation issues for the two components are mentioned.

6.1 Implementation Scope

During the implementation, the focus is placed on the topics related to the establishment of trust relationships. The following aspects of the realization concept—as defined in chapter 5—are part of the reference implementation:

- Request, creation and activation of AIKs and AIK credentials containing the extended key usage attribute authorizing the certificate owner to issue trusted tickets.
- Creation of a signing key certified by the previously created and certified AIK.
- Creation of trusted tickets as a signed SAML assertion containing all the necessary status information (QuoteValue, EventLog, AikCredential) of the identity provider.
- Additional functionality such as the initialization of the privacy-CA and the creation of event log entries because of the missing operating system support.

Currently not implemented are those parts of the concept which act in their conventional roles only. These are namely: the identity provider itself, the ticket-issuing web server and the ticket-receiving service provider. An

application running on the client to request and process the trusted tickets is also not part of the current implementation scope.

6.2 Architecture

The main tasks of the implementation can be divided in five major parts. For each of these parts, the functionality is clustered in an individual application. The following components have to be considered:

- **InitPca** As a prerequisite for the other applications, a key hierarchy consisting of key pairs and certificates for the root-CA as well as for the CAs used to create EK and PE certificates has to be established. In addition, a key pair for the privacy-CA and the EK and PE certificates are created if the TPM does not possess these two credential types.
- **AikKeyGenerator** For the generation of the AIKs and the corresponding credentials, the functionality of the privacy-CA and the identity provider with its TPM are combined in one application. This means that no communication protocol has to be implemented between the two parties. The main task of the AikKeyGenerator is the creation of the AIKs and the credentials according to step one to six of the protocol outlined in figure 5.1.
- **SignatureKeyGenerator** This application is responsible for the creation of a certified signing key. After the creation of an RSA key pair, the public key is signed with the previously created private AIK. By means of this certification, an external party can be assured that the signing key has been created and will be protected by a trustworthy TPM. The necessary task is described in step seven of figure 5.1.
- **ProduceEventValues** As the current operating system does not offer the support for the creation of an event log, this application simulates this functionality. An arbitrary number of events is added to different PCRs of the TPM.
- **IdentityProvider** The identity provider can be used for the creation of a trusted ticket. Internally, it uses different other classes that help to create the SAML assertion and integrate the status information in the attribute statement. Currently, the authentication and authorization decision statements are not set because normally they are provided by the regular identity provider application. Due to integrity issues, an XML signature over the contents of the assertion is calculated with the TPM's certified signing key. Mainly, the activities cover step six of the protocol illustrated in figure 5.2.

6.3 Infrastructure

6.3.1 Base System

The base system used during the implementation consisted of an Ubuntu 6.10 installation with a Linux kernel version 2.6.17. As a prerequisite for the other components, additional software had to be installed. Details can be found in the particular installation documentation of the components described in the following sections. As the later on used libraries for controlling the TPM are implemented in Java, the Java environment was required which means that the Java SE 6 development kit was installed on the system. Eclipse version 3.2.2 was used as the development application.

6.3.2 Trusted Platform Module

During the implementation, a software TPM emulator as well as a hardware TPM were used. The used TPM emulator [55] was developed by Mario Strasser at the Swiss Federal Institute of Technology (ETH) Zurich [55]. For the compilation and installation process, the build environment for external kernel modules of the 2.6.X Linux kernels was used. Since Version 0.5, the emulator consists of:

- **tpmd**: a user-space daemon which implements the actual TPM emulator and can be accessed by means of Unix domain sockets.
- **tpmd_dev**: the kernel module that provides the device /dev/tpm and forwards the commands to the tpmd.
- **tddl**: a device driver library for the TPM emulator.
- **tpm_dev**: the (obsolete) kernel-space TPM emulator.

As a hardware solution, Infineon's TCG 1.2 compliant TPM (SLB 9635 TT 1.2) installed in an HP Compaq nc8430 notebook was available. In order to be able to use the TPM in the operating system and any application, the TPM functionality had to be activated in the BIOS. Should a reset of the TPM be necessary, it can be conducted in the BIOS as well. The device drivers for the TPM were already part of the Linux kernel modules (tpm_infineon).

6.3.3 Trusted Software Stack

IBM's open-source Trusted Software Stack (TSS) Trousers (Version 0.2.9) [25] was used with the TPM emulator as well as the hardware TPM. Trousers aims to be compliant with the TCG's 1.1b and 1.2 TSS specifications [25]. However, the version 1.2 compliance is not yet fully integrated. Therefore,

the Institute for Applied Information Processing and Communication (IAIK) at Graz University of Technology offers a patch [23] which intends to make Trousers run on Infineon 1.2 TPMs. This is because the TPM specification 1.2 deprecates numerous commands which therefore are no longer implemented in Infineon 1.2 TPMs. Additionally, the patch contains a workaround to also run properly with the TPM emulator from the ETH Zurich, which claims to be a 1.2 TPM but does not support all the required functions.

The TSS consists of a TCS daemon and a TSP shared library (see section 3.4.1 for more details). In addition, two different kinds of persistent storage files are managed by the TSS. The persistent storage files can be perceived as databases for keys indexed by a UUID. The application's TSP library is responsible for the maintenance of the *user persistent storage*. However, the *system persistent storage* is controlled by the TCS and as a consequence stays valid after TCS restarts and system resets until an application requests the removal of certain entries. By default, the TCS daemon is not reachable over the Internet; however, it is possible to configure the daemon in a way that several functions can be accessed from foreign machines.

6.3.4 Java Libraries

In order to implement the necessary TPM functionality in Java, several libraries available at [22] and [23] have to be installed on the system:

- **jTSS Wrapper**: As the TSS was typically developed in pure C programming language, it can therefore not directly be used from other languages such as Java. That is why the IAIK/OpenTC jTSS Wrapper provides language bindings for Java which offer an object-oriented Java API to the Trusted Service Provider Interface layer of the TSS. The jTSS Wrapper is developed and maintained at the Institute for Applied Information Processing and Communication at Graz University of Technology. During the reference implementation, the partially pre-built version 0.2.5 of the jTSS Wrapper was used.

- **jTpmTools**: The IAIK/OpenTC jTpmTools are a set of command line tools for basic interaction with the TPM and the TSS. Mainly, the parts of the jTpmTools version 0.2 which define several constants for the key and certificate generation were used in the implementation.

- **TcCert**: The version 0.2.1a of this library was used to create and use the certificates specified by the TCG.

- **JCE**: In order to supplement the security functionality of standard Java, the version 3.142 of the IAIK provider for the Java Cryptography Extension (IAIK-JCE) was used. The JCE provides a set of APIs and implementations of cryptographic functions such as hash calculation,

message authentication codes, different encryption algorithms as well as key and certificate management.

- **Sample Applications**: During the implementation, the sample applications and test cases provided by the IAIK were used, which can be redistributed and modified under the terms of the GNU General Public License (version 2).

For the integration of the XML functionality—in particular the creation of the trusted ticket—the following libraries are necessary:

- **OpenSAML**: The implementation of the SAML 1.1 and 1.0 specifications is mainly used internally in the Shibboleth project. However, a library for external usage in Java applications, which provides the functionality to create SAML assertions, is available too. In the implementation, version 1.1 of OpenSAML [26] was used.
- **XMLSec**: For the creation of a signed SAML assertion, version 1.4 of the Java library for XML Security [56] had to be used [56]. The library supports the XML-Signature as well as the XML Encryption Syntax and Processing in accordance to the W3C Recommendations [2] and [21].

6.4 Key Generator

6.4.1 Tasks

The key generator applications are responsible for the tasks related to key generation and management. This includes the generation of AIKs and the corresponding credentials as described in section 5.3.2. In addition, signing keys certified with the previously generated AIK have to be created and the keys have to be registered in the persistent system storage of the TPM. The participating parties and the protocol messages are described in figure 5.1.

6.4.2 Design

The package `at.hagenberg.fh.trusted.keys` contains the classes for the implementation of the key generator:

- `Defines:` In this class, several constants used by all the other applications are defined. These are for example secret passwords, modes, file system paths, UUIDs etc.
- `InitPca:` The initialization of the root CA, the creation of the necessary credentials and the keys for the privacy-CA are part of this class. All the created keys and certificates are stored in external key stores in the file system of the platform.

- `AikKeyGenerator:` This class creates the AIKs and the AIK credentials for the TPM. It combines the tasks of the identity provider with the TPM and the privacy-CA. Normally, these two parties have to be separated into two applications on different systems with an appropriate communication protocol in between. Nevertheless, for this demonstration it is sufficient to combine the two tasks in one application paying particular attention to adequate information separation. The generated AIK is registered in the persistent system storage of the TPM and the related AIK credential is exported to the file system of the platform.

- `SignatureKeyGenerator:` This class implements the functionality to generate RSA signature keys which are certified by a previously generated AIK of the TPM. The certified signing key is stored in the persistent system storage of the TPM and the corresponding certificate is stored externally in the platform's file system.

6.4.3 Implementation

The class for the initialization of the privacy-CA mainly uses the functions provided by the TcCert library and the constants defined in the TpmUtils. For the specification of the attributes used for the certificates, different initialization files (ca.ini, ek.ini and pe.ini) have to be used. The paths to the initialization and output folder are specified in the `Defines` class.

In order to be able to send commands to the TPM in the Java application, a `Context` object representing a connection to the TSS Core Service—running on the local or remote platform—has to be created. Mainly, this object is responsible for the resource and memory management, the creation of working objects, the establishment of a default policy for working or TPM objects and the access to the persistent storage databases. `Policy` objects are used to store authorization data for working objects such as keys or encrypted data. If the policy object stores authorization data for the TPM object, it represents the TPM owner.

Before any other commands can be executed, the ownership of the TPM has to be taken. The class `AikKeyGenerator` offers a method to take the ownership of the TPM with the help of the jTSS wrapper during which the owner authorization secret is set and the storage root key is created.

The first step during the creation of AIK credentials is the TPM's generation of the CollateIdentityRequest. In listing 6.1, the code excerpt for the creation of this request is shown. After the creation of the AIK, the usage and migration policies for the key have to be defined and assigned to the newly generated key object in order to be able to create the finished identity request. The information in the identity request is encrypted using the public key of the privacy-CA.

```
1 // create the AIK
2 aikKey_ = context_.createRsaKeyObject(TcTssDefines.
      TSS_KEY_TYPE_IDENTITY | TcTssDefines.TSS_KEY_SIZE_2048 |
      TcTssDefines.TSS_KEY_AUTHORIZATION | TcTssDefines.
      TSS_KEYUSAGE_IDENTITY | TcTssDefines.TSS_KEY_NON_VOLATILE |
      TcTssDefines.TSS_KEY_NOT_MIGRATABLE);
3
4 // define usage policy for the AIK
5 TcIPolicy aikUsagePolicy = context_.createPolicyObject(
      TcTssDefines.TSS_POLICY_USAGE);
6 aikUsagePolicy.setSecret(Defines.AIK_POLICY_SECRET_MODE,
      Defines.AIK_SECRET);
7 aikUsagePolicy.assignToObject(aikKey_);
8
9 // define migration policy for the AIK
10 [...]
11
12 // create CollateIdentityRequest
13 identityRequest = tpm.collateIdentityRequest(srk_, pcaPubKey,
      client_GetIdLabel(), aikKey_, TcTssDefines.TSS_ALG_AES);
```

Listing 6.1: *AikKeyGenerator.java:* Creation of the CollateIdentityRequest.

Upon receipt of the identity request, the privacy-CA decrypts and validates the request. After successful processing, the AIK credential is created. In this implementation, the privacy-CA adds the extended key usage attribute, which authorizes the owner of the AIK credential to issue trusted tickets, by default. As the TcCert utilities do not allow the integration of an additional key usage attribute in the appropriate initialization file (aik.ini) taken for the creation of the certificate, the attribute has to be manually added (see listing 6.2). The response message is sent back to the platform with the TPM, which has to activate the AIK credential as the next step. During the activation of the AIK credential, the message is decrypted and the TPM can register the key in the persistent system storage and export the credential to the file system.

During the generation of the certified signing key, a new RSA key pair is generated. It is important to state that the key pair has to be generated with the attributes *signing*, *authorization* and *not migratable*. Especially the last two attributes are required because an identity key is used to certify the signing key in the application. As defined in the TCG specification, an identity key can only be used to certify non-migratable keys which require authorization. In addition to the public and private part of the signing key, an X509 certificate containing the AIK's signature value over the public signing key has to be manually generated.

```
1 String oidStr = "1.3.6.1.5.5.7.3.17";
2 String name = "trustedTicketIssuing";
3 String shortName = "tTI";
4 ObjectID oid = null;
5 if (!ObjectID.hasRegisteredName(oidStr)) {
6   oid = new ObjectID(oidStr, name, shortName);
7 }
8
9 ExtendedKeyUsage keyUsage = new ExtendedKeyUsage();
10 // add the purpose id to the KeyUsage attribute
11 keyUsage.addKeyPurposeID(oid);
12 // add the extension to the certificate
13 tempCert.addExtension(keyUsage);
14 // sign the certificate with the private key of the pca
15 tempCert.sign(AlgorithmID.sha1WithRSAEncryption, pcaKey_);
```

Listing 6.2: *AikKeyGenerator.java:* Integration of the extended key usage attribute in the AIK credential.

6.5 Identity Provider

6.5.1 Tasks

In this reference implementation, the main task of the identity provider is the generation of the SAML assertion. The attribute statement integrated in the assertion (see section 5.3.3) contains the platform's status information. Due to integrity reasons, the whole content of the SAML assertion has to be digitally signed using the certified signing key managed by the TPM. Figure 5.2 illustrates the participating parties and the necessary protocol messages.

6.5.2 Design

The package `at.hagenberg.fh.trusted.identityprovider` contains the main classes for the implementation of the identity provider:

- `ProduceEventValues`: As the current operating system does not support the creation of an event log consisting of measurement values of the running programs and configurations used, this application simulates the creation of arbitrary events belonging to different PCRs in the event log.
- `IdentityProvider`: This class is the main class for the identity provider, which calls all the necessary sub functions.

All the functionality related to the creation of the SAML assertion is centralized in the package `at.hagenberg.fh.trusted.ticket`:

- `TrustedTicket`: This class is used to create the trusted ticket. Currently, the SAML assertion only contains the attribute statement because in the final version, authentication and authorization decision statements are directly added by the identity provider.
- `XMLQuoteValue`: In order to create the SAML attribute specifying the quote values of a TPM, this class has to be used.
- `XMLEventLog`: The functionality to create the XML data for the event log attribute is provided by this class.
- `XMLAikCredential`: The attribute specifying the platform's AIK credential used to quote the PCRs has to be created with this class.

In the package `at.hagenberg.fh.trusted.ticket.tpm`, all the classes for the communication with the TPM and for the processing of TPM information are stored:

- `TpmUtils`: This class is used for the extraction of the status information from the TPM. Furthermore, all the additional information required for the integration in the SAML attribute statement is extracted. In addition, utilities for signing arbitrary data with a signing key managed by the TPM are provided.
- `QuoteValue`: This class is used to extract all the data necessary to specify the contents of the `QuoteValue` attribute from an object of the class `TpmUtils`.
- `EventLog`: In order to access the event log data, this class has to be used. It extracts the necessary information from the `TpmUtils` object and allows external access to the data.
- `PcrEvent`: This class is used to manage all the information required for an XML element `PcrEvent`.
- `Pcr`: This class provides functions to manage all the information in an XML `Pcr` element.
- `AikCredential`: For the management of the AIK credential specified in an object of the class `TpmUtils`, this class has to be used.
- `KeyValue`: This class is used to store key and value pairs in order to allow easy access to the particular fields.

For the realization of the XML signature over the contents of the SAML assertion, a class in the package `org.opensaml` which is part of the OpenSAML library, has to be adapted:

- `SAMLSignedObject`: This class is the abstract base class for all SAML objects that can be signed. These objects are SAML assertions, SAML requests and response messages.

6.5.3 Implementation

For the simulation of the event log creation, the functionality of the TPM to extend PCR values is used. This means that the new PCR value is mathematically concatenated with the old one and a new entry which contains more specific information about the event is added to the event log. The event log entry includes data such as version, type, PCR index, PCR value and the descriptive event data itself.

The SAML assertion used as a trusted ticket is built with the help of the OpenSAML library. In order to create the XML structure for the assertion, all the predefined classes and methods of the library have to be used. At the beginning of the creation, the current status information has to be extracted from the TPM. This is achieved by the creation of a `TpmUtils` object which stores all the necessary information. During the creation of the SAML attribute statement, this object is used to fill in the XML data in the particular attributes. An example for the final SAML assertion can be found in appendix A.

In general, the class `TpmUtils` offers all the TPM functionality needed during the creation of a trusted ticket. The first of these tasks is the process of quoting the TPM. After the specification of those PCRs which have to be quoted and the loading of the AIK into the TPM, the actual quote process can be started. In a `TcIPcrComposite` object, all the indices of the PCRs to be quoted are marked. Next, the random nonce used during the signature generation is set in the `TcTssValidation` object which stores the signed PCR value after the signature creation with the signing key. It has to be stated, that during the quote process, a composite hash value is calculated based on the selected PCR indices in the PCR composite object. In addition to the quote value itself, the event log containing more detailed information about the PCR calculation has to be extracted for the specified PCRs. To complete the information needed for the integration in the SAML attribute statement, the AIK certificate has to be read from the file system by the `TpmUtils`.

Another major task of the `TpmUtils` is the calculation of digital signatures over arbitrary content with a signing key managed internally by the TPM. Listing 6.3 shows the method responsible for the signing process. First, the signing key has to be read from the persistent storage and unwrapped (decrypted with the storage key). Next, the policy object specifying the authorization information for the key object is created and assigned with the key. After these steps, the hash value object can be created and initialized with the digest value to be signed. Finally, the hash value is signed using

```
1 public byte[] signWithSigningKey(byte[] digest) throws
      TcException {
2   // Read and load (decrypt with SRK) the signing key
3   TcIRsaKey key = readKey(Defines.UUID_SIGNING_KEY);
4   key.loadKey(srk_);
5
6   // Create policy object for the signing key
7   [...]
8
9   // Create hash value
10  TcIHash hash = context_.createHashObject(TcTssDefines.
        TSS_HASH_SHA1);
11  TcBlobData digestBlob = TcTssStructFactory.newBlobData().
        initByteArray(digest);
12  // set the hash value to the digest value to be signed
13  hash.setHashValue(digestBlob);
14
15  // sign the hash value
16  TcBlobData signatureBlob = hash.sign(key);
17  byte[] signature = signatureBlob.asByteArray();
18  context_.closeObject(hash);
19  return signature;
20 }
```

Listing 6.3: *TpmUtils.java:* Signing arbitrary data with a signing key managed by the TPM.

the previously loaded signing key.

In order to create a signed SAML assertion using the signing capabilities of the TPM, a new method has to be inserted in the `SAMLSignedObject` class. The method `signWithTpm()` is based on the standard signing method of the class but instead of the regular Java Crypto Providers it uses the TPM to sign the digest value. After the specification of hash and signature algorithms (SHA1 and RSA), an empty `XMLSignature` object is created. Next, those elements of the SAML assertion which have to be signed are specified, and the X509 certificate used to validate the signature is included in the key information field. As the next step, the `SignedInfo` element has to be extracted from the XML signature (see listing 6.4). After the calculation of the digest, the value can be extracted from the element and signed with the help of the `TpmUtils`. After the calculation of the signature, the value has to be encoded according to the base64 encoding rules and inserted in the appropriate place of the XML signature structure.

```
1 // Get the signed info and generate digest
2 SignedInfo info = sig.getSignedInfo();
3 info.generateDigestValues();
4
5 // Extract digest value
6 Reference ref = info.item(0);
7 byte[] digest = ref.getDigestValue();
8 byte[] signatureValue = utils.signWithSigningKey(digest);
9
10 // Insert signature value in the XML signature
11 Document doc = sig.getDocument();
12 Text signatureText = doc.createTextNode(Base64.encode(
       signatureValue));
13 Element element = sig.getElement();
14 NodeList nodes = element.getChildNodes();
15
16 for (int i = 0; i < nodes.getLength(); i++) {
17   Node item = nodes.item(i);
18   if (item.getNodeName().equals("ds:SignatureValue")) {
19     item.appendChild(signatureText);
20   }
21 }
```

Listing 6.4: *SAMLSignedObject.java:* Signing XML data using the capabilities offered by the TPM.

Chapter 7

Analysis

During the implementation, the previously set objectives could be reached to a full extent. The creation of the keys and certificates as well as the trusted ticket containing the status information was successful. However, there are several issues which have to be kept in mind with regard to the realization concept. In the following, these critical aspects are discussed.

7.1 Integration of Status Information

According to the OASIS standard [35, p. 21], the attribute statement is usually used by the SAML authority to specify additional information about the subject, which is in our case the client. However, in the realization concept outlined in this thesis, the attribute statement is used to transport the status information of the SAML authority. Even though this usage scenario does not totally conform to the initial specification, the ticket-receiving service provider knows that the particular attributes in the statement do not describe the subject but the SAML authority itself. Therefore, it is no problem that the status information is inserted in the attribute statement.

However, during the implementation, several other options for the integration of the status information could be enumerated:

- First of all, the binding of the status information to the issuer would be desirable in the assertion. Unfortunately this is not possible as the issuer is only specified as an attribute but never as an independent XML element which allows the specification of additional information.
- Another option would be the XML signature and its optional `KeyInfo` element which allows the signer to specify additional key information required by the recipient to validate the signature. As described in the XML signature standard [2], this includes information such as keys, names, certificates or other public key management information. In addition to the predefined types, applications can extend those types

or replace them with their own data elements. The contents of the `KeyInfo` field can be integrated in the calculation of the XML signature by using the appropriate `Reference` element.

- As an alternative to the key information in the XML signature, the `SignatureProperties` element in the `SignedInfo` element could be used. As specified in the XML signature standard [2], the signature properties consist of additional information related to the generation of the signature(s). By using an appropriate `Reference` element, the signature properties can even be included in the signature generation.

7.2 Complex Architectures

From the client's perspective, the SAML authority acts as the interface (policy enforcement point) to the identity management infrastructure which issues the trusted ticket. However, it is possible that not only the SAML authority is involved in the issuing of the trusted ticket. Additional parties—such as the policy decision point or the authentication server—can be located on different machines behind the SAML authority (see figure 7.1) and act together in issuing one collective ticket. In such a scenario, it is not totally clear whose status information has to be integrated in the ticket as all the parties are involved in the decision process.

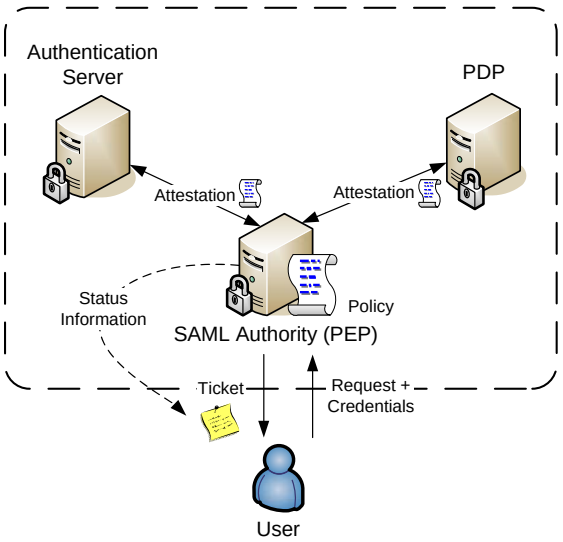

Figure 7.1: Integration of status information in a complex identity management architecture.

Several techniques exist for the appropriate integration of the status information:

- One of the possible solutions is the integration of each participant's status report in the assertion. This enables external parties to validate the status of each of the involved systems. However, the amount of data which has to be transported would be enormously high.
- As the policy enforcement point acts as the common interface to the identity management architecture, it is possible to make this system responsible for the system status of the policy decision point and the authentication server belonging to its domain. As illustrated in figure 7.1, the SAML authority issues a policy for its identity management domain which has to be kept by the other parties. Internally, the SAML authority has to continuously validate the status of the other parties by using for example the assertion mechanisms offered by the TCG infrastructure if each of the systems is equipped with a TPM. As a result, it is sufficient if only the SAML authority includes its status information and possibly the policy for the identity management domain in the assertion. As a consequence, a transitive trust relationship is established by the SAML authority, which allows an external party to trust not only the authority itself but also the infrastructure behind the policy enforcement point.

7.3 Event Log Size

An operating system implementing an integrity measurement process could easily lead to a very huge event log. As the whole event log has to be transmitted to the ticket-receiving service provider to enable a thorough validation, the overhead of the status information in the assertion can be relatively high in relation to the authentication and authorization information.

- A possible solution to reduce the amount of data can be the usage of adequate compression algorithms or other strategies to avoid redundancies, as quite often the events contain similar information (e.g. event type, version).
- Another solution to reduce the size of the transmitted SAML assertion is the usage of SAML artifacts. The attribute statement is not included in the original assertion but instead an artifact which enables the ticket-receiving service provider to query the SAML authority separately for the attribute statement. By using this technique, the status information would be directly transmitted from the SAML authority to the ticket-receiving service provider. If the digital signature would

be used to transport the status information instead of the attribute statement (as presented in section 7.1), this separate transmission of the status information with the help of SAML artifacts is not possible.

- Moreover, the usage of virtualization techniques could lead to a reduction of the event log size. This means that in the virtualized environment, only those applications and services are running which are relevant for the identity management process.

7.4 AIK Credentials

For the embedding of the authorization information in the AIK credential, the extended key usage attribute has been used. Even though, according to the TCG specification [60, p. 37], this attribute should not be used, the usage in this scenario does not lead to any problems. In particular this means that those applications which cannot interpret the attribute can either reject the certificate or ignore the modification according to their policy. The identity provider is responsible for the usage of the appropriate AIK and the corresponding credential depending on the application scenario. If a client requests a ticket for a foreign identifier domain, the identity provider has to use the AIK with the authorizing attribute in the credential. In this case, the ticket-receiving service provider has to be able to interpret the modified credential because otherwise it is not possible to establish the trust relationship with the foreign identifier domain.

7.5 Authorization of the Privacy-CA

During the ticket-receiving service provider's validation of the AIK credential, it is not enough to only verify whether the AIK credential contains the authorizing extended key usage attribute. As the key usage attribute could have been inserted by any arbitrary privacy-CA which might not have validated the EK public key or the issuer of the identity provider's TPM, it is also important not to forget the validation of the AIK credential issuer during this process.

Chapter 8

Conclusion

8.1 Results

In this thesis, a concept for the establishment of trusted infrastructures for identities has been successfully developed. These trusted infrastructures enable the establishment of trust relationships across multiple identifier domains in the context of identity management. Different approaches have been presented and evaluated with regard to the optimal integration of the trusted computing architecture. The most suitable solution has been chosen and taken as the basis for the development of a concrete realization concept. Even though some of the required adaptions of the trusted computing architecture do not fully conform to the TCG specification (e.g. additional tasks for the privacy-CA, extension of the AIK credential), they are acceptable as they are only minor changes which can be realized without changing the global TCG specification (e.g. TPM architecture, EK certificate structure).

Those parts of the concept, which represent the major functions for the establishment of trust relationships have been realized in a reference implementation. These are the creation of cryptographic keys and related certificates as well as the trusted ticket containing the status information. The second fundamental part is the validation of the trusted ticket at the ticket-receiving service provider. As the operating system support for the integrity measurement is currently not available, the validation of the trusted ticket has not been implemented thus far. In addition to these two major parts, a communication architecture (e.g. requesting, transmitting the tickets) is required. As this part only plays a minor role in the main context of trust establishment, it is not part of the current reference implementation.

An important fact is that the thesis introduces an academic view of the possibilities for the establishment of trust relationships. Several application fields which extend the main purpose of trust establishment were identified in section 4.7. However, none of these usage scenarios has been analyzed in more detail.

In general, the trusted computing technology is already quite well developed. The usage of the TPM in individual applications is possible without any problems. However, several parts—such as the implementation of the privacy-CA or the operating system support for integrity measurement—are not fully developed yet which means that there are still many future research topics in this area.

8.2 Outlook

As this thesis consists of fundamental research work in the area of the establishment of trust relationships in identity management architectures by using the trusted computing architecture, there are plenty of starting points for new research topics or the more detailed analysis of already discussed aspects.

One of these topics is the reference implementation. In the current version, the tasks of the privacy-CA and the identity provider are combined in a single application. The separation of these tasks in two separate applications with an appropriate communication protocol is a future implementation task. In addition, the integration of the already implemented ticket generator in a comprehensive architecture consisting of ticket-issuing web server, identity provider, client application and ticket-receiving service provider is an open issue.

Another important topic is the analysis of application scenarios in addition to the establishment of trust relationships. As already stated, some approaches have been mentioned in the thesis. However, they have to be analyzed in more detail in order to decide whether they are applicable.

Apart from these issues, there are several approaches which allow an extension of the current architecture. One of these possibilities is an adaption of the protocol which allows the service provider to present its current platform status to the client if the client requests a service. This would allow the user to distinguish whether the server and the offered service can be trusted. Another extension is the establishment of transitive trust relationships if the SAML authority, the identity provider and the authentication server are not situated on a single machine (as presented in section 7.2). In addition to these issues, the formulation of generic access-control policies focusing on multiple service providers offering similar services is another important research topic. Depending on the application scenario, message replay attacks have to be kept in mind as well. Especially if clients can use their tickets in multiple identifier domains, it is a challenging exercise to implement protection mechanisms against this kind of attack. But above all, the implementation of integrity measurement mechanisms in current operating systems is one of the most important issues. As long as these mechanisms are not fully available, trusted platforms will not be able to create expressive event logs and

as a consequence external parties will not be able to validate the status of a foreign platform.

This analysis shows that there are plenty of new research topics in the area of trusted computing and identity management. The thesis can be seen as one of the first steps in this research area which offer starting points for future projects. Particularly because the importance of trusted computing is steadily rising, it is definitely worth paying attention to its usage in the context of identity management in the course of future research projects.

Appendix A

Trusted Ticket

```
1 <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:xsd
    ="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3
    .org/2001/XMLSchema-instance" AssertionID="AssertionID"
    IssueInstant="2007-04-30T11:28:42.175Z" Issuer="Identity␣
    Provider" MajorVersion="1" MinorVersion="1">
2  <AttributeStatement>
3   <Subject>
4    <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
       format:unspecified" NameQualifier="www.fh-ooe.at/
       identityprovider">
5     identityProvider
6    </NameIdentifier>
7   </Subject>
8   <Attribute AttributeName="QuoteValue" AttributeNamespace="
      http://www.fh-ooe.at/ns">
9    <AttributeValue>
10    <QuoteValue>
11     <ExternalData>
12      sTsv003hopSSkmBy94u3vXPZGzI=
13     </ExternalData>
14     <Data>
15      AQEAAFFVT1THn8zjVdPF6bZlw2Ue3VPJMbxF97E7L9NN4aKUkp
          JgcveLt71z2Rsy
16     </Data>
17     <ValidationData>
18      cgDt5pjYKN+Z65RF8cdBo1Mc7tWftpFs0A4sB0fg0EQDufOUzEsCK+os
          HhlC9aTQWzcKydBNcRVBCHoMtlifCvBAKZdhufc82EO1bwup3NaJc
          NJlHEZjN+1LfmGB68ahqW752UDaxd2JAryo2Rf500tEL50XGuSxd4
          Lsw5S/QGFPyAnOWJao/8THfTjPkteuxGM3/mpNcvDiqrNqrPw3tEz
          MhirmbDi68UO3FeXgHr0I7VbU+K+qvuo6Z2tt7phvvhmMVzsQvqg4
          5PvRNegVQyzmIfBGF/3B8A03gBwFOjXBKmyCtVlr6hMxrBYnkbUeP
          oBWyOZO/sE6JB7yU0OS0A==
19     </ValidationData>
20    </QuoteValue>
21   </AttributeValue>
```

```
22  </Attribute>
23  <Attribute AttributeName="EventLog" AttributeNamespace="http:
       //www.fh-ooe.at/ns">
24   <AttributeValue>
25    <EventLog>
26     <Pcr index="0">
27      <PcrEvent index="0">
28       <TcTssVersion>Version: 1.1.0.0</TcTssVersion>
29       <PcrIndex>0</PcrIndex>
30       <EventType>7</EventType>
31       <PcrValue>
32        ZT5pt4iL9i1aHb9Svu+a+FsRbK0=
33       </PcrValue>
34       <Event>
35        VABoAGkAcwAgAGkAcwAgAGEAIAB0AGUAcwB0ACAAZQB2AGUAb
             gB0ACAAZgBvAHIAIABQAEMAUgAgADAALgA=
36       </Event>
37      </PcrEvent>
38      <PcrEvent index="1">
39       <TcTssVersion>Version: 1.1.0.0</TcTssVersion>
40       <PcrIndex>0</PcrIndex>
41       <EventType>7</EventType>
42       <PcrValue>
43        2qj/3B7kE3t90dZbqILEfMo/auA=
44       </PcrValue>
45       <Event>
46        VABoAGkAcwAgAGkAcwAgAGEAIAB0AGUAcwB0ACAAZQB2AGUAb
             gB0ACAAZgBvAHIAIABQAEMAUgAgADAALgA=
47       </Event>
48      </PcrEvent>
49     </Pcr>
50     <Pcr index="1">
51      <PcrEvent index="0">
52       <TcTssVersion>Version: 1.1.0.0</TcTssVersion>
53       <PcrIndex>1</PcrIndex>
54       <EventType>7</EventType>
55       <PcrValue>
56        3RkoUqfaQfNgMIlcMuXs9JKnqyg=
57       </PcrValue>
58       <Event>
59        VABoAGkAcwAgAGkAcwAgAGEAIAB0AGUAcwB0ACAAZQB2AGUAb
             gB0ACAAZgBvAHIAIABQAEMAUgAgADEALgA=
60       </Event>
61      </PcrEvent>
62     </Pcr>
63     <Pcr index="4">
64      <PcrEvent index="0">
65       <TcTssVersion>Version: 1.1.0.0</TcTssVersion>
66       <PcrIndex>4</PcrIndex>
67       <EventType>7</EventType>
68       <PcrValue>
69        Ft8ETQbZYdkg2wXDH/KoXGHt8Vw=
70       </PcrValue>
71       <Event>
```

```
72          VABoAGkAcwAgAGkAcwAgAGEAIAB0AGUAcwB0ACAAZQB2AGUAb
                gB0ACAAZgBvAHIAIABQAEMAUgAgADQALgA=
73        </Event>
74       </PcrEvent>
75      </Pcr>
76     </EventLog>
77    </AttributeValue>
78   </Attribute>
79   <Attribute AttributeName="AikCredential" AttributeNamespace="
        http://www.w3.org/2000/09/xmldsig#">
80    <AttributeValue>
81     <X509Certificate>
82      MIIG+jCCBeKgAwIBAgICAQMwDQYJKoZIhvcNAQEFBQAwQzELMAkGA1UEBh
          MCQVQxCzAJBgNVBAoTAkZIMQ4wDAYDVQQLEwVGSCBUQzEXMBUGA1UE
          AxMORkggQUlLIENBIFJvb3QwHhcNMDcwNDE5MTA0ODAzWhcNMDgwND
          E5MTA0ODAzWjAAMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
          AQEAi0fmE/ep3MGENdFWjJPEABtXOEjccpbsKnmcU9VLo8dD68vbrk
          /+r/jQBKFRZ57svKW8erQzxdAZEnRjiqcwzb3OEQXaUHIBDAs9R//j
          ThzU40Z/HqdZXmYrAquDsButA4DLV1KK/pEM2Yxby653Pw/jLF7ECE
          UsGHPe8ESQVnr5NY9fMUtqiMbTfYbnyVCfEv1OixAZasymWu9gliuq
          XM7wkOX0KunpK5QQS55p2afVwEOxx+gIXtiuUiXDmdQDIo9NnrYJfb
          MfkFASqrm3WEDWMi3kl4t87w5d3qpZrH0bpa2aGC14+VnvpvtK/Css
          R5DkZ0Gvjc0bp2mqaZmCkQIDAQABo4IEOTCCBDUwDwYDVR0TAQH/BA
          UwAwIBADCBxAYDVR0RAQH/BIG5MIG2pEcwRTEWMBQGBWeBBQIBDAtp
          ZDo0OTQ2NTgwMDEXMBUGBWeBBQICDAxTTEQ5NjMwVFQxLjExEjAQBg
          VngQUCAwwHaWQ6MDEwNkRDMEExFzAVBgVngQUCBAwMVFBNIEVtdWxh
          dG9yMRQwEgYFZ4EFAgUMCW1vZGVsIG9uZTEQMA4GBWeBBQIGDAVhbH
          BoYaAmBgVngQUCDwwdQUlLX1RydXN0ZWRfSWRlbnRpdHlfUHJvdmlk
          ZXIwggEpBgNVHSAEggEgMIIBHDBZBgJRDTBTMCMGCCsGAQUFBwIBFh
          dodHRwOi8vZmguYXQvYWlrX3AuaHRtbDAsBggrBgEFBQcCAjAgDB5U
          Q1BBIFRydXN0ZWQgUGxhdGZvcm0gSWRlbnRpdHkwYgYCUQswXDAiBg
          grBgEFBQcCARYWaHR0cDovL2ZoLmF0L2VrX3AuaHRtbDA2BggrBgEF
          BQcCAjAqDChUQ1BBIFRydXN0ZWQgUGxhdGZvcm0gTW9kdWxlIEVuZG
          9yc2VtZW50MFsGAlEMMFUwIgYIKwYBBQUHAgEWFmh0dHA6Ly9maC5h
          dC9wZV9wLmh0bWwwLwYIKwYBBQUHAgIwIwwhVENQQSBUcnVzdGVkIF
          BsYXRmb3JtIEVuZG9yc2VtZW50MCUGA1UdHwQeMBwwGqAYoBaGFGh0
          dHA6Ly9maC5hdC9haWsuY3JsMIIBgwYDVR0JBIIBejCCAXYwFgYFZ4
          EFAhAxDTALDAMxLjECAQECAQIwHAYFZ4EFAhExEzARMAkCAQECAQEC
          AQAEBEFBQUEwLgYDVQQ0MScwCTAHBgVngQUDATALMAkGBSsOAwIaBQ
          AwDTALBgkqhkiG9w0BAQcwgYgGBWeBBQISMX8wfQIBAAEBAIABAIEB
          AKNLFgMyLjIKAQMKAQABAf+AAQCBBCoDBAWiFRYEWFlaMTAJBgUrDg
          MCGgUAAwIAeIMEUwQFBqQVFgRYWVoyMAkGBSsOAwIaBQADAgB4pA0W
          BTE0MC0xCgEBAQEAhQEAFhBodHRwOi8vd2hhdC5ldmVyMIGCBgVngQ
          UCEzF5MHcCAQCgSxYDMi4yCgEDCgEBAQH/gAEAgQQqAwQFohUWBFhZ
          WjEwCQYFKw4DAhoFAAMCAHiDBFMEBQakFRYEWFlaMjAJBgUrDgMCGg
          UAAwIAeKENFgUxNDAtMQoBAQEBAIIBAQEBABYQaHR0cDovL3doYXQu
          ZXZlcjATBgNVHSUEDDAKBggrBgEFBQcDETBrBgNVHSMEZDBigBT9u0
          9OdF3L7Vdfgcmi/Vuf5VD6uKFHpEUwQzELMAkGA1UEBhMCQVQxCzAJ
          BgNVBAoTAkZIMQ4wDAYDVQQLEwVGSCBUQzEXMBUGA1UEAxMORkggQU
          lLIENBIFJvb3SCAQ0wDQYJKoZIhvcNAQEFBQADggEBAMxludPeuNBR
          8MECVCtDqBrXBSxxhnXlg/27RxmjcQHju+2z+R3SELXd8zmosaVScQ
          L3VLySZynwjKL2LRB4jdqZZeYZcblQyT+ZzdEOGkjgnZZUKGG7Ppms
          eVnSGuVSGM+joiwxQ7s9wxecve8nSJ8Z0+rYYpYRc1YE/EmPDGqw4I
          eG8WhifVGwN1z/hC2Yx/FQB2MBKtFDGW5UYAkcWTyDbyrePMP5eH2n
```

```
          F54ThYCUqpZ2qnuY13A8FRRCtLm9zR5W8edZq3yri17D7bhKgEK/Tj
          REONgQa7EmKzfTAZ0c84Q60kktcN48iLfkprLyEysedLEFJEFibEhK
          U7khJm4=
83     </X509Certificate>
84    </AttributeValue>
85   </Attribute>
86  </AttributeStatement>
87  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
88   <ds:SignedInfo>
89    <ds:CanonicalizationMethod Algorithm="http://www.w3.org
        /2001/10/xml-exc-c14n#">
90    </ds:CanonicalizationMethod>
91    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
        xmldsig#rsa-sha1">
92    </ds:SignatureMethod>
93    <ds:Reference URI="#AssertionID">
94     <ds:Transforms>
95      <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig
          #enveloped-signature">
96      </ds:Transform>
97      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc
          -c14n#">
98       <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org
           /2001/10/xml-exc-c14n#" PrefixList="code␣ds␣kind␣rw␣
           saml␣samlp␣typens␣#default␣xsd␣xsi">
99       </ec:InclusiveNamespaces>
100     </ds:Transform>
101    </ds:Transforms>
102    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
         xmldsig#sha1">
103    </ds:DigestMethod>
104    <ds:DigestValue>
105     ogH5KycFIQohDpmx/hox8ejif9A=
106    </ds:DigestValue>
107   </ds:Reference>
108  </ds:SignedInfo>
109  <ds:SignatureValue>
110   CLxfQpY/eDuUOzq/9jFiqKOqV7PFlTZM3R6jWWP8zLcnwFHF2zYk3+4QUot
      Yixw82JCvSvenBpc5XV/cm26DJpi9tWtxTCd3U0tp27pnieECXbNNlJB
      KNuMyR9adQnt0RBrVA7LQweKvHsaEzFKyMUKgwFIn1iOKma6yM7gnn5a
      6IYrN/zkHwGneWa1DCsoLezViugpcTCvqo8JSWLKYZu0Qf06gdQWwycQ
      vUIFfryOgZBn/JZ/CoA+ez/LHdWRwf4J6MnuacW1Gdyj9V32fXdQfBGw
      ll94T0C/HDtmYtI/HOUBjQwzJ0xUueMIwDtLntGd8x5jQQ/L6JRP6Jfb
      hJg==
111  </ds:SignatureValue>
112  <ds:KeyInfo>
113   <ds:X509Data>
114    <ds:X509Certificate>
115     MIICtDCCAZwCBhJBAnD8OzANBgkqhkiG9w0BAQUFADA7MQswCQYDVQQGE
        wJBVDELMAkGA1UEChMCRkgxETAPBgNVBAsTCFRDIEdyb3VwMQwwCgY
        DVQQDEwNBSUswHhcNMDcwNDIzMTEzMTQzWhcNMDgwNDIzMTEzMTQzW
        jAAMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhbi7+ga
        Wjnyb3DqUX2zVpNlH6KU4Kx3Rs4kv17N/lqMXAFCYylv3s/toxW+i/
        FckgDr/b0XNCJB3PCdLLqV8gN5vkhvodfHjuAhVhZTwj7iI6NrLod4
```

```
            etRhPKFbf4ZlkN//yzzzb7LVP3uppKH2Zl3ecxwAZG19vxu/bbKzGm
            wqWmVjMZB1fiOMsiYmVMD6Kf5e7+bF1KX6sEGnaIKWpcNDIRHx7N6X
            a6PStGrFBSOb/jTRd4ZxlqnaZDd7YyFTeDcIG+4VFkmiiTvto26lyS
            LTp+SySJFlbIxoVkrA2dDRNBaTRgnEK1cxFxhTOrEc5QHm9/n6m02w
            hpb0YsSbUBwIDAQABMA0GCSqGSIb3DQEBBQUAA4IBAQBYaiakGnb2n
            HSBz0atD08EWgZWGLA9AcyhH25t74yinpDxn4TLn12+zByOfF1NN1P
            FwEH4r51BHu6yjYkhkLbBfuJI568hkC/9J9b2xeLOaKa369Ilm7PCF
            tx9Dg1XBxWGZQkWNcbd7ovUeNdmQAg34C6stG3/N1H+JXKxKSSDXyh
            S7vd2dvxOrFN2BEGkIzlD1P/IZPXkldD3w7/0pKKACCE+pTa3oezwA
            aYAnpgOvWlw/xDLwkWoos3nUWrFaf3GpleSfluarasH0fW2XnVfGOD
            msJ1GgZUYe1crp7qyJPHRsLslLdAIHbl9iohYcSqv0aVoIe5GzwN4R
            ZoVI7Oy
    </ds:X509Certificate>
   </ds:X509Data>
  </ds:KeyInfo>
 </ds:Signature>
</Assertion>
```

Appendix B

Contents of the CD

B.1 Thesis

The electronic version of the thesis.

Path: /

B.2 Bibliography

This folder contains an electronic version of those bibliography items which are electronically available.

Path: /Bibliography/

B.3 Source Code

In the following folder, all the input and output data files and the source code of the applications are stored:

Path: /Sourcecode/

B.4 Javadoc

The documentation of the source code is stored in the following folder:

Path: /Javadoc/

B.5 Software

In the following folder, all the used programs and libraries are stored:

Path: /Software/

List of Acronyms

AACP	Asymmetric Authorization Change Protocol
ADA	Authorization Decision Assertion
ADCP	Authorization Data Change Protocol
ADIP	Authorization Data Insertion Protocol
AIK	Attestation Identity Key
CRTM	Core Root of Trust for Measurement
DAA	Direct Anonymous Attestation
EK	Endorsement Key
IAIK	Institute of Applied Information Processing and Communication
IDM	Identity Management
IMA	Identity Management Architecture
OASIS	Organization for the Advancement of Structured Information Standards
OIAP	Object-Independent Authorization Protocol
OSAP	Object-Specific Authorization Protocol
PCR	Platform Configuration Register
PDP	Policy Decision Point
PE	Platform (endorsement)
PEP	Policy Enforcement Point
RM	Reference Manifest
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SAML	Security Assertion Markup Language

SOAP	Simple Object Access Protocol
SRK	Storage Root Key
TBB	Trusted Building Block
TCG	Trusted Computing Group
TCS	TSS Core Services
TDDL	TPM Device Driver Library
TPM	Trusted Platform Module
TSP	TCG Service Provider
TSS	Trusted Software Stack
UUID	Universal Unique Identifier
WSS	Web Services Security
XACML	eXtensible Access Control Markup Language

Glossary

Certificate Management Messages over CMS

The Certificate Management Messages over CMS (CMC) protocol is a product of the IETF which uses the PKCS#10 Certificate Request Syntax standard for the basic request format and the PKCS#7 Cryptographic Message Syntax for the protection of the exchanged messages.

Common Criteria

The Common Criteria (CC) is an international standard (ISO/IEC 15408) for the evaluation of security products and systems. It offers a framework for the definition of security requirements and the evaluation based on different assurance levels.

Digital Rights Management

Digital Rights Management (DRM) refers to technologies used by publishers or copyright owners to control access to or usage of digital content.

Hash Function

A cryptographic hash function is a function which takes a long input message of arbitrary length and produces a string of fixed length as output. The hash function has to assure that even very similar input messages lead to completely different outputs. A hash function is also known as one-way function because it is extremely difficult to find an input message leading to a specific output message. Another important factor is that it has to be almost impossible to find two messages which lead to the same arbitrary output.

HMAC

A keyed-hash message authentication code (HMAC) is a type of message authentication code which uses a cryptographic hash function in combination with a secret key. A HMAC function can be used to verify data integrity and authenticity of the message at once.

Identifier Domain

An identifier domain is exactly the area in which a user is known by a particular identity provider (trusted third party).

Kerberos

Kerberos is a ticket-based computer network authentication protocol, which allows individuals to communicate over an insecure network to prove their identity to another party in a secure manner. The protocol is based on symmetric cryptography and requires a trusted third party.

LDAP

The Lightweight Directory Access Protocol (LDAP) is an application protocol for querying and modifying directory services running over TCP/IP.

OASIS

The Organization for the Advancement of Structured Information Standards (OASIS) is an international, non-profit organization which deals with the development of standards mainly related to e-business and web services.

Protection Profile

A Protection Profile (PP) is a document used in conjunction with a Common Criteria evaluation. Typically, it specifies the security requirements for the evaluation object. Depending on the evaluation assurance level, the evaluation is more or less detailed.

Public Key Encryption

Public key (asymmetric) encryption is a form of cryptography in which a user owns a key pair consisting of a mathematically related public and a private key. It is not possible to derive the private key from the public key and vice versa. Messages encrypted with the public key can only be decrypted with the belonging private key. The public key can be widely distributed whereas the private key has to be kept secret.

Remote Procedure Calls

The Remote Procedure Call (RPC) is a technology which allows a computer program on one machine to call a subroutine or procedure executed in another address space (typically on another computer on the network).

Symmetric Encryption

Symmetric encryption is a form of cryptography in which a single secret key is used for both, encryption and decryption.

W3C

The World Wide Web Consortium (W3C) is the main international standards organization responsible for the specification of standards related to the World Wide Web.

X.509 Certificates

In general, the X.509 standard is an ITU-T standard for public key infrastructure. Among other things, it specifies standard formats for public key certificates (X.509 certificates) and a certificate path validation algorithm.

XML

The eXtensible Markup Language (XML) is a general-purpose markup language which allows to specify text and additional information about the text. The additional information is expressed using markups which are embedded within the primary information. XML especially uses elements which can be additionally characterized with attributes and the actual value.

XML Encryption

XML Encryption is specified in a W3C recommendation which defines an XML syntax for digital encryption of XML content.

XML Key Management Services

The XML Key Management Services (XKMS) protocol defined by the W3C is a request-response protocol based on SOAP for the credential management for existing CAs in the PKI industry. It consists of the XML Key Registration Service and the XML Key Information Service Specification. The XKMS protocol offers a framework to express certificate management functions in XML in providing a wrapper over legacy CA services designed for X.509 certificates.

XML Signatures

XML Signatures are specified in a W3C recommendation that defines an XML syntax for digital signatures. In general, XML signatures are quite similar to PKCS#7 but instead the XML framework offers more extensibility and is especially aimed at signing XML documents. The XML signature itself is specified in XML syntax too.

Bibliography

[1] ASN.1 Information site, Object identifier (OID) repository, April 2007. http://oid.elibel.tm.fr.

[2] M. Bartel, J. Boyer, B. Fox, B. LaMiacchia, and E. Simon. XML-Signature Syntax and Processing W3C Recommendation. *World Wide Web Consortium*, February 2002. http://www.w3.org/TR/xmldsig-core/.

[3] N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah. *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap.* IBM Press, 2005.

[4] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145, 2004.

[5] Bundesamt für Sicherheit in der Informationstechnik. Das Trusted Platform Module (TPM) und vertrauenswürdige Informationstechnik, March 2007. http://www.bsi.bund.de/sichere_plattformen/trustcomp/infos/tpm_report/index.htm.

[6] J. Camenisch. Better Privacy for Trusted Computing Platforms. *Computer Security - ESORICS 2004. 9th European Symposium on Research in Computer Security. Proceedings (Lecture Notes in Computer Science Vol. 3193)*, pages 73–88, 2004.

[7] J. Camenisch. Protecting (anonymous) credentials with the trusted computing group's trusted platform modules v1. 2. *Proceedings of the 21st IFIP International Information Security Conference (SEC 2006)*, 2006.

[8] J. Camenisch, E. Brickell, and L. Chen. Direct anonymous attestation: Achieving privacy in remote authentication, 2004. Available at http://www.zisc.ethz.ch/events/ISC2004Slides/folien-jan-camenisch.pdf.

[9] J. L. Camp. Digital identity. *Technology and Society Magazine, IEEE*, 23(3):34– 41, 2004.

[10] A. Durand. Three tiers of identity. Digital ID World Newsletter, March 2002. http://www.digitalidworld.com.

[11] C. Eckert. *IT-Sicherheit. Konzepte, Verfahren, Protokolle*. Oldenbourg, 3. edition, September 2004.

[12] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization (RFC 3281). *Internet Engineering Task Force, Network Working Group*, April 2002.

[13] S. Fischer-Hübner. *It Security and Privacy: Design and Use of Privacy-enhancing Security Mechanisms*. Springer, 2001.

[14] B. Galbraith, W. Hankison, A. Hiotis, M. Janakiraman, D. V. Prasad, R. Trivedi, and D. Whitney. *Professional Web Services Security*. Wrox Press, 2002.

[15] S. Gürgens and C. Rudolph. AIK Certification. 2006.

[16] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. Technical report, W3C, June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

[17] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 2: Adjuncts. Technical report, W3C, June 2003. http://www.w3.org/TR/2003/REC-soap12-part2-20030624/.

[18] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC 3280). *Internet Engineering Task Force, Network Working Group*, April 2002.

[19] J. Hughes and E. Maler. Security Assertion Markup Language (SAML) Version 1.1 Technical Overview, Committee Draft. Technical report, OASIS, May 2004.

[20] IBM Corporation and Microsoft Corporation. Security in a Web Services World: A Proposed Architecture and Roadmap, 2002. Available at http://www.verisign.com/wss/.

[21] T. Imamura, B. Dillaway, and E. Simon. XML Encryption Syntax and Processing W3C Recommendation. *World Wide Web Consortium, February*, December 2002. http://www.w3.org/TR/xmlenc-core/.

[22] Institute for Applied Information Processing and Communication (IAIK), Graz University of Technology. Secure Information and Communication Technologies, April 2007. http://jce.iaik.tugraz.at.

[23] Institute for Applied Information Processing and Communication (IAIK), Graz University of Technology. Trusted Java, April 2007. http://trustedjava.sourceforge.net.

[24] Intel. Intel Trusted Execution Technology, April 2007. http://www.intel.com/technology/security.

[25] International Business Machines Corporation (IBM). TrouSerS - the open-source TCG Software Stack, April 2007. http://trousers.sourceforge.net.

[26] Internet2. OpenSAML - an Open Source Security Assertion Markup Language implementation, April 2007. http://www.opensaml.org.

[27] Internet2. Shibboleth, April 2007. http://shibboleth.internet2.edu/.

[28] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope. Trust requirements in identity management. In *ACSW Frontiers '05: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, pages 99–108, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[29] S. Josefsson. The Base16, Base32, and Base64 Data Encodings (RFC 3548). *Internet Engineering Task Force, Network Working Group*, October 2006.

[30] J. Kemp, S. Cantor, P. Mishra, R. Philpott, and E. Maler. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS, March 2005.

[31] N. Kuntze and A. U. Schmidt. Transitive trust in mobile scenarios. In *Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006)*, LNCS 3995:73-85, 2006.

[32] N. Kuntze and A. U. Schmidt. Trusted computing in mobile action. In *Peer-reviewed Proceedings of the ISSA 2006 From Insight to Foresight Conference*. Information Security South Africa (ISSA), 2006.

[33] N. Kuntze and A. U. Schmidt. Trusted ticket systems and applications. In *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.

[34] Liberty Alliance Projet. The Liberty Alliance Project, April 2007. http://www.projectliberty.org.

[35] E. Maler, P. Mishra, and R. Philpott. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) Version 1.1. Technical report, OASIS, September 2003.

[36] E. Maler, P. Mishra, and R. Philpott. Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) Version 1.1. Technical report, OASIS, September 2003.

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

[38] Microsoft Corporation. Trusted Platform Module Services in Windows Longhorn, April 2005. Available at http://www.microsoft.com/resources/ngscb/WinHEC05.mspx.

[39] Microsoft Corporation. BitLocker Drive Encryption: Value-Add Extensibility Options, May 2006. Available at http://www.microsoft.com/whdc/system/platform/hwsecurity/BitLockerExt.mspx.

[40] Microsoft Corporation. Microsoft Windows Vista Security Advancements, June 2006.

[41] Microsoft Corporation. Next-Generation Secure Computing Base, April 2007. http://www.microsoft.com/resources/ngscb.

[42] Microsoft Corporation. System Integrity Team Blog, March 2007. http://blogs.msdn.com/si_team.

[43] Microsoft Corporation. Windows Communication Foundation, April 2007. http://wcf.netfx3.com.

[44] C. Mitchell. *Trusted Computing*. Institution of Engineering and Technology, November 2005.

[45] N. Mitra. SOAP Version 1.2 Part 0: Primer. Technical report, W3C, June 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[46] H. M. Mountain, J. Kopecky, S. Williams, G. Daniels, and N. Mendelsohn. SOAP Version 1.2 Email Binding. Technical report, W3C, June 2002. http://www.w3.org/TR/2002/NOTE-soap12-email-20020626.

[47] G. C. Necula and P. Lee. *Proof-carrying Code*. School of Computer Science, Carnegie Mellon University, 1996.

[48] E. Newcomer. *Understanding Web Services. XML, WSDL, SOAP and UDDI.: XML, WSDL, SOAP and UDDI*. Addison-Wesley Longman, Amsterdam, May 2002.

[49] OASIS. OASIS eXtensible Access Control Markup Language (XACML) TC, April 2007. http://www.oasis-open.org/committees/xacml/.

[50] OASIS. OASIS Security Services (SAML) TC, April 2007. http://www.oasis-open.org/committees/security/.

[51] OASIS. OASIS Web Services Security (WSS) TC, April 2007. http://www.oasis-open.org/committees/wss/.

[52] A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. *Proceedings of the 2004 workshop on New security paradigms*, pages 67–77, 2004.

[53] R. Sailer, L. van Doorn, and J. P. Ward. The Role of TPM in Enterprise Security. *Number RC23363 (W0410-029), October*, 6, 2004.

[54] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. *Proceedings of the 13th USENIX Security Symposium*, pages 223–238, 2004.

[55] M. Strasser. TPM Emulator, April 2007. http://tpm-emulator.berlios.de.

[56] The Apache XML Project. Apache XML Security, April 2007. http://santuario.apache.org.

[57] Trusted Computing Group. Trusted Computing Platform Alliance (TCPA) Main Specification, Version 1.1b, February 2002. Available at https://www.trustedcomputinggroup.org.

[58] Trusted Computing Group. TCG Software Stack (TSS) Specification, Version 1.10 Golden, August 2003. Available at https://www.trustedcomputinggroup.org.

[59] Trusted Computing Group. Work Group Charter Summary, 2005. Available at https://www.trustedcomputinggroup.org.

[60] Trusted Computing Group. TCG Credential Profiles, Specification Version 1.0, Revision 0.981, January 2006. Available at https://www.trustedcomputinggroup.org.

[61] Trusted Computing Group. TCG PC Client Specific Implementation Specification For Conventional BIOS, Version 1.20, Revision 1.00, July 2006. Available at https://www.trustedcomputinggroup.org.

[62] Trusted Computing Group. TCG Specification - Architecture Overview, Version 1.2, 2006. Available at https://www.trustedcomputinggroup.org.

[63] Trusted Computing Group. TPM Main Part 1 Design Principles, Version 1.2, 2006. Available at https://www.trustedcomputinggroup.org.

[64] Trusted Computing Group, Infrastructure Working Group. Reference Architecture for Interoperability (Part 1), Version 1.0, 2005. Available at https://www.trustedcomputinggroup.org.

[65] Trusted Computing Group, Infrastructure Working Group. Architecture Part II - Integrity Management, Version 1.0, 2006. Available at https://www.trustedcomputinggroup.org.

[66] VeriSign. Enabling Trusted Web Services, April 2007. http://www.verisign.com/wss/.

[67] P. J. Windley. *Digital Identity.* O'Reilly Media, 1. edition, August 2005.