Universität der Bundeswehr München

A Concept for a Trustworthy Integration of Smartphones in Business Environments

Joerg Vieweg

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Peter Hertling
1. Berichterstatter:	Prof. Dr. Gabi Dreo Rodosek
2. Berichterstatter:	Prof. Dr. Udo Helmbrecht
1. Prüfer:	Prof. Dr.-Ing. Mark Minas
2. Prüfer:	Prof. Dr.-Ing. Helmut Mayer

Tag der Prüfung: 25.07.2013

Neubiberg, August 2013

Contents

Abstract

Smartphones are commonly used within business environments nowadays. They provide sophisticated communicational means which go far beyond simple telephone capabilities. Email access and particular apps on the device are examples of their versatile abilities. While these features allow them to be used in a very flexible way, e.g. in different infrastructures, they impose new threats to their surrounding infrastructure. For example, if used in an environment which allows the installation of custom apps, malicious software may be placed on the device. In order to mitigate these threats, a detailed awareness combined with the possibility to enforce certain constraints on such devices need to be established. In detail, it is necessary to include such devices into a decision making process which decides about the policy compliance of such devices. The policy used in this process defines the rules which apply to the particular infrastructure, e.g. if custom apps are allowed or if a specific software version may not be allowed. However, even when relying on this process, there is one limitation as it does not include a trust-based evaluation. This leads to the problem that a malicious smartphone might compromise the information used for the decision making process which should determine the policy compliance of this device. This renders the overall approach ineffective as the decision wether a device is policy compliant or not may be false. Given that, the thesis presented here provides means to evaluate the trustworthiness of such information to allow a trustworthy decision making about the policy compliance. It therefore introduces two things: (1) a generic trust model for such environments and (2) a domain-specific extension called Trustworthy Context-related Signature and Anomaly Detection system for Smartphones (TCADS). The trust model (1) allows to specify, to calculate and to evaluate trust for the information used by the decision making process. More in detail, the trust founding process of (1) is done by introducing so-called security properties which allow to rate the trustworthiness of certain aspects. The trust model does not limit these aspects to a particular type. That is, device-specific aspects like the number of installed apps or the current version of the operating system may be used as well as device independent aspects like communicational parameters. The security properties defined in (1) are then used to calculate an overall trust level, which provides an evaluable representation of trust for the information used by the decision making process. The domain-specific extension (2) uses the trust model and provides a deployable trust-aware decision making solution for smartphone environments. The resulting system, TCADS, allows not only to consider trust within the decisions about the policy compliance but also enables to base the decisions solely on the trust itself. Besides the theoretical specification of the trust model (1) and the domain-specific extension (2), a proof of concept implementation is given. This implementation leverages both, the abilities of the generic trust model (1) as well as the abilities of the TCADS system (2), thus providing a deployable set of programs. Using this proof of concept implementation, an assessment shows the benefits of the proposed concept and its practical relevance. A conclusion and an outlook to future work extending this approach is given at the end of this thesis.

Zusammenfassung

Smartphones sind in heutigen Unternehmensnetzen mittlerweile nicht mehr wegzudenken. Über einfache Telefonie-basierte Fähigkeiten hinaus bieten sie Eigenschaften wie zum Beispiel Email-Zugriff oder hohe Anpassbarkeit auf Basis von Apps. Obwohl diese Funktionalitäten eine vielseitige Nutzung solcher Smartphones erlauben, stellen sie gleichzeitig eine neuartige Bedrohung für die umgebende Infrastruktur dar. Erlaubt eine spezifische Umgebung beispielsweise die Installation von eigenen Apps auf dem Smartphone, so ist es über diesen Weg möglich, Schadprogramme auf dem Gerät zu platzieren. Um diesen Bedrohungen entgegenzuwirken, ist es zum einen nötig Smartphones in der jeweiligen Umgebung zu erkennen und zum anderen, Richtlinien auf den jeweiligen Geräten durchsetzen zu können. Die durchzusetzenden Richtlinien legen fest, welche Einschränkungen für die jeweilige Umgebung gelten, z.B. die Erlaubnis zur Installation von eigenen Apps oder die Benutzung einer bestimmten Softwareversion. Aber auch wenn eine entsprechende Lösung zur Einbeziehung von Smartphones in die Infrastruktur verwendet wird, bleibt ein Problem ungelöst: die Betrachtung der Vertrauenswürdigkeit von durch das Smartphone bereitgestellten Informationen. Diese Einschränkung führt zu dem Problem, dass ein entsprechend kompromittiertes Smartphone die Informationen, welche zur Entscheidungsfindung über die Richtlinienkonformität des Gerätes verwendet werden, in einer Art und Weise ändert, welche den gesamten Entscheidungsprozess ineffizient und somit wirkungslos macht. Die hier vorliegende Arbeit stellt daher einen neuen Ansatz vor um einen vertrauenswürdigen Entscheidungsprozess zur Regelkonformität des Gerätes zu ermöglichen. Im Detail werden dazu zwei Ansätze vorgestellt: (1) Ein generisches Modell für Vertrauensürdigkeit sowie eine (2) domänenspezifische Abbildung dieses Modells, welches als Trustworthy Context-related Signature and Anomaly Detection system for Smartphones (TCADS) bezeichnet wird. Das Modell für Vertrauenswürdigkeit (1) erlaubt die Definition, Berechnung und Auswertung von Vetrauenswürdigkeit für Informationen welche im Entscheidungsprozess verwendet werden.Im Detail basiert die Vertrauenswürdigkeitsbestimmung auf Grundfaktoren für Vertrauen, den sogenannten Sicherheitseigenschaften. Diese Eigenschaften bewerten die Vertrauenswürdigkeit anhand von bestimmten Aspekten die entweder Gerätespezifisch und Geräteunabhängig sein können. Basierend auf dieser Bewertung wird dann eine Gesamtvertrauenswürdigkeit, der sogenannte Trust Level berechnet. Dieser Trust Level erlaubt die Berücksichtigung der Vertrauenswürdigkeit bei der Entscheidungsfindung. Teil (2) der Lösung stellt, basierend auf dem Modell der Vertrauenswürdigkeit, ein System zur vertrauensbasierten Entscheidungsfindung in Smartphone Umgebungen bereit. Mit diesem System, TCADS, ist es nicht nur möglich, Entscheidungen auf ihre Korrektheit bezüglich der Vertrauenswürdigkeit zu prüfen, sondern auch Entscheidungen komplett auf Basis der Vertrauenswürdigkeit zu fällen. Neben dem allgemeingültigen Modell (1) und dem daraus resultierenden domänenspezifischen System (2), stellt die Arbeit außerdem einen Tragfähigkeitsnachweis in Form einer Referenzimplementierung bereit. Diese Implementierung nutzt sowohl Fähigkeiten des Modells der Vertrauenswürdigkeit (1) als auch des TCADS Systems (2) und stellt ein nutzbares Set von Programmen bereit. Eine Evaluierung basierend auf diesem Tragfähigkeitsnachweis zeigt die Vorteile und die Praktikabilität der vorgestellten Ansätze. Abschließend findet sich eine Zusammenfassung der Arbeit sowie ein Ausblick auf weitereführende Fragestellungen.

1 Introduction

Contents

This chapter gives an introduction to the thesis. First, the motivation is explained outlining the problems which arise when integrating smartphones into a business infrastructure. Based on these findings, the second part presents the research questions which are addressed within this thesis. Finally, the outline of the thesis including a short overview of each chapter is given.

1.1 Motivation

Smartphones are commonly used nowadays, not only within the private sector but also in business infrastructures (cf. [1]). Due to this and the special properties of such phones, like the app-centric architecture and their versatile usage profile, providers of such infrastructures need to pay attention to them. In particular, the threats which are imposed by such devices are different to other threats in such infrastructures [2]. Taken the current situation concerning smartphones, there is a huge amount of different types of threats for such devices [3]. It starts with the operating system of the device itself, like the Android operating system [4], that has become the most popular mobile OS with a current market share of about 68 percent [5, 6], being itself vulnerable to certain attacks. For example, particular Android OS versions have been vulnerable to an attack which could be triggered by special data sent to Android's volume manager allowing to gain root access [7]. This provides a way to perform a privilege escalation attack (cf. [8]). Another example, which came up during the time of writing this thesis is an insufficiently protected memory access on some of Samsung's Android-based phones. It allows an attacker to gain root access out of an app by accessing the memory [9]. A detailed analysis about the problems with Android's OS can be found in [10, 11]. In addition, even higher level parts of the OS might provide a possibility for attackers to compromise the device and use it for their purposes. Android uses so-called permissions to limit the rights that a program (i.e. an app) has on the system. A flaw of this system presented in [12] can be used to gain additional rights with a consequence that a program may behave in an unexpected way.

Even if mitigating the threats which are caused by the operating system itself, there are additional problems which need to be taken into account when dealing with smartphones. Their specialised architecture, which condenses most functionality in apps, introduces another group of threats: the so-called malapps. Malapps allow an attacker to use the smartphone for their purposes without the legitimate user of the smartphone having knowledge about this. For example, Schlegel et al. [13] present a malapp called soundcomber. This malapp acts unnoticed on the phone and allows to record the sound from the phone's microphone. While this may be not critical at first view, it can be used to mount so-called sensor sniffing attacks (cf. [14]). If carried out in the appropriate environment, e.g. during a business meeting where a company's internal planning is discussed, this could have a rather critical impact. The presented types of attacks are not only common to Android-based devices but can also be found for other device types. Given the iOS operating system used on Apple's devices like the iPhone or the iPad, the iSAM approach explained in [15] provides a combined attack by first jailbreaking (i.e. removing the manufacturers restrictions, cf. [16]) the device and afterwards by installing a malapp. More examples for malware, not only for Android and iOS but also for Symbian-based devices are given in [17]. These examples show two things: (1) the compromise of smartphone devices is not limited to a particular type (i.e. OS or manufacturer) and (2) with the smartphone's platforms developing further, the threats develop as well. Moreover, when integrating such devices into network-based infrastructures, these devices have to be managed as well. In particular, smartphones render a special class of devices and must be treated in a proper way if including them into a network-based environment. Current approaches provide a way of performing a network-based management along with an intrusion detection for this class of devices. That is, a decision making about the smartphone's compliance with the network's policy is performed. This is done by collecting certain kind of information (i.e. properties, cf. [2]) from such devices, like the installed apps or their operating system. In addition, information provided by the other components within the network is also used. Examples for these kind of information sources may be the intrusion detection systems or components like the DHCP server or the DNS server. By using the combination of the smartphone and network-based information, an overall view of the network is provided. This view does include the smartphones as well as the other network components. Approaches like the CADS system [2, 18, 19] provide this view. Besides the network-based view, complex policies can usually be defined. These policies are used for handling smartphone devices within the infrastructure. Handling refers to a range of actions which can be taken after making a decision about the device's compliance to such policies. These actions range from a simple monitoring of a smartphone to a complex enforcement including a whole set of components. Furthermore, approaches are based on collecting information from a set of components, not only from the smartphones but also from components within the infrastructure. For example, if a phone uses a certain service of the network, these services provide information about the usage. The service usage may be monitored, and if the usage exceeds certain limits or other constraints, like the physical access location of the smartphone, an action is triggered. Although this provides a holistic, policy-based decision making process, another problem is identified. The information collected from the particular sources is not distinguished in terms of trustworthiness. The assumption of the system is, that every collected information is true and trustworthy. This may be appropriate for certain scenarios. However, it is a limited view when considering the problems above, as the information is provided by different sources like the smartphones itself or the infrastructure. If one of these sources is compromised, it may provide false information. This false information would afterwards being used within a decision making process. The compromise of such a source may be either intentionally or unintentionally. An attack which was successfully carried out is only one example for an intended compromise. An unintentional compromise could be due to a misconfiguration of the system. Given the smartphones itself, they are commonly more threatened than the infrastructure-based components which are statically residing at one place and do not offer the same versatility as the smartphones. For example, the malapps which have been mentioned above, may be enhanced to compromise the information collection process on the smartphone. This becomes even more critical, when considering approaches like bring your own device (BYOD, cf. [20]), which allows employees to use their own equipment, in particular their own smartphone, within the company's infrastructure. Due to the fact, that this employee uses that device for multiple tasks, including personal ones where security considerations are not enforced (i.e, security considerations may play a minor role) there is a rather high chance of the device becoming compromised. Furthermore, devices used by employees which are provided by the company get usually administered by the company's IT support. This is not the case when allowing a BYOD strategy. Given all that, it is necessary to differentiate between the trustworthiness of the data independently from where this data was collected. In case of BYOD, it needs to be distinguished between data collected at devices which are administered by the company and employee-owned devices. This would improve the decision making process that is based on these information essentially. At first, it would be more reliable against attacks which aim on providing false information leading to a false decision, thus addressing issues like false positive and false negative rates. Furthermore, the measured trust itself can be interpreted as data which influences the decision making process for a more versatile system.

1.2 Research Questions

There are three research questions which are addressed in this thesis:

1. Which data and characteristics can be used to derive trust?
2. How can trust be defined and calculated?
3. How is it possible to use trust in the decision making process?

The first question deals with the problems of analysing the basis (e.g. data, characteristics of smartphones, environment) for the calculation of trust. More precisely, if a device is compromised, certain characteristics of this device indicate this. For example, if a smartphone is compromised by a malapp, the running processes would indicate this. In this case, the processes represent a characteristic of the device. Such characteristics can be used as factors, influencing the device's trustworthiness. By providing several of these fine grained low level trust factors which do not only address the component (i.e. the device) but also the communication channel of this device, an adequate basis is given. This is done in detail in section 4.2.2. Due to the fact, that there are several trust deriving factors provided and not only one, answering this part leads to the second research question.

This second question deals with the problem of the trust definition and calculation. The basic definition of trust is given in section 4.2.1. A more refined version of this definition, which includes the findings for answering the first research question is given in 4.2.3. If trust can be derived, which is done by answering the first question, there must be a way to calculate trust. This is necessary to combine the single characteristics addressed in the first question into a usable trust value. Section 4.2.3 explains the process that is used to answer this question. As a result, a trust level describing the trustworthiness of data is provided. Furthermore, as it is likely that trust is not based on only one single factor (i.e. characteristics of the device) but on a varying set of factors, answering this question provides a way of combining these factors into an evaluable trust measurement.

Although addressing the first two questions allows to derive and calculate trust, there is no method of actually using this trust evaluation within a decision making process. Answering the third question provides this method of using trust in the decision making process. In detail, a way is given to express a domain-specific trust measurement which can be included into the scenario's decision making process. Sections 4.3 and 4.4 answer this question by giving a generic data representation of the trust model and a domain-specific extension. The domain-specific extension combines all previous results into an actually usable system, which allows a trust-aware decision making. Furthermore, necessary components are introduced and explained within these sections.

By answering all three questions, it is possible to collect data and evaluate this data in terms of their trustworthiness, particularly in relevant scenarios.

1.3 Outline of the Thesis

The thesis is organised as shown in figure 1.1. Chapter 1 emphasises the general situation when using and managing smartphones in business environments. Chapter 2 deals with the particular use cases: First of all, the reference infrastructure, which defines the basic environment, is introduced. Based on this reference infrastructure, four particular scenarios are analysed: the first deals with general trustworthiness of collected information, the second deals with expressing policy-based decisions, the third shows a way to provide tailored services and finally, the fourth shows a forensic-based approach. Summarising this, the chapter is finished with the identification of the overall requirements that need to be fulfilled in order to solve the problems described. The current state of

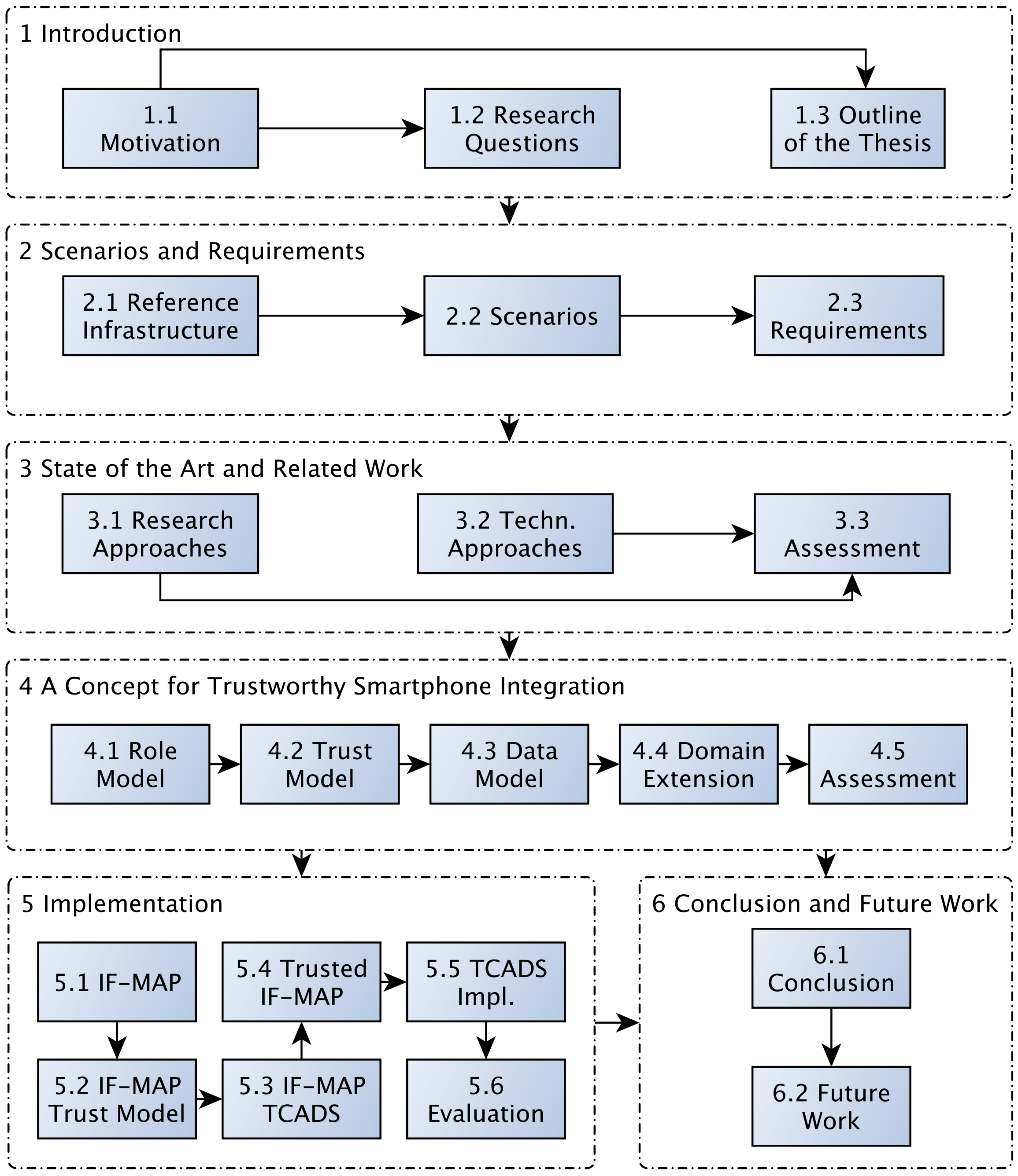


Figure 1.1: Outline of the thesis. Chapters are shown dashed with the appropriate sections (rectangles) inside them. Arrows depict the common thread and the usage of particular results from previous chapters and sections.

the art in research and technology is analysed in chapter 3. The relevant related work is categorised and analysed in terms of the fulfilment of the requirements stated in the previous chapter. The chapter is finished with a conclusion of the requirements and their fulfilment by the related work as well as the technological-based approaches. Based on these findings, a generic trust model and a domain-specific extension of this model is developed in chapter 4. The trust model introduces a trust evaluation system. In detail, it combines basic (i.e. low level) trust factors to high level trust values which allow to evaluate data in terms of their trustworthiness. The low level trust factors, so-called security properties, are assigned to each relevant component within the network environment. In addition, not only to the component itself but also to the communication channels between the components. If a component provides data for the decision making process, the security properties are combined into a single trust level for the collected data. This trust level can be evaluated (i.e., concrete values can be assigned to it). The evaluation is used within the domain-specific extension which provides a trust-aware decision making system. Chapter 5 describes an implementation of the domain-specific extension by the use of IF-MAP and provides a proof of concept by this. Furthermore, an evaluation of the proof of concept implementation is given in this chapter. It is based on a particular investigation of performance effects and on the results of tests with a defined set of test cases summarising the scenarios, thus also showing the practical usage of the overall approach (i.e., of the generic trust model as the domain-specific extension leverages this model). Finally, chapter 6 summarises all parts including a final assessment of the approach and gives an outlook to future work.

2 Scenarios and Requirements

Contents

This chapter introduces the reference infrastructure and the relevant scenarios upon this infrastructure. All scenarios are based on the ESUKOM [21] research project which deals with smartphone security in business infrastructures. Due to this, these scenarios represent real world use cases and point out current problems in this research field. By analysing these scenarios relevant requirements are identified.

2.1 Reference Infrastructure

The following section describes the infrastructure that is common to all scenarios. The components which are used to form the basic environment are widely adopted. They are therefore commonly found within company (i.e. business) networks. Figure 2.1 depicts the overall layout and the components that are part of the infrastructure. More details are given in the following.

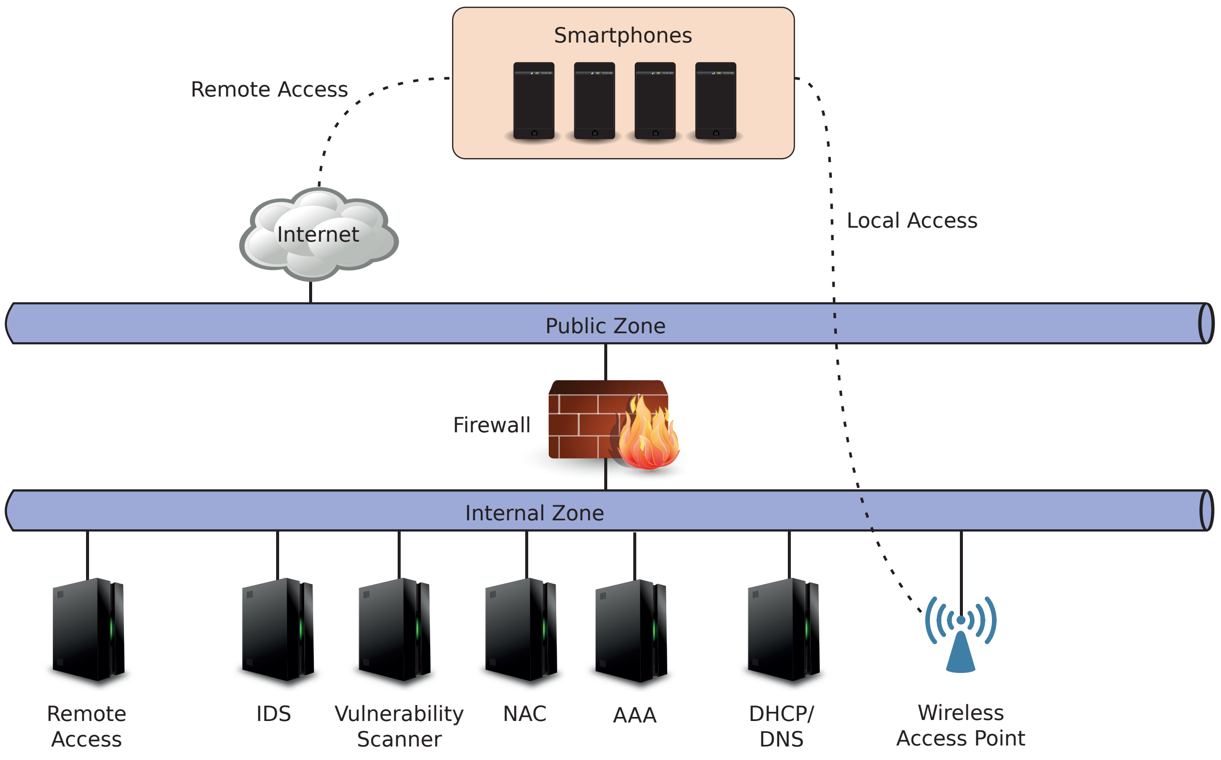


Figure 2.1: Reference Infrastructure (cf. [2]).

2.1.1 Zone Topology

The topology of the network is divided into two separated zones, the public and the internal zone. The public zone describes the logical area, which is located outside of the company's network and therefore also outside of the company's administrative influence. There may be arbitrary devices in this section which can be connected to the network by two different means. The first connection possibility is the direct connection using a wireless technology, for example by leveraging the 802.11 [22] standard. The term direct connection refers to a direct, logical network connection without any intermediary components in between. The second option which may also be used for connection purposes is the indirect connection. This is the case if the device connects to another network which is connected to the Internet and tunnels trough both networks. For example, this is the case, if a Virtual Private Network (VPN) is used. The infrastructure distinguishes between these two types of connection means: local access is used if a direct connection is meant and remote access if using an Internet-based connection.

In detail, a more fine grained version of the topology is actually used. It is formed by placing another zone in between the public and the internal zone. This zone, commonly referred to as demilitarised zone (DMZ), consists primarily of components which are used to separate the internal and the public zone from each other, e.g. firewall systems. Furthermore, this zone provides the possibility to place components which should be accessible from outside the network without the need to allow a direct connection to the internal zone. A VPN gateway or a proxy server are examples for such kind of components. For simplicity issues, this kind of zone is interpreted as part of the internal zone within this thesis.

There are two types of components which can be found within the zones: infrastructure-based components and endpoints. Infrastructure-based components are required in order to provide the basic function of the network along with services the infrastructure should provide while endpoints are user controlled devices which make use of the infrastructure and the services.

2.1.2 Infrastructure Components

The basic infrastructure used by the scenarios consists of the following infrastructure-based components. These components provides either a service to endpoints or control certain aspects, like access decisions, of the network.

Access Point An access point enables the use of wireless connection methods for the direct connection to the network. It is used by endpoints to gain access to the network.

Firewall A firewall models the perimeter between the public and the internal zone. It controls the inbound as well as the outbound network traffic and is able to block certain types of communication on demand. Access to a particular service requested by endpoints is also controlled by this device. That is, if a component recognises an unauthorised service access, the firewall may block this access for the specific device.

Intrusion Detection/Prevention System An IDS/IPS system monitors the internal zone of the network and provides information about intrusions into this network. In this infrastructure, the network's traffic along with the state of the platforms is monitored. That is, there are sensors on each component of the infrastructure which collect data about this particular component. This data is then used by the IDS, combined with the traffic analysis, for signature and anomaly detection means. The overall behaviour is controlled by a policy which defines the necessary parts, like signatures and anomalies. When the system detects a violation of this policy, it reacts and triggers certain events like a rule change on the firewall. Due to this behaviour, the IDS can be considered as decision making component of the network as it decides about the fulfilment of a certain policy.

Vulnerability Scanner This system is responsible for monitoring particular platforms. I.e., it scans a device and checks for certain properties. The discovered properties of the particular platform may then be used as input for the IDS or to generate alert messages.

Network Access Control System A Network Access Control system works in conjunction with the firewall to allow or disallow the overall access to the private zone. It therefore takes properties of the device which wants to access the network (i.e. a smartphone) and verifies them against a policy. Furthermore, a user based access decision can be made: the user of the device which wants to gain access needs to authenticate herself in order to be given access privileges. The access decisions as well as the information collected by the NAC system can be used by other components, for example the IDS, in order to allow a more sophisticated decision making process.

Remote Access System A Remote Access System provides means to indirectly connect to the network. This can be done for instance by using a Virtual Private Network Server as Remote Access System. It allows the smartphone to connect through the internet to the infrastructure.

AAA An AAA (Authentication, Authorisation and Accounting) system provides a backend within the infrastructure holding all user and system-specific credential information. That is, if another infrastructural component needs user-specific account data, it can request these data from this component. This component commonly works together with the NAC system in order to share user-specific data.

DHCP and DNS A DHCP system is responsible for managing the addresses used in the infrastructure while the DNS manages name queries. The DHCP provides address-specific information and the DNS provides information about name queries and record changes.

All of these infrastructure-based components, excluding the IDS, itself are monitored. This is done by using appropriate sensors on the particular platforms. These sensors communicate with the IDS and allow to detect intrusion and react on events within the network.

2.1.3 Endpoints

Within this infrastructure, only one type of endpoints is in the focus: smartphones. In a real world environment, there would also be other types, like laptops or special devices like terminals. More precisely, smartphones connect to the public zone of the network either directly or indirectly. A direct connection is established if the user of the phone connects (usually using a wireless network-based connection) at the company's location to the infrastructure. An indirect connection is used if the smartphone is used anywhere else. This indirect connection may then either happen by leveraging the VPN service or via the cellular network. If the smartphone is successfully connected, they can use services provided by the network. These services are not explicitly introduced by the reference infrastructure as they may be of an arbitrary type. Examples for such services are an email service or a business suite providing calendars and reminders.

As with the infrastructure-based components, endpoints which are connected to the infrastructure are also monitored by the appropriate sensors. That is, the IDS is aware about the actual endpoints and their state. This is being used to enforce policies on this kind of devices.

2.2 Scenarios

There are four relevant scenarios for this thesis which are based on the ESUKOM [21] research project. The aim of the ESUKOM project was to develop a solution for integrating smartphones into business and company networks. Due to this, all four scenarios are based on real world use cases provided by the companies which participated within the ESUKOM project. Each of the scenario expresses one or more demands an actual company provided to the ESUKOM project. Furthermore, all scenarios are based on the reference infrastructure which was also agreed upon within the ESUKOM project. Both, the reference infrastructure and the relevant scenarios are results of the requirement analysis within the ESUKOM project. In particular, the first scenario deals with collecting data from a smartphone in a trustworthy manner. The second scenario emphasises the need for a policy-based decision making process which includes trust measures. The third scenario shows the use case of providing certain services of the infrastructure based on trust and the fourth scenario shows the special case of a forensic evidence collection.

2.2.1 Scenario I: Trustworthy Data Collection for Smartphones

By using smartphones within a company's infrastructure, they need to be included into the network's policy. That is, smartphones are supposed to be controllable on the same level as other infrastructural components are. To do so, the reference infrastructure's IDS can be used. This is possible, as it allows to evaluate certain properties of the smartphone against the network's policy, i.e. it is acting as a decision making component. This is done by collecting data that describes the smartphone and evaluating this data using the IDS. In detail, the describing data is collected by an amount of sensors located either on the smartphone directly or somewhere on a component within the infrastructure. If the smartphone changes certain states which are measured, the sensors will provide the appropriate data describing this change, allowing the IDS to always operate on the last measured state concerning the smartphone. As already stated, the sensors may be located on the smartphone as well as on other devices. Due to the devices being used and deployed in different ways, their security level is highly varying. The smartphone itself may be exposed to more threats in terms of compromise than an infrastructure component like the DHCP server as it is more likely that the smartphone is being exploited due to its usage profile. As the sensors are located on the device, their security level is also highly varying. Furthermore, this level is not only variable in terms of the device the sensor is placed on but also in terms of time. A device may be compromised at one point and starting to act malicious. This malicious behaviour may also apply for the sensor itself, resulting in false measurements. To counter this, the IDS must be able to receive the data from the sensors in a trustworthy manner. This would allow the IDS to distinguish between the trustworthiness of such describing data. Decisions which are made by the IDS would therefore be based on this trustworthiness, thus the IDS treats describing data in accordance to their trustworthiness. This allows to use arbitrary sensory sources as the IDS can decide on its own about the trustworthiness of the data collected by these sensors.

Example Use Case

Given a smartphone which is used in arbitrary situations and for unspecified tasks. The user of the smartphone, which is also an employee of the company, installs certain apps on the device, which may be either malicious or unwanted. A malicious app could for example start to send a high amount of SMS and thus generates a lot of costs for the user or the company. Besides an app being specifically malicious, there may also be apps installed on the device which are considered unwanted from the company's infrastructure point of view. That is, if the the smartphone is taken into the infrastructure of the company the user is working for, the infrastructure needs to react. An example for such an app would be a program allowing to stream a live feed of the smartphone's camera into the internet. This may be critical under certain conditions, for example if the smartphone is located within a classified meeting situation. To detect the apps currently present on the smartphone, data are collected on the smartphone. This data is then used by the IDS to simply perform a signature-based matching of the installed apps against a blacklist the company holds. If a malicious or unwanted app is detected, the user of the smartphone is informed and the smartphone's access to the infrastructure is denied. In this case, the IDS needs to verify the trustworthiness of the collected data. This is necessary to prevent false positives as well as false negatives. A false positive would happen if the IDS denies the smartphone's access although no particular app is present on the phone. The other case, a false negative where no enforcement actions are taken, is even more critical: the IDS does not detect the app which is actually present on the device. To prevent this, the IDS needs to take care about the trustworthiness of the data collected. Thus it is able to rate the data and improve their detection rate. Furthermore, if the sensor on the smartphone itself is compromised, the IDS detects this as well and takes further actions.

2.2.2 Scenario II: Trust-based Policy Enforcement

As already stated in the first scenario (2.2.1), by using smartphones in the context of a company's environment, they have to be integrated in the policy as well. The IDS system used within the reference infrastructure is being controlled by a policy which defines reactions to certain events. In detail, the policy that is used for the IDS holds information about the contextual parts, like time or location, and signature and anomaly parts. These building blocks define the reaction to certain types of data and their measured values. As the last scenario showed, data should be collectable in a trustworthy manner. That is, the IDS can distinguish between data in terms of its trustworthiness. Given this and the policy situation just described, there is a need for controlling trust related aspects within this policy. Due to this, the policy must be able to hold trust related elements in addition to the parts mentioned above. Having this, if the IDS handles a certain amount of data, it also checks this data's trustworthiness as it is defined in the policy. This may be on the one side by evaluating the trustworthiness in a contextual mean, thus the IDS simply processes only trustworthy data. On the other side, the IDS must also be able to use the data's trustworthiness within their policy for signature and anomaly detection. Having this, the party which is responsible for defining the policy is able to define arbitrary trust-related cases within the policy. Without this, there is no way of directly using the possibility of a trustworthy data collection.

Example Use Case

If a smartphone tries to access the private zone of the network, its state is monitored by the appropriate sensor residing on the device. This sensor collects the appropriate data from the smartphone and provides it to the IDS. The IDS holds a policy, which defines certain conditions where the smartphone should be isolated from network access. Besides conditions that are based on actual data values, like the detection of malicious apps on the device, there are also conditions which are based on the trustworthiness of the data which are collected from the smartphone. If the smartphone now starts to act weird, i.e. sends implausible data, in terms of the data collected, the IDS recognises this and lowers the trustworthiness of the smartphone. At a certain point, this results in a situation where one of the trust-based policy rules matches. As written above, this rule implies, that the smartphone needs to be investigated further and therefore being isolated in terms of network access.

2.2.3 Scenario III: Context-related Service Provisioning

With the smartphone being able to access the services provided by the network and the components located within the private zone, there is a need to distinguish between different contexts the smartphone can have. That given, there need to be an authorisation process which is based on the smartphones state as well as on the context of the smartphone. The reference infrastructure does also provide means to realise this. The authorisation is first done by evaluating certain data of the smartphones, such like apps or the smartphone's operation system. This first step belongs to the last mentioned scenario, as it is simply a more detailed version of the policy-based enforcement. The second step, which operates on the smartphone's context, forms this scenario, which is different from the scenario above. There are three types of contextual parameters which are going to be used to authorise the smartphone in terms of service access. In this case, authorisation refers to providing the service access-based on the contextual parameters of the smartphone. The three types applicable are time related context, location related context and trust related context. As the name clearly expresses, time related describes the time the service access was measured by a sensor. This sensor may be either on the service or on the smartphone. If a service should only be accessed to a certain time, this context is evaluated. Furthermore, to allow a real provisioning, a service request may be answered differently depending on the time of the request. The second type, location, allows to base access decisions and the response behaviour upon the device's location. Equally to the time, a service might only be accessed if the device is located appropriately. The last type, allow to base the service access onto the trustworthiness of the collected data. That is, the service should only answer requests, if the device is in a trusted shape. Furthermore, there may be different reactions of the service-based on different states of trustworthiness.

From the technical perspective, the IDS is able to correlate directly on the contextual parameters and base decisions on this correlation.

Example Use Case

When accessing a service by a smartphone, the smartphone must be connected to the network. Due to this, there should be already a certain amount of collected data from the smartphone. To judge about the smartphone's trustworthiness, the IDS is used to evaluate these collected data. If there are data with values differing from the expected values, the IDS lowers the trustworthiness of the smartphone. If the smartphone is now actually accessing the service, the service uses the infrastructure mechanisms and requests an access decision from the IDS (i.e. from the logical point of view, the IDS may also deny access based on its policy). As the trustworthiness of the smartphone was lowered due to the suspicious data collected by the sensor on the device, the IDS can correlate on the trust context of the access request. If the context is below a predefined value inside the policy, the IDS denies the smartphone the access to the service. The same way can be used to check if the request was made within a valid time, e.g. the company's working hours or from the right location, like a location somewhere on the company's property.

2.2.4 Scenario IV: Secure Evidence

Secure evidence describes the scenario, where it is necessary to generate a proof of an action or incident (cf. [23, 24]). Usually, such a proof is usable at a legal court. This type of scenario is not bound to a special type of incident or action. This means, each time an operation is done within the network, a proof could be generated. While there are a lot of actions where no proof will be needed, such as an allowed access to a resource or a service provided by the network, there are also critical operations. Those critical operations are usually some kind of an incident. Examples are a network-based attack, resulting in data theft or a denial of service attack. Such incidents usually result in legal actions to be taken, which require a proof of the incident. The so-called Evidence Generator (EG) is a special component which extends the reference infrastructure and resides inside the network's private zone. The EG takes responsibility for generating the required proof of an action. As the EG is also part of the network it may be involved in an incident. However, as the EG is defined as trustworthy, this special case needs to be treated separately and is therefore out of scope for this particular scenario. As defined above, to fulfil the needs when incorporating a smartphone into a company's network, the reference infrastructure with the IDS and its sensors is used. Due to this, the EG connects to the IDS, retrieving the collected data on the IDS and generating the required proof. In detail, the EG takes current sensor values as well as historic values and bundles them together into the appropriate proof. The proof consist therefore of the current state of the network as well as the changes which leaded to this state. The EG provides further operations, which use the proof. As explained, it is responsible for requesting and retrieving the proof of an action or an incident. Besides that, it verifies the authenticity and integrity of the proof as well as attesting the integrity state of an endpoint within the network at the time the proof was generated. Furthermore, it secures the non-repudiation of the proof itself as well as of the attested integrity states and, in some cases, secures the confidentiality of the proof and the integrity states. Considering this, there are several points which need to be addressed in order to achieve a court-proof evidence generation. As the EG is solely responsible for providing verified and proven data and the methodology of collecting the information is unique to each incident or action, the proof provided by the EG must be considered as true. That is, the evidence generation needs to be done in a trustworthy manner, i.e. the trustworthiness of the collected data needs to be taken into account. Besides that, the proof provided by the EG needs to be stored. This process has to ensure, that the proof is not altered while handling it and that the trustworthiness of the EG can be determined. If the EG is not treated as trustworthy, the proof is unusable.

Example Use Case

If the IDS detects a severe breach of its policy, i.e. a huge amount of signature and anomaly conditions have been triggered, it may instruct the EG to create a forensic snapshot of the current state of the infrastructure. The EG then starts to generate the proof by accessing the IDS and retrieves a current set of collected data from it. To get a most accurate picture of the situation, the EG would also request all historic information available. By receiving this information, it can then create the actual proof. In addition to the data collected by the IDS using the appropriate sensors, there may be more information included, like attestation information of particular devices within the infrastructure. After the proof is created, it is provided for further usage.

2.3 Requirements

Based on the scenarios described above, six most relevant requirements are derived that are addressed in this thesis:

R1: Trust specification, calculation and evaluation This requirement describes the ability to specify, to derive and calculate trust for collected data. It is mainly derived from scenario 2.2.1 which demands a mechanism that allows the IDS (i.e., the decision making system) to evaluate the collected data in terms of their trustworthiness. To do so, it is (1) necessary to specify trust, i.e. derive trust and (2) to use the derived trust within an overall calculation which provides a trust measure per data. Given the reference infrastructure, to fulfil this requirement, there needs to be a trust evaluation method (3) for each chunk of data collected and handled by the IDS. In order to provide this, a trust derivation technique must be defined which does not only include the components handling the data (e.g. the smartphones) but also the communication of the data throughout the system. This technique addresses (1), thus providing a basic measure of trust. Based on this derivation, a calculation approach can be defined which eventually provides an evaluable trust value for each chunk of collected data and therefore addresses (2). Besides this, the component which carries out the decisions made by the IDS needs also to be aware of the trustworthiness of this decision. This is necessary to address (3), thus allowing the IDS a trust-based decision making process in the infrastructure. In addition to the directly expressed demand in the first scenario, all other scenarios require a solution as well. This is due to the problem, that every piece of data which is being used for decision making may be untrustworthy. In detail, if the data is untrustworthy and there is no way of recognising this, potential unwanted results may be the case.

R2: Trust history The IDS within the reference infrastructure allows to store the actual collected data and their current sensor values. They are updated each time a new measurement value is received from a sensor. In its current shape, the IDS does not store the old (i.e. updated) values of the sensors as it operates event triggered. Additionally, as there is currently no trust-based evaluation done by the IDS, trust values are also not stored. Scenario 2.2.4 showed the situation, where the infrastructure's current state is pictured. Furthermore, it also included historic data values along with their trust state within this picture. Due to this, there is a demand for a storage, i.e. a history, which stores the old values. In addition to this, the trustworthiness of the data is changing from time to time and from update to update. Due to this, the history must not only include the data values but also their trustworthiness at the time this value was currently valid. Besides scenario 2.2.4, this history mechanism is also required by scenario 2.2.2. This is due to the correlation of trust values which is only possible if not only the actual trustworthiness is known but instead a history of the trustworthiness. To fulfil this requirement, there needs to be a way to provide both, a data storage and based on this, a history of trustworthiness.

R3: Trust-based correlation The IDS, which is provided by the reference infrastructure for the task of deriving decisions, is based on collected sensor data. More in detail, it uses the value of the data which expresses a measurement value from the sensor. Scenario 2.2.2 as well as 2.2.3 demand the possibility to not only use the data's value (i.e. the sensor's measurement) but also the trustworthiness of this data for correlation purposes. To derive a decision which is based both on the data's value as well as on the data's trustworthiness, there needs to be a mechanism to allow a direct correlation on the trustworthiness. That is, the IDS is able to not only include the data's trustworthiness into its decision making process, which would be possible by fulfilling R1, but to solely base the decision upon the data's trustworthiness instead of the actual measurement value. Given the previous requirement which demands a storage of data and their trustworthiness, it is possible to reason upon the trend of the trustworthiness. This allows for more sophisticated tasks, like a trust-based enforcement of devices.

R4: Extending policies with trust This requirement addresses the problem of using the IDS' policy for trust-based decision making. In short, fulfilling this requirement allows to extend policies with trust. Besides the first and the last scenario, all other scenarios demand an integration of the trustworthiness into the policies which are used by the IDS. Without this trust extension, there is no possibility of addressing the trustworthiness from the policy's point of view (i.e. using trust within a specific rule). Given the second scenario, the policy-based enforcement which should be based on trustworthiness, there must be a way of expressing the conditions and rules which lead to the described enforcement. Furthermore, providing specific services for different trustworthiness cases of the collected data is also required. Due to that, without an appropriate policy integration there may be only a static definition of cases which results in a system that cannot be configured appropriately. To solve this limitation, there needs to be an appropriate mechanism to integrate the trustworthiness into one ore more policies. In addition, there must be also a way to include trustworthiness into the already existing policy structures of the IDS. This aims to extend the IDS' policy in a way to fully support the trustworthiness of the collected sensor data.

R5: Extensibility in terms of used data and trust calculation methods Scenario 2.2.1 describes the situation where the IDS uses data from arbitrary sensory sources. As described in the first requirement, the IDS should be able to judge about the trustworthiness of these data. Given those two points, it is necessary to support arbitrary sensors and thus arbitrary mechanisms the trustworthiness is based upon. In detail, the measurement which is used to derive trust for a chunk of data must not be limited to a particular type. If the measurement is limited, there is no possibility of using new sensors within the system as no trustworthiness could be derived. Furthermore, the calculation process itself must not be limited to a certain type or algorithm. Without this flexible calculation, there is no way of using new trust measurement methods as there would be no algorithm supporting these new methods. Given these two points, it is required, that arbitrary sources may be used to derive trust and that these trust derivation is flexible in terms of the calculation method used. This allows also to exchange the calculation method based on the actual domain.

R6: Ability of seamless integration To actually use the system which is described in the scenario, there is the need to provide it in a most integrable way. That is, it is required that the system is able to be integrated into existing infrastructures. Without fulfilling this requirement, it is not possible to deploy the system within a working environment. When developing a solution for the previous requirements, it must therefore be taken care of this last requirement. By fulfilling this requirement, the system can be deployed within an already existing infrastructure in a less intrusive manner, i.e. only a minimum of existing components need to be customised in order to support the whole system.

Those six requirements need to be addressed in order to support the presented scenarios. The following chapter presents the relevant related work which has already been done and which may be used to fulfil certain parts of the requirements. Furthermore, it is distinguished between research-based approaches and technological-based approaches. All approaches are compared against the requirements above.

3 State of the Art

Contents

This chapter evaluates work which has already been done against the identified requirements. First, it gives an overview of the research centric work by summarising the most important research contributions. Second, technologies that provide certain kinds of mechanisms to fulfil the requirements are described. Summarising both parts, the assessment compares the research contributions as well as the technologies against the requirements and identifies the gap.

3.1 Research-based Approaches

First of all, research-based approaches are presented and discussed. The approaches itself are categorised within three particular types, thus the evaluation starts with more generic approaches and finishes with approaches that address specific problems.

3.1.1 Intrusion Detection

Taking the reference infrastructure and the scenarios which operate upon this infrastructure, approaches for intrusion detection are analysed. Based on this analysis, further relevant categories, in particular work which has been done in the field of different intrusion detection types and trust-based approaches, are analysed. A basic definition and different types of intrusion detection systems and their working principles can be found in [25, 26, 27, 28, 29]. Given the categorisation of intrusion detection made within these publications, there is one approach which can be characterised as a network-based intrusion detection system (cf. [30]) that is tailored for use in smartphone-enabled networks and infrastructures. Due to its capability of handling smartphones and the combination of several different research fields, this approach is analysed first.

CADS

CADS abbreviates Context related Anomaly and Signature Detection for smartphones and is described in [18, 19, 2]. It allows to collect so-called Features, which represent a particular property and their measured value of a device. In detail, the Feature consists of the following relevant elements.

- A Feature id allows to uniquely identify the Feature. Note that two Features may have the same id if they measure the same property for the same device. This would be the case, if for example the Feature that has been measured first will be updated with a new value that represents this measurement.

- The value of the Feature (Feature value) contains the actual measurement value. E.g., if the Feature holds the IP address of a device, this IP address would be stored as measured value within this field.

- In addition to this two fields, there is another field which addresses the Feature's context. Most important, the time stamp of the measurement is stored as a time context in this field. Besides time-related entries, the Feature could also contain another context types, like the location. The CADS approach does not limit this field to a certain type of context.

As already stated, the Feature expresses a certain property of the measured device. While it describes one particular device, the collection may not only be done by and on the device itself but by arbitrary devices which are located in the same network.

Architecture The architecture of CADS (figure 3.1) is formed by four components: the Feature Collector, the Feature Provider, the Feature Consumer and the Correlation Engine. A detailed explanation of the components is given in the following.

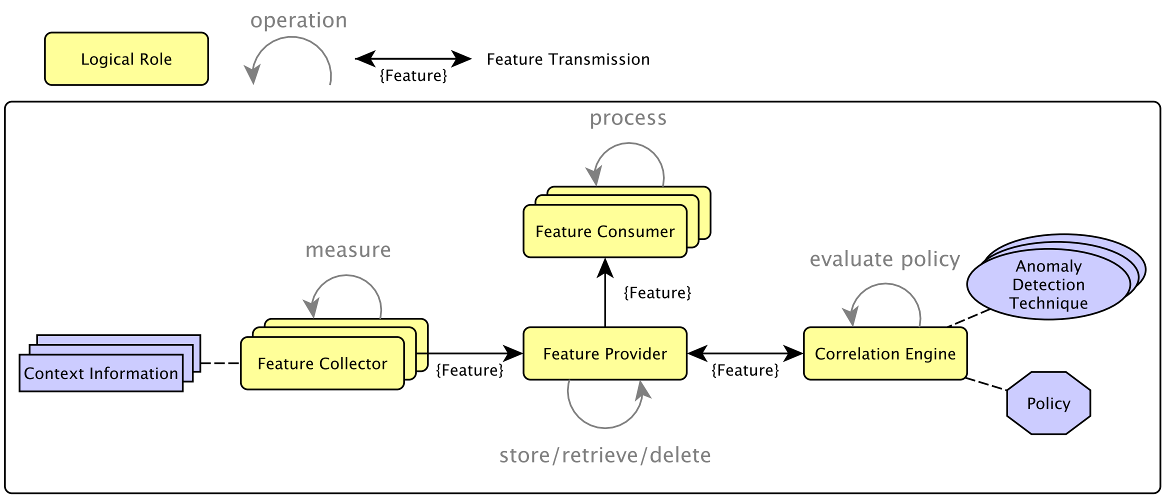


Figure 3.1: CADS architecture as defined in [2].

Feature Collector The Feature Collector is responsible for collecting (measuring) and aggregating the required Features. There may be an arbitrary amount of Feature Collectors on different platforms and devices in the network.

Feature Provider The collected Features are sent to the Feature Provider which stores them. Furthermore, a component may not only request storage of Features but also the request to retrieve a certain Feature and the request to delete a no longer used Feature. The architecture demands that there is only one central Feature Provider as part of the network.

Correlation Engine A Correlation Engine is responsible for accessing the Features and correlating them to apply anomaly and signature detection algorithms. This is done by evaluating a defined policy and the appropriate Features. Prior to its actual use in the environment, there may be a training phase. This training phase allows the Correlation Engine to learn the necessary reference values used for anomaly detection means. The CADS approach does not define the actual method used to perform this training phase and allows to use arbitrary methods.

Feature Consumer Decisions made by the Correlation Engine are propagated to the Feature Consumer. Propagate refers to the process of sending the decisions as Features to the Feature Provider followed by the Feature Consumer accessing these decisions on the Provider. The Correlation Engine therefore uses Features to store decisions (i.e. decisions are stored as value of the Feature). These Features are received by the Feature Consumer (after they where transmitted to the Feature Provider) and used to trigger and process further actions.

Based on this, a general communication flow within the CADS system can be expressed:

- The Feature Collector sends the Feature to the Feature Provider. The Feature includes measured values, which are used to describe aspects of the smartphone. For example, the installed apps on the smartphone may be expressed by a Feature. After the Feature is received on the side of the Feature Provider, it is then stored within the Feature Provider's database, thus keeping it ready for further usage.

- At any time, the Feature is then further transmitted to the Correlation Engine. That is, the stored Feature is retrieved from the Feature Provider's database and then send to the Correlation Engine. Given the example from above, the stored app measurement would be given to the Correlation Engine which then evaluates the Feature against its policy.

- After the Correlation Engine has finished processing of this Feature, it may create another one which is then transmitted back to the Feature Provider. The Feature Provider takes this Feature and stores it again within its local database. For example, the Correlation Engine would make an enforcement decision based on the app Feature explained above. This enforcement decision would be provided as new Feature to the Feature Provider.

- A Feature Consumer can then use this Feature for further actions. That is, the Feature Provider retrieves the stored Feature out of its local database and sends it to the Feature Consumer. Within the named example, the enforcement decision would be provided to an appropriate Feature Consumer, e.g. a firewall. This firewall would then perform the enforcement.

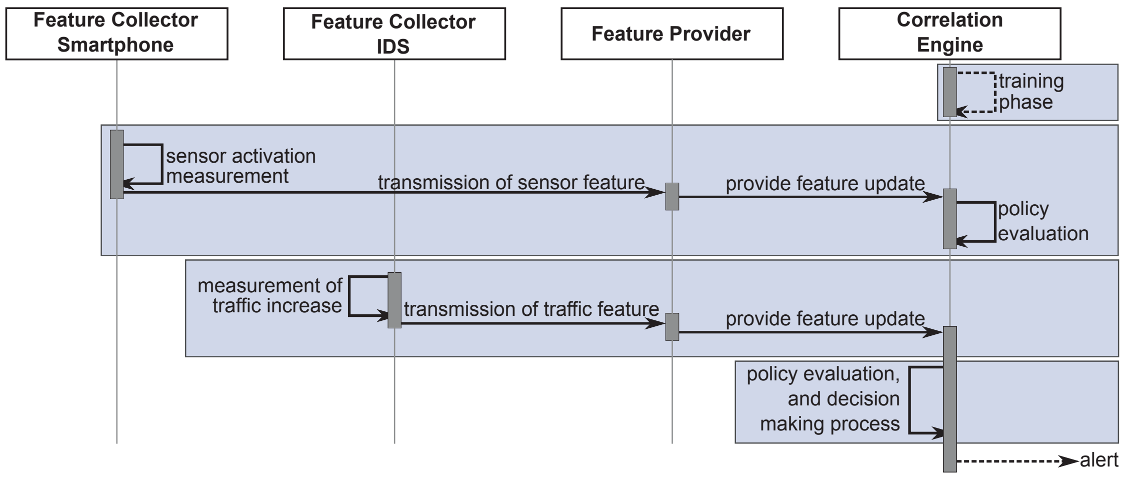

Feature Collector Smartphone
Feature Collector IDS
Feature Provider
Correlation Engine
training phase
sensor activation measurement
transmission of sensor feature
provide feature update
policy evaluation
measurement of traffic increase
transmission of traffic feature
provide feature update
policy evaluation, and decision making process
alert

Figure 3.2: Example Flow given by [19].

After the optional training phase, the processing of the Feature starts with the Feature Collector on the smartphone. In this example case (Figure 3.2), a sensor activation is measured. The sensor may be the smartphone's microphone or the camera within this example. The Feature Collector takes this Feature and transmits it to the Feature Provider, which provides the Feature further to the Correlation Engine. The Correlation Engine is then able to check the Feature's value against its policy and optional against the learned reference values. As the condition for raising an alert is not fulfilled yet, the Correlation Engine does not react on the Feature in this example. This changes when the second Feature Collector which is located on the IDS within the network measures an increase of the smartphone's traffic. This measured Feature is provided to the Correlation Engine again by using the Feature Provider. Having both Features, the sensor activation as well as the increase of traffic, the Correlation Engine now raises an alert as both Feature values together violate the policy in this example. This alert is given to the Feature Provider, which sends it to the firewall system. The firewall, which acts as a Feature Consumer, uses the Feature and isolates the smartphone, thus effectively disabling the network access for the smartphone. This simple example shows how the infrastructure and the appropriate CADS components are used.

Summarising this, CADS is capable of performing both, misuse and anomaly detection. Misuse detection characterises the ability to detect static properties of an entity, i.e. unwanted configuration like installed apps that are malicious for smartphones. Misuse detection in the field of intrusion detection has been widely researched, details can be found in [31, 32, 33]. In contrast to static properties of a system, anomaly detection is used to recognise behavioural-based actions. That is, CADS is able to detect changes in the way a particular smartphone devices behaves e.g., a sudden increase of traffic which may be a hint for a data exfiltration taking place. As with misuse detection, the technique of anomaly detection is also one basic ability of intrusion detection systems and has been researched intensively. More information about detailed mechanisms can be found in [34, 35, 36]. The architecture of the CADS system can be interpreted as a special version of a hierarchical intrusion detection system (cf. [37, 38]) due to the Correlation Engine as decision making component operates on high level information. The sensors, i.e. the Feature Collectors, abstract particular measurements and decide on their own when to publish a high level alert to the Correlation Engine. Due to this, CADS cannot be seen as a conservative acting centralised intrusion detection system. In addition, although CADS is being considered as a distributed intrusion detection system it is not considered to be peer to peer based with isolated nodes as the final decision making is centralised. Node and peer to peer based intrusion detection relies on a consensus of several nodes which form a decision together (cf. [39, 40]). Comparing the approach against the requirements, it is clearly to see that CADS does not provide a solution to fulfil R1. However, as CADS provides a Feature-based storage, correlation mechanisms and a policy, R2 to R4 can be considered as partially fulfilled. The last two requirements (R5 and R6) have also been considered when developing CADS, thus R5 is partially fulfilled as trust is missing but R6 is completely fulfilled.

Other approaches

Besides the categorisation of centralised or distributed approaches, intrusion detection can also be categorised into host-based, network-based and hybrid approaches approaches. [25, 26, 27, 28, 29, 30]. Host-based intrusion detection systems are used to monitor a single platform (i.e. a host), usually by measuring certain aspects of the platform. In contrast to this, network-based intrusion detection system use a network-based monitoring approach, for example by analysing the traffic within this network. Both types provide different measures to detect certain properties, like malware or ongoing attacks. A hybrid approach combines both types into a single system, e.g. by using sensors to monitor the traffic and host-based techniques to monitor a system's state. However, as the scenarios require collection and correlation of data, the categorisation used for this chapter is based on the system being either centralised or distributed. Both categories may consist of a host-based, a network-based or a hybrid approach. That is, the categorisation used here can be considered orthogonal to the system-based classification.

In [41] Bidou et al. describe an architectural model based upon the concept of having a centralised component which evaluates data to detect network intrusion or other threats. Furthermore, the authors propose an implementation of the centralised component which is commonly called security operation centre (SOC), thus calling their implementation SOCBox. It consist of several sub-systems, in particular a module to gather data from arbitrary sources, an integrated correlation engine as well as several databases (e.g. a vulnerability database). The overall architecture is able to gather event-based data from sources like host-based intrusion detection systems and evaluate it against database entries or perform correlation tasks upon the data. Given that, requirements R2 to R4 are addressed to some extent. While this approach is usable in terms of detecting intrusions or threats, there are two potential problems. As the SOCBox acts as a centralised component it may likely be a single point of failure. This first issue is also identified by the authors and resolved in an improved version of the architecture (see below). Another problem is the lack of trustworthiness in the data sources. There is no possibility for the SOCBox to distinguish the received data between valuable or invaluable in terms of trust, thus R1 cannot be fulfilled by this approach. As the approach is rather static, it provides a limited extensibility (R5) and is difficult to integrate (R6).

Ganame et al. propose an improved version of the architecture described above, solving the problem of isolated intrusion detection systems. They describe the overall idea in [42], which is based on gathering network relevant information to recognise attacks. Furthermore the authors distinguish between wide networks and smaller network segments, thus inventing certain layers of detection within their approach. The lowest layer is formed by localised detection engines, collecting data from one isolated network segment. The result of this detection is then distributed one layer upwards to a so-called global analyser, which has a broader view of the overall network. The authors call this architecture a distributed security operation centre (dsoc) which is an improvement of their previous approach which uses a centralised operation centre (cf. [41]). Although this improvement offers a solution to the single point of failure problem, proposed by the authors in [41], and allows for more extensibility (R5) it still has some possible drawbacks. Like other approaches, which may be either centralised or distributed, there is no process of verifying the trustworthiness of the collected information (R1). That is, the local detection as well as the global detection engine have no possibility to select received information by means of their trustworthiness. This could possible result within a situation where a rogue data collector publishes malicious information, thus triggering unwanted actions.

A good example for countering threats of data exfiltration from inside the network is presented by Ramachandran et al. [43]. The approach presented by the authors is based on collection information about the networks standard behaviour, learning this behaviour and deriving a behavioural model out of the learned information. This allows to detect unusual behaviour and raising alerts. More in detail, the authors aim is to prevent data exfiltration out of the network by profiling each host within the network. To learn a hosts profile, data is collected by using SNMP-based data exchange and calculation based on kernel density estimation and correlation coefficients (R2 and R3). Although the authors show that their approach might be promising, there are still some open questions regarding this approach. Besides others like scalability and applicability (R5 and R6) the main question refers to the trustworthiness of the collected data. While the model itself is sound, it is rather limited in terms of verifying the data sources trustworthiness, and thus the data itself. In particular, this might be a problem when trying to use a behaviour-based model to detect certain kinds of insider attacks. Comparing this against the requirements, particularly against R1, the approach offers no solution for trust-based reasoning.

More work has been done in the area of correlating data in order to perform a detection of anomalies or misuse. In [44] Cullingford shows an approach, where multiple detection engines work together. Aim of this approach is to minimise the false alarm rate. Although the approach performs well in the scenarios described, the author does not include the problem of sensors being untrustworthy. Furthermore, by combining several decision making systems to gain a high level decision, the problem becomes even more critical.

3.1.2 Trust-enhanced Intrusion Detection Systems

This section summarises work which has been done in the field of intrusion detections systems that incorporate a trust evaluation.

Zhou et al. give an in-depth overview of the current concepts of intrusion detection systems in [45]. The summary is given on the background of large scale attacks which aim on multiple networks but is also applicable for single network IDS concepts. The authors distinguish between the already mentioned three general types of intrusion detection systems: a centralised approach, the hierarchical approach and a fully distributed approach. The centralised approach is based on a single component gathering information by their sensors in one single point. Furthermore, the sensors provide raw data without any form of pre-correlation and selection. The hierarchical approach extends this idea by giving the sensors the ability to filter data and aggregate data. This allows to form a hierarchy of sensors as well as correlation components which ends on the highest level in one single component. The CADS system introduced in the previous section can be assigned to this group of approaches as it has a high level centralised decision making component but is not based on raw data. The third group of approaches, which is currently widely discussed in the field of intrusion detection is fully decentralised. That is, all of the sensors perform a special kind of decision making which consists of some kind of negotiation between the sensors, which is not based on one single centralised component. The authors also summarise the benefits of each approach as well as the disadvantages. Although they address problems like developing a single point of failure or the pre-selection of data, there is no detailed view onto the problem of the sensor's trustworthiness (R1). Other approaches in the field of intrusion detection include this issue as a problem and try to develop solutions for it. In addition, Leckie et al. have also deployed some of the proposed concepts for different tasks. In [46] the authors describe an intrusion detection system working in a collaborative manner based on a peer to peer infrastructure for the detection of phishing domains.

Cuppens et al. introduce the CRIM intrusion detection system in [47]. The authors point out, that low level alerts may not be usable to determine if an intrusion is taking place. Thus, the false alarm rate of such systems that use conventional low level alarms are rather high. Due to this, the CRIM system combines alerts to a higher level. This is done by several modules. The first module, the alert base receives alerts and stores them, thus providing a data storage (R2). The alerts are received by the particular intrusion detection components sending their results using the IDMEF [48] message format in order to exchange the low level results. The second module, the alert clustering, takes the alerts and clusters them into appropriate groups, thus effectively generating higher level alerts. Those groups, i.e. clusters, are then merged by the next module. The result of this alert merging module are so-called global alerts which can now be used by a correlation engine (R3). This correlation engine, represented by the module called alert correlation, produces so-called candidate plans which are checked in the last module against the original intention. If the match is successful, a reaction may occur. The authors have tested their approach and could prove, that the use of this system significantly reduces the false alarm rate for the attacks investigated. Additionally, the approach is being presented as integrable and extensible in terms of data sources (R5 and R6). However, as trust is not being considered explicitly, the basic requirement R1 cannot be fulfilled.

In [49] Janakiraman et al. propose early, that scalable and reliable network intrusion detections mechanisms are needed. In detail, the authors focus on the distributed collection of data in conjunction with a distributed processing of this collected data, thus addressing requirement R2 and R3 without the trust consideration. This negates the identified problems of single point of failure architectures. Besides that, Indra, an intrusion detection tool developed by the authors using a peer to peer (p2p) approach for intrusion detection is presented. Indra allows to share information between peers, thus allowing to identify attacks which are carried out on several machines independently. Within the Indra approach, the authors identify several difficulties. One of those difficulties reflects the problem of information trustworthiness between peers. That is, as Indra is peer to peer based and has no central trust authority which would be able to evaluate the information in terms of their trustworthiness, a decentralised trust approach (cf. [50]) needs to be applied. As an example for such a decentralised approach, the authors suggest the so-called web of trust approach used within PGP [51]. Although this is an early approach, it already shows the need for trust considerations between data collectors (R1).

Keromytis et al. present another collaborative approach, called SABER, in [52]. Besides other, the SABER architecture is based on intrusion and anomaly detection components and a high level coordination infrastructure. The intrusion and anomaly detection system work by using two methods. The first one is a so-called surveillance detection. It monitors the network traffic and evaluates it against known bad indications, thus performing a misuse detection based on signatures. The second method, which is used to detect anomalies, is based on training of normal behaviour. That is, after the behaviour has been learned, abnormal behaviour (like traffic) creates an alert (cf. [53] for misuse and anomaly detection methods). SABER uses a publish/subscribe system to inform its components about events happening in the network or decisions made by the high level control instance. Furthermore, an event-based correlation system is used on the side of the high level control instance. This allows to quickly react on multiple events, like a DoS attack. The authors have tested their work in several scenarios, with an overall good outcome. While SABER may be used as a general framework to build up collaborative system to prevent attacks, there is one important point missing. The authors do not address the point of attacking the SABER system itself. I.e. there is no way to determine if a SABER's component might be compromised in any form. That is, R1 is not fulfilled by the approach. Furthermore, the system operates on events, thus there is no dedicated storage for collected data (R2). However, SABER provides an event-based correlation (R3) and is designed to be extensible (R6) in terms of sources.

As the problems of using data resources which may not be trustworthy were identified, a lot of work has been published within this area. One of the earlier approaches is presented by Duma et al. in [54]. The authors point out, that collaborative intrusion detection system approaches sound promising with limitations due to their trust issues, thus emphasising R1. That is, although there are promising approaches, they all make strong assumptions about the trustworthiness of their data sources. This results in a situation, where a data source might provide false information (intentionally or unintentionally) which leads to a reaction which was not intended. As the authors have analysed several general approaches which are not limited to collaborative working intrusion detection systems but nearly all collaborative working systems which collect information of data sources, the problem of trustworthiness of data is also not limited to intrusion detection approaches. To circumvent this problem, the authors suggest a trust management system, which pre-assigns trust values to all peers and recalculates them based on trustworthiness given by other peers. That is, they provide a solution for R1. However, they are only using pre-assigned values which only allow to calculate an initial trust value. If the situation changes, the proposed system is unable to react on this change. Due to this, R1 must be considered as only partial fulfilled. As they propose a collaborative approach, R2 and R3 are also fulfilled to some extent. Extensibility and integrability are not considered by their approach.

In [55], Fung et al. present an approach to incorporate a trust management system into a collaborative working intrusion detection system which can be seen as an improvement of the approach of Duma [54] et al. It is based on communicating test messages between peers, which actually are host-based intrusion detection systems working together (cf. [56]), and reasoning based on the answers of this message about a peers trustworthiness. A more refined version of this framework, also presented by Fung et al. is shown in [57]. Within this paper, the authors propose an approach for a trust-based framework within a collaborative working intrusion detection system. Furthermore, the aim of the framework is a peer to peer based intrusion detection architecture, such as Indra [49]. However, the authors do not address trust issues within its intrusion detection system which relies on a single headed control instance. That is, the approach presented by the authors focuses on the trustworthiness of distributed nodes, which represent host-based intrusion detection systems and which are communicating via a peer to peer layer. To reason about a nodes trustworthiness, the requesting node sends a so-called test message to the node which is under investigation. With the test message being a request to rank an alert, the node which is to be investigated answers with a so-called severity. Furthermore, the node to be investigated is not able to distinguish between a real alert ranking request and the test message. As the requesting nodes knows the real severity of the alert, it can judge about the trustworthiness of the other node by verifying the received answer. This process is randomly but periodically often performed. Robustness studies made by the authors show an positive impact by using their model. Another version of the approach is presented in [58]. Like the previously explained version, the approach is also based on the trust definition of a Bayesian network (cf. section 3.1.3 and [59]). In difference to the previous versions of their approach, the authors propose a dirichlet-based distribution (cf. [60]) model to evaluate the test message exchange between particular nodes. Although this improves the alarm rates, as the paper clearly shows, the problems of founding trust on test message exchange, which has been described in detail above, remains the same. Due to this, R1 can be considered partially fulfilled with the limitation that trust is only being derived by the test message procedure which makes the approach difficult to control (R4) and inextensible (R5).

Another research field where intrusion detection plays an important role can be found in the area of cloud computing. In [61] Joshi et al. present an approach for carrying out an actual intrusion detection within these kind of environments. Their approach is based on the use of the services and resource the cloud provides to determine about intrusion detection. Furthermore, they do not only provide detection mechanisms but also prevention methods. Although the approach itself might be promising, the authors only deal in a very limited manner about the problem of deriving trust. That is, as they make use of inter-domain cloud providers, there must be a mechanism to put trust on these different domains, thus assigning trust onto the service providers. The authors do not propose a particular solution but emphasise, that this problem need to be dealt with.

Intrusion detection is also important in the already mentioned area of wireless ad hoc networks. Due to this, there has also been a lot of work on establishing efficient ways of performing intrusion detection in such environments. [62] gives an overview about recent research results. The approaches which have been developed, can be divided into two general groups: approaches that deal with trust and approaches which imply their sensors to be trustworthy, thus not explicitly addressing trust. Particular early approaches of the second group which can be characterised as building blocks of these research area can be found in [63, 64, 65, 66]. A selection of more sophisticated approaches are described in [67, 68, 69, 70]. The first group, which incorporates the measurement of the sensors used, has also been worked on. Some of the approaches introduced are explained in the following.

The first approach belonging to this group, called TIDS, is presented by Deb and Chaki in [71]. Trust is derived from special messages exchanged between the nodes being part of the network. In detail, each node is able to calculate a trust measurement for each of its neighbouring node. If the trust derived becomes low in terms of a specified threshold, the intrusion detection system can react to this event (R3). Furthermore, the trust calculation is not only used for intrusion detection purposes but also to determine a safe route for messaging through the network. As the number of potential nodes is not limited but only nodes may be integrated, R6 can be considered fulfilled partially. The approach presents a reasonable but not extensible method for deriving and calculating trust. However, the evaluation method is rather limited as it statically defined to only evaluate it for determine the route. Thus, R1 is partially fulfilled by the approach while R5 cannot be fulfilled.

Yeom and Park propose the idea to use a special kind of agents to allow an efficient intrusion detection within the field of mobile ad hoc networks. Their paper [72] uses the approach of modelling the intrusion detection system based on the human immunological system. That is, they introduce the named special mobile agents which act like a human blood cell. The agent is therefore able to distinguish between actions which are considered to belong to the normal behaviour of the system and actions which are uncommon, thus being considered as intrusion. The differences of these two types of actions are based on the humans blood cell ability to distinguish between "self" and "non-self". Although the approach is promising in terms of a distributed intrusion detection, there is no way to verify integrity of such an agent (R1). This may be critical, even more as the authors aim to address mobile ad hoc networks which are highly dynamic, if one of the agents becomes compromised in an arbitrary manner. Due to the properties of a mobile ad hoc network, the approach is designed with integrability and extensibility (R5 and R6) in mind.

3.1.3 Trust Management and Means of Deriving Trust

A very early approach which aims more in the direction of social trust is introduced by Aberer and Despotovic in [73]. Although the discussion in this paper emphasises on peer to peer networks and their trust issues with unknown participants, the findings made are relevant. This is due to newer intrusion detection approaches commonly are working as peer to peer systems. The authors point out, that it is necessary to judge about a participants trustworthiness in order to perform more sophisticated tasks than simple file transfer. They therefore develop a concept to base the trustworthiness of a participant, i.e. of a node, on their behaviour. In detail, the behaviour is analysed thus forming a reputation of this node. This reputation can be computationally be expressed and taken for further decisions about this node, like communication with it. While the approach has been already refined, the paper shows clearly, that there is a need for deriving the trustworthiness of a node in such environments. That is, requirement R1 can be considered as fulfilled to some extent. A storage for data is not directly mentioned within the paper although the IDS bases decisions on that data (R2). Using the peer to peer approach, more nodes may be used, thus providing a good integrability (R6). However, R5, is only fulfilled in a very limited form due to the problem of only using a node's behaviour as trust derivation technique.

While not directly related to distributed environments but also on social interactions between agents, Miu et al. show the problem of deriving trust in [74] from a rather non technical point of view. It is made clear, that for example trust is not directly derivable from reputation. This is due to reputation being eventually mutual in the described scenario. Furthermore, the authors propose their own model which clearly shows the difference between trust and reputation, the definition of reputation, the definition of trust as value which is valid between two parties and the calculation approach for determining the actual trust. As already explained, the scenario of the others is rather non technical. Despite this, the concepts described are also to some extent applicable to technical environments.

Mobile ad hoc networks often require a secure routing of messages transmitted within those networks. That is due to that problem that nodes connecting to these networks may be untrustworthy and thus could compromise the message. In [75] Liu et al. propose a dynamic trust model which allows for a secure routing of such messages in mobile ad hoc networks. The authors assign a so-called trust level to each node. This trust level indicates the trustworthiness of the node and allows to select the most trustworthy path through the nodes a message should take (R1). More in detail, the trust level can take six different values, ranging from untrustworthy to very trustworthy. The trust level is calculated using a special algorithm, based on the input of an intrusion detection system each node must possess. The authors assume, that the IDS located on each node is possible to detect attacks which aim to change the integrity of the messages transmitted throughout the network. Furthermore, they also assume that the IDS on each node can detect other kinds of attacks of nodes within their radio range. Based on these findings, the trust level of the neighbouring nodes can be calculated and thus a secure path can be selected. Although this approach is promising, there are two general problems which need to be addressed: the distinction of the different trust level values needs to be mapped for each particular environment (R5) and furthermore, it must be ensured that the IDS on the node is able to successfully detect all relevant attacks (R6). In addition, work related to the proposed idea can also be found in [76].

Another approach is presented by Azzedin et al. in [77]. The aim of the authors is to develop a trust model for grid computing environments (R6) allowing to define so-called secure and trustworthy domains. To achieve this, the authors distinguish trust into so-called identity and behaviour trust. Identity trust expresses the confidentiality about the authenticity of an entity, thus is more technical-based. Behaviour trust is based on the reputation of an entity and requires more sophisticated means to be measured. Furthermore, the authors propose the use of a trust level for entities. Like in other approaches, this trust level may take pre defined values of a limited set expressing the different stages of the entity's trustworthiness. Besides the trust level, the authors also introduce the concept of direct and reputation-based (indirect) trust. Direct trust can be expressed if two entities are directly connecting while reputation-based trust is used if there is no direct connection between the two entities. To actual calculate the trust levels for an entity, the authors propose a mechanism with several intermediate stages, like a required trust level for each of the components being part of the calculation. With using this mechanism, the paper deploys an effective way to calculate the direct and reputation trust (R1) within grid environments. The part of the identity trust is being considered well known, thus the authors do not propose special means to determine this kind of trust. This can be considered as a limitation, e.g. if the identity trust calculation is being compromised in any way, the overall trust level may be false. Furthermore, no extension possibilities are discussed by the authors, thus the approach is limited to the proposed mechanism (R5).

Another area where the term trust was introduced in is described by Blaze et al. in [78]. The authors show the drawbacks of authorisation models which are identity-based. That is, instead of performing an authorisation based on the lookup of a certain identity and handling some kind of an access control list, the authors propose to use a trust engine. This trust engine is unaware of the actual identity but is able to derive trust (commonly policy-based) for the component the authorisation decision is about. Another early paper founding the area of trust management was published by nearly the same authors as [79]. Although both papers are not directly related to the requirement R1, they clearly show the need for deriving trust and provide very first approaches on how to do so.

In [80] Thomas et al. picture the problem of using identity related information for decision making in the area of federated identity management. As the concept proposes, identities are used to allow for a high scalability in terms of the described federated environments. The authors point out, that within this context, several distinguished trust domains can be defined for the service providers used in these federated environments. These service providers have to rely on identity-based authorisation information, which may be untrustworthy due to the fact that they are not necessarily calculated from the context of the service provider's domain but from the local domain where the access is originated from. That is, the service provider is unable to judge about the trustworthiness of the authentication process as it is not part of its local domain. To counter this, the authors propose to use a so-called level of trust (R1) which measures the strength of the authentication process and attaches it to the authorisation information (R2). Although this idea is not directly transferable into the specific environment presented above, the idea of measuring an authentication process in terms of its trustworthiness sounds promising at all.

Authorisation and authentication in open environments is also treated by Bhargava et al. in [81]. The authors work out, that relying only on specific credentials like passwords to perform these tasks may not be sufficient. This is due to the lack of credentials implying trustworthiness, in the authors specific scenario for a certain user. To avoid this problem, they propose the use of evidence (i.e. credentials) and trust to perform authentication and authorisation. To do this, the paper describes an enhanced role-based access control scheme, which incorporates a users trust. This trust is based on the behaviour of the user and may vary. That is, the approach provides another behaviour-based model which can be utilised to address R1. Due to the fact, that only behaviour-based trust is incorporated, R1 must be considered only partially fulfilled. Furthermore, the approach itself is limited to this technique, i.e. there is no extension possible (R5).

The distinction between trust, that is based on an identity and trust which is behaviour-based is also pointed out by Papalilo and Freisleben in [82]. Their work aims on establishing trust mechanisms in grid like environments by the use of bayesian networks. The basic idea of using bayesian networks for trust derivation was already developed in [83]. In detail, the authors define a network, which defines the relationship between identity trust, behavioural trust and their combinations, simply called trust (R1). However, with a focus on grid properties. Furthermore, they introduce methods to derive behavioural trust for an entity by basing it upon measurable features like service quality and processing speed, which is to some extent controllable (R4). Given these parts, the authors propose a method for dynamically calculating trust. This allows to determine the trust of an entity in a fine grained and flexible way, thus allowing to use the approach in different particular grid-related scenarios (R5 and R6).

The use of Bayesian networks to describe trust is also used in [59] by Hailes et al. The approach introduces a trust framework based on a bayesian formalisation. Furthermore, the formalisation made does not only distinguish between trustworthy and untrustworthy but defines a level of trust which may be arbitrary but discrete. The authors derive the overall level of trust based on the direct trust as well as on recommendation trust. In detail, the trust derivation uses several sub stages, that are stored (R2), forming an altogether value (R1). Besides the calculation itself, the framework is also capable of evolving trust.

The framework is tested again by routing messages through several nodes in a network, with some of the nodes acting maliciously. Due to this, no direct controllability is given by the approach (R4). Additionally, the trust derivation (R1) is also limited, which is a result of the missing bootstrapping procedure. In detail, the authors do not yet present a method for defining initial trust values.

Another framework which can be used for trust management in distributed environments is introduced by Sun et al. in [84]. The approach used by the authors consists of several stages. The first stage, consisting of the basic definition of trust introduces the authors interpretation of the term trust in computer networks. Based on this definition, the authors introduce their basic propagation axioms, thus forming the system their management framework should be developed in. Given these two parts, the authors analyse several attacks mounted on distributed environments, in particular in the area of routing traffic through arbitrary nodes. Using the findings made, the authors are then able to define their trust management system. It consists of five building blocks which handle the trust calculation (R1), the trust storage (R2), the trust process as well as the detection of malicious nodes (R3 partially). The actual trust derivation used is based on the HADOF approach presented in [85]. That is, the authors rely on a technique, where a node can determine if packets are dropped by another node. While the approach with its trust management itself can be considered rather promising, the authors limit their actual trust derivation by the use of HADOF (R1 and R5). While the authors proof their approach to be effective in the area of node-based ad hoc networks, it is unclear if the approach is working in other distributed environments (R6). Due to the limitation of the actual trust derivation (R5), this needs to be investigated further, although the authors propose their approach to be usable in arbitrary distributed environments.

Complex trust management plays also an important role in the area of wireless sensor networks. In [86] Bao et al. present an approach to leverage a trust-based intrusion detection system in such an environment. The trust derivation is based on two factors: so-called social trust as well as quality of service. Social trust is derived from a nodes honesty in terms of trust [87, 88, 89]. That is, factors like false self reporting and abnormal trust recommendations from nodes are used for derivation purposes. The quality of service based trust derivation uses technical measurements, like the energy consumption of a node. Based on this, the authors deploy a trust management system providing a hierarchical, trust-based intrusion detection mechanism. Furthermore, the authors add a probability mechanism to minimise the false alarm rate. While experiments made by the authors show the approach to be promising, there are still some open question. The authors themselves propose to intensify the research in the area of trust derivation, like including more techniques for measuring social trust.

Wang et al. propose the IDMTM, abbreviated for intrusion detection mechanism based on the trust model. Their paper [89] introduces the IDMTM system allowing an intrusion detection in the area of ad hoc networks. The idea of the IDMTM system is based on two parts: the usage of a so-called evidence chain and the measurement of so-called trust fluctuations (R1). The evidence chain combines several evidence of malicious behaviour for a certain node, thus it acts as basic policy (R4). That is, the evidence chain holds well known malicious node behaviour, e.g. particular types of attacks which may be carried out by a node. It can be seen as a prototype of a node combining all possible malicious behaviour into one single instance. By taking nodes and matching their behaviour against this chain, malicious nodes may be detected. This is done by defining a threshold which indicates the point, where enough parts of the chain are matched, so the node is treated as malicious. Furthermore, a node's trust fluctuation is used. Trust fluctuation expresses the change of a node's trust level. If this change is greater than a pre defined threshold (e.g. the trust level's standard deviation), the node is also being considered untrustworthy thus being malicious. Combining these two approaches, the IDMTM system is introduced. Unfortunately the approach is only usable in node-based environments (R6) and provides no extension of the trust fluctuation mechanism (R5). By using a simulated environment, the authors prove their approach to be promising in terms of detection and, more importantly, in terms of the reduction of false alarms.

Wireless networks may also be used in ad hoc vehicular applications, where they are realised as multicast networks. Chang et al. propose the use of a Markov chain-based trust model to determine a node's trustworthiness in [90]. The goal of the work is to provide a reliable authentication method which can be used for node authentication in this type of network. The authors propose to calculate a so-called trust value for a node. This trust value is based on the previously determined trust manner of the node. This trust manner is derived from the previous operations a node has performed, e.g. joining or leaving a multicast group in a trustworthy manner. Using this, each neighbouring node can derive trust, thus forming the Markov chain-based model (R1). The node with the highest trust value is then being selected as certification authority used for the actual authentication process. While the authentication itself is rather out of scope for this work, the derivation of trust must be taken into account. The authors not only propose to calculate the trust values for each node but also to distribute this values for successful selection of a certification authority. Although each node is aware of the calculated trust, the building blocks used for the derivation process are only fixed behaviour-based measurements (R5). The authors have proven their approach to be promising in the described scenario but also realise that more work is necessary to use it in other, more general environments (R5).

In [91] Ma et al. propose the synthesize trust degree evaluation model (STDEM) for use in dynamic distributed environments. Aim of the model is the derivation of a trust level for a participating component in such environments (R1). The solution given by the authors is based upon two different trust derivations and their appropriate combination. First, a direct trust degree is calculated. This degree is based on means of direct measurable actions, thus comes from a neighbouring component. Second, the indirect trust degree is calculated. It is mainly based on the reputation of the component and is being propagated by recommendations from third parties. The combination of both parts is performed by a so-called synthesization, which then forms the synthesize trust degree. Part of this calculation is a previously attenuation of the trust degrees which takes time dependent trust into account. Simulations made by the authors show, that the approach is useful, but only within their defined scenario (R6). Furthermore, the trust calculation is fixed, thus R5 cannot be fulfilled.

Another way of deriving trust is presented by Sadeghi and Stüble in [92]. Their paper aims to overcome the limitations of the attestation mechanism proposed by the Trusted Computing Group (TCG). That is, the TCG's approach (cf. [93, 94, 95, 96]) is based upon verifying a parties trustworthiness based on certain measurements taken on that party's platform. To do this, relevant specifications of the platform are compared against desired specifications. This introduces the problem, that each particular configuration must be known in order to provide a reference value for the measurements. Practically, this is rather difficult due to the nearly unlimited amount of different platform configurations. The paper introduces therefore the idea of basing this measurement on properties. Instead of the actual system-specific measurements, properties express only a certain feature. This allows to abstract particular features, e.g. a trusted boot process (cf. [97]) can be abstracted into a property expressing security of a platform. This allows to define a more general attestation method, which derives a platform's trustworthiness, that is no longer based on the specific configuration of that platform (R1, R5, R6). The definition of the properties itself is outsourced to a third party. This third party is considered to be trustworthy and therefore responsible for providing the correct properties, which limits the fulfilment of R5. There has been more work following this, mostly in the area of using properties for secure boot applications with available platform configurations and in the area of providing effective protocols. This work can be found in [98, 99, 100]. While the trust derivation itself is very promising, at least in the field of using the approach for attestation there needs to be more work done. This is mostly due to the use of a third party as property provider. Furthermore, as this approach belongs into the field of Trusted Computing, a more detailed analysis can be found in section 3.2.

As already stated, grid computing also demands a management of trustworthiness for the resources provided. Vijayakumar et al. show another approach on how to derive trust in [101]. The authors goal is to provide a measurement of the trustworthiness of a grid's resource provider. In detail, the approach presented calculates a so-called trust factor for a resource (R1). This trust factor is the combination of so-called security factors and security attributes. Both parts are combined using a particular function and form the overall trust factor. Although not written in the paper, security factors can be seen as expression for the direct trust while security attributes express the indirect trust. In this case, indirect trust is derived from reputation properties, like the authenticity of a provider. The security factors, thus the direct trust, is formed by a set of characteristics which may be valid for a service provider. For example, the set contains entries like intrusion detection capabilities or anti-virus capabilities. All the entries within this set are weighted in terms of their impact on the trust calculation. This allows to express differences in terms of trust for the characteristics. While located in another field, this approach does also base the trust derivation on some kind of properties, like Sadeghi's approach [92]. Although this approach does limit the capabilities which may be used for deriving trust, the idea itself sounds rather promising. An extended version of the approach can be found in [102].

There has been more work done in the area of trust management and trust derivation. The approaches presented in [103, 104, 105] all rely on a particular combination of direct and indirect trust. While direct trust may or may not be addressed, indirect trust is always derived using a reputation system. Taking all these result, a summary can be given.

3.1.4 Summary

The first approach presented, CADS, allows a holistic decision making process for smartphone enabled environments [2]. Above all, it was developed to address the special properties of smartphones which are included within a network, thus it provides outstanding integration and extension capabilities (R5 and R6). Due to this, it is well suited enough to be used as basic environment. CADS provides capabilities, for example the architecture consisting of Feature Collector, Feature Provider, Correlation Engine and Feature Consumer which is very useful for solving the problems given by the scenarios. More in detail, the concept of using a Feature Provider already allows to store the collected sensor data (i.e. the Features) on a centralised place. This can be considered as a good basis for the complete fulfilment of R2. Besides this, the Correlation Engine and the related policy provide also a sophisticated basis for R3 and R4. Although it does not provide any trust related mechanisms, its smartphone centric approach makes it well usable as basic environment.

Summarising the first group, intrusion detection approaches in general, it is easy to see that research has decreased in this area. This is due to the fact, that special environments, like wireless sensor networks or ad hoc networks need special, thus tailored treatment. In terms of the requirements stated in section 2, it can be summarised, that none of the presented approaches offers an out of the box capability to fulfil them completely. Besides the requirements which demand and easy integration and extension (R5 and R6) and a policy-based rule definition (R4), none of the other requirements may be fulfilled even by customising the presented approaches. While the latter ones define their sensors to be inherently trustworthy, thus negotiating the problem stated in R1, which may be acceptable in their own scenario, the earlier approaches do not even mention or define the problem of trustworthiness. Looking back at the latter approaches, some of them recognise the problem and emphasise that there needs to be more work in order to address it in a more general way. However, as there is not applicable solution, it comes clear that there is really an issue when collecting measurements from sensors and basing decision upon this collected information.

The second group which has been evaluated provides tailored intrusion detection approaches for particular environments. That is, in difference to the first group, the solutions proposed are only usable in the specific scenario. This scenario is always given by either a mobile and wireless ad hoc network, some kind of sensor network or grid and cloud computing environments. Due to this, there are several different intrusion detection types in this group. Starting by centralised approaches and hierarchical approaches, the range ends by fully decentralised, peer to peer based approaches. In terms of trustworthiness (R1), there are two distinct subgroups identifiable in this group. The first group does not actively research the problem of their sensors trustworthiness, which is mostly due to the scenario the solution is proposed for does not directly demand this. This subgroup is rather small as most of the approaches at least include the trustworthiness problem in their consideration. Due to this, the second subgroup includes some approaches which demand the addressing of the trust problem. One example for this are all approaches presented for mobile ad hoc networks. Due to the scenario of nodes which are randomly part of the network, trustworthiness of these nodes must be addressed. Most of the solutions made for this node-specific trustworthiness aim into the direction of a secure routing of messages in such a network.

Taking the third group, which is about deriving the actual trust measurement, also into account, it can be stated that there are two distinct types of trust. The first type, called direct trust is formed by specific properties of an entity. For example, if considering a node in an ad hoc network, this could be the battery consumption of this node where a high or abnormal consumption indicates a lower trustworthiness. That is, direct trust is also referred to a direct measurable value derived from a property of the system. In difference to this, indirect trust expresses trust measurements which have not been made on the entity itself but are proposed from another system. In detail, a common term used for this is the reputation of a system. If this reputation is high, i.e. other systems have successfully operated with this system, the indirect trust of this system can be considered higher. Furthermore, indirect trust is usually, but must not be exclusively, behaviour-based as it maps the actions of a certain system into a measurable value used for trust calculation. Combining these two trust types, an overall measurement for an entity's trustworthiness can be expressed. The combination of particular trust indicators, which may even be more fine grained than just indirect and direct, is also widely researched. The tailored solutions evaluated here allow for an effective combination and expression of trust, thus providing particular solutions to R1. While this is necessary in the approaches scenarios, the calculation methods are always static. That is, none of the approaches presented is able to change the calculation method in order to address a different situation. Providing a widely usable solution based on these approaches is rather impossible due to this. Additionally, the methods used to derive trustworthiness are also very scenario centric, thus static. They are always based on a limited set of system properties, which is usually not enhanceable, some kind of test message exchanged or a reputation measurement received from a third party. This idea of using system properties is rather straightforward and sounds overall promising. Unfortunately, most of the approaches limit the set of usable properties and thus making it impossible to apply the solution to arbitrary scenarios. The same limitation can bee seen for the test message exchange. It is useful in a very limited manner as this exchange might be compromised. If it becomes compromised, the whole trust derivation is unusable. The third approach, using reputation information seems to be promising as long as the methods used on the third party generating this reputation are clearly defined. This is missing in some approaches. Given this, requirement R5 and in some cases R6 are not fulfillable by the approaches of this group.

A summarised view on the requirements is given at the end of this chapter. Besides that, the other overall group of approaches, the technological-based approaches is presented next. It is based on the Trusted Computing initiative and founds trustworthiness on high level properties of a system. Although these approaches aim not directly for intrusion detection and decision making, the idea of using abstracted properties for trust derivation is rather interesting. Due to this, the following sections evaluate the abilities Trusted Computing is able to provide.

3.2 Technology-based Approaches

This section presents approaches which are based on one ore more technological specifications. In difference to the research-based approaches, these specifications are industry driven to provide best practices for companies. All of the approaches presented here belong to the overall field of Trusted Computing (cf. [106] and [107]). Trusted Computing can be considered as approach which provides means to base the security of a system on a hardware-based root of trust. That is, the use of trusted computing technologies allows to verify a system's integrity. This verification can be done in addition to a user's verification, which may for example be done by performing an authentication of the user. Although the user might behave as expected, trustworthiness can only be derived if the system itself is also to be considered trustworthy, which is applied by providing a mechanism to allow a system verification. There are two overall groups of specifications analysed here: (1) Specifications that are based on the concept of a Root of Trust and Trusted Platforms as well as (2) specifications that are based on Trusted Network Connect. Group (1) gives a definition of trust and provides means to derive and specify trust of a component. That is, it must be considered here as it presents approaches, which are valuable for the fulfilment of R1 and R2. Group (2) combines the approaches of group (1) with a standardised protocol layer. This protocol layer aims on providing a very high level of platform independence and interoperability. Due to this, it must be considered as it may provide a solution to fulfil R5 and R6. Furthermore, the proof of concept implementation of the already presented CADS approach, which was considered valuable, is based on the second group to achieve this high level of interoperability. Besides the analysis of the specifications itself, research-based extensions to the specifications are also analysed here as they may provide a more detailed (i.e., improved against the specifications itself) basis for the fulfilment of R1, R2, R5 and R6.

Trusted Computing itself is composed of several specifications which are created and maintained by the TCG. The TCG describes itself in [107] as a non profit organisation which is based on a membership principle of industry acting corporations and research institutions. That is, corporations and institutions can become a part of the TCG and take responsibility for parts of the specification process. The amount of actual members is steadily growing since the founding of the group under its previous name Trusted Computing Platform Alliance (TCPA). Furthermore, the TCG is divided into several sub groups which deal with a special field of interest, for example an infrastructure group or a trusted network connect group. Each of these sub groups maintains a particular set of the overall specifications. The specification process itself can be considered to be semi open to the public. This means, that there is first an internal, closed phase where the specification is only available for members to evaluate it, followed by a public review phase where the group accepts comments from all audiences. Based on these two steps, a specification becomes published after successful review of both parts.

As trusted computing aims on providing means to place trust on a system, the definition of trust itself needs to be analysed again. This is done in the following section.

3.2.1 Trust Definition and Trusted Computing

As written in the previous section, there is no consensus about the term trust and its actual meaning. However, in order to fulfil R1, a clear definition is necessary. Nearly all of the approaches presented above have their own definition. Particular different definitions can be found in [108, 91, 54]. When evaluating all these definitions which are different in terms of their effective results, there is nevertheless one similarity. All found at least a part of the term trustworthiness on the expected behaviour of a component or system. That is, if a system behaves the way it was expected, it is considered to be trustworthy. The Trusted Computing Group uses this basic definition for their own, stating in [109] that trust "*is the expectation that a device will behave in a particular manner for a specific purpose*". That given, the device can be considered trustworthy if it behaves as expected. Comparing this definition with the approaches presented above it is clear that the term behavioural trust needs to be clarified. Behavioural trust is mainly based upon the interactions of a system between another system. Given the TCG's definition of trust, this foundation of behavioural trust cannot be considered as being complete. This is due to the fact, that the TCG's definition bases the overall trustworthiness of the system on an expected behaviour of this system. This must not be mixed up with the behavioural term used in the approaches above. Instead of measuring the system's interactions and using them for trust derivation, the TCG's definition describes the trustworthy situation of a system based on the behaviour. In detail, a system may be considered trustworthy if it fulfils certain requirements and acts like expected due to this. For the remainder of this chapter, the TCG's view on trust is used: A system is considered trustworthy if it does certain actions like it is supposed to. There is no derivation of trust by different behaviour like in the approaches presented above. To enforce and measure this expected behaviour, trusted computing uses a special trust anchor. The following section introduces this trust anchor and explains the concepts which are used to inherit actual trust.

3.2.2 Root of Trust

The trust anchor used in trusted computing is formed by the so-called root of trust (cf. [110]) whereon all trustworthiness for a system is based up on, thus building a foundation for fulfilling R1. As already stated, trusted computing relies on a hardware root of trust. This hardware root of trust is given by the so-called Trusted Platform Module (TPM) (cf. [93, 94, 95]) and the components related to the TPM. The TPM itself is a special hardware component residing in a more or less protected way on the system's physical representation, usually the mainboard. It is a necessary component to fulfil the TCG's requirements for a so-called Trusted Platform. The term Trusted Platform is summarised later in this chapter. First, a detailed view upon the TPM is given, in particular about the features which provide the derivation of trust (R1).

Trusted Platform Module

The Trusted Platform Module is realised as a microchip providing the necessary functions to form a trusted platform. Its internal architecture is depicted in figure 3.3. The TPM can

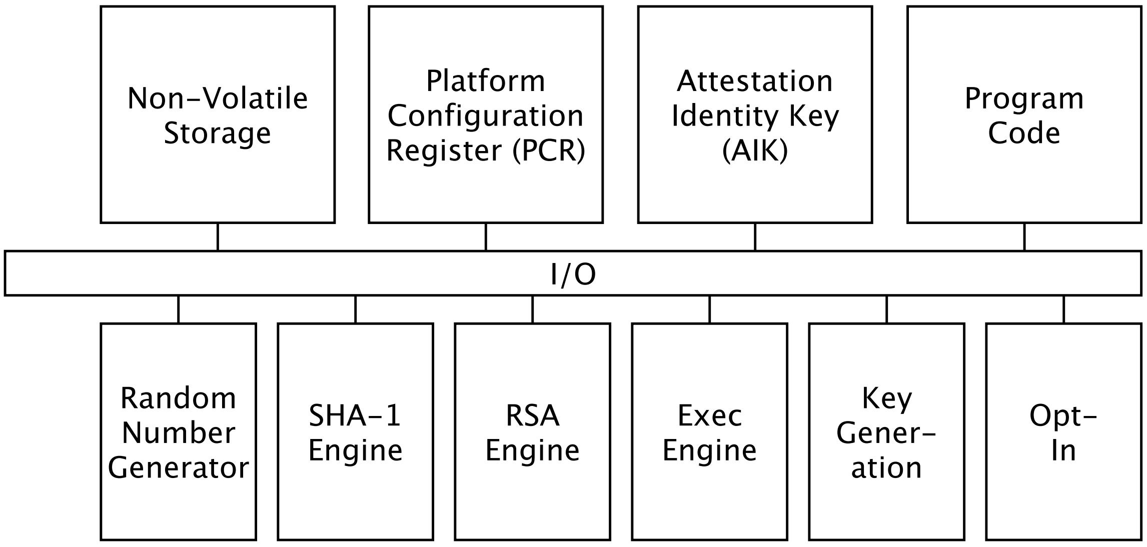


Figure 3.3: TPM architecture defined by [93].

be interpreted as a special kind of cryptographic processor (i.e. a co-processor) which is statically bound to the hardware. Although the TPM tries not to counter hardware-based attacks, e.g. accessing the platform in physical means and removing the entries stored in memory by cooling down this memory (cf. [111, 112]), it provides minimal tamper resistance. This means, that it should be practically hard to remove the TPM from its corresponding hardware or compromise it using a hardware-based attack.

To verify a platform's integrity a so-called integrity measurement takes place. The result of this measurement is securely stored using the TPM. This is done by utilising the so-called Platform Configuration Registers (PCRs). PCRs can be seen as a special kind of memory, holding a size of 20 bytes (i.e. a SHA-1 hash). These 20 byte values may be measurements of certain system configurations, like a configuration file or a boot image. The term measuring refers to performing a SHA-1 hash operation with the object to be measured. The PCRs are initialised with well known values when the system starts up and may only be changed by the use of a special operation. This operation, called `extend()` takes a new value and inserts them into the appropriate PCR. The insert process is not just changing the PCR's entry but invoking an algorithm which takes the stored value, concatenates it with the new value and finally hashes the resulting value into the PCR: $PCR_{new} = SHA1(PCR_{old} + digest_{new})$ [113]. This makes it impossible to simply put arbitrary and desired values into a PCR due to the use of a hash function and the concatenation with the already stored value. To successfully manipulate this process, it would be necessary to already change the first value stored within the PCR. This is rather hard to achieve, as the first measurement performed is invoked by another special component. This component, commonly referred to as Core Root of Trust for Measurement (CRTM) is represented by the first program code that is being executed when the system starts. It is responsible for performing the first particular measurement, usually the system's BIOS. The CRTM is placed on some hardware component on the system (likely on the CPU), it is usually not part of the TPM itself, thus forming the second part of the root of trust.

After measurements have taken place in the way described, they can be accessed by a third party. This is done by another function provided by the TPM, the so-called quote operation. Invoking `quote()` with the appropriate PCR(s) to be quoted, instructs the TPM to provide the value stored in the PCR. It is not only provided as plain value but is signed using the TPM's key hierarchy. The TPM possesses several keys which may be used for different tasks. There is a so-called Endorsement Key, which is unique for each TPM, thus making it possible to derive that a message signed with this key comes from a valid TPM. As this implies privacy issues, i.e. always using the same key, there is a set of derived keys, the so-called Attestation Identity Keys (AIKs). By using this keys, it is possible to verify if a message signed with one of them comes from a valid TPM but not from which particular TPM. These keys are used to sign the message generated when invoking the quote operation. That is, the PCR value is simply signed with one of these AIKs so that the verifier can prove this value to be originating from a TPM.

Using this measurement and storing ability, the TPM has another capability which has to be considered for the fulfilment of R1. This capability, which is called sealing, allows to bind certain information onto a particular state of the system. That is, sealing takes informations and encrypts them using the stored PCR values, thus making them inaccessible as long as the system does not reach that particular state. In detail, when the system starts up, the PCR stores certain measurements, like the state of the system's BIOS and the state of the boot image used. If some of the measured components change, their measurement value does also change. This implies another PCR value than the one which the information has been sealed to. It is therefore impossible to unseal the data without the use of unchanged components. While this mechanism might be used to securely store information on the system, which can only be accessed if the system is not compromised, i.e. in a well known state, it provides an even more useful application. It can be used to only boot the system up, if all measured value are as expected. That is, it is possible to define a particular state the system must have to properly boot up. If combining this with a X.509 certificate (as the proof of the system itself), this can be used to remotely verify, that the system is within an expected state. Due to this, trust may be derived from this information. The process of verifying this state is known as attestation and will be explained more in detail in section 3.2.3.

Summarising this, the root of trust in trusted computing-based systems is not only formed by a single component but by several components and certain values. That is, there are on one side hardware-based parts while there are also software-based parts. The hardware-based parts consist above all of the TPM itself. It holds the abilities to handle the results from measurements in a secure and trustworthy way, thus in a way that there is no lost in trustworthiness when storing measurements. Besides that, all other hardware components must be considered also as parts for a root of trust. This is due to the fact, that hardware-based components cannot be measured by the TPM appropriately. If one of these components implements malicious functions which are not exposed but used secretly, there is no way of recognising this. In addition to the hardware, the measurement itself can be considered as one part which forms the root of trust. In detail, the pre defined values which are used as reference when comparing the measurements are either interpreted as trustworthy or untrustworthy values. Untrustworthy values are stored as reference while every other value is being interpreted as trustworthy. Taking these two parts, is can be concluded that a trusted system will act as expected, and is therefore being trustworthy, if it complies with its hardware parts and the measurements taken. If the second part, the measurements are not the way they were expected, the process of measuring itself is still being trustworthy. This allows to securely identify systems which do not comply to a particular expected state. Furthermore, this allows to derive the actual properties which need to be taken into account if a system wants to be evaluated in terms of its trustworthiness, thus providing a particular solution for R1.

Given these building blocks, in particular the TPM, a so-called Trusted Platform can be defined. A detailed explanation of a Trusted Platform and the concept if a Chain of Trust is given in 7.2.

3.2.3 Trusted Network Connect

Another one of the TCG's specification is the so-called Trusted Network Connect (TNC) specification. It provides a standardised mechanism to use the techniques presented above, like an attestation of a device within a network. From the requirements point of view, leveraging certain capabilities of TNC may be well suited to fulfil R5 and R6. This is the case as one of the primary goals of TNC is to provide a high level of interoperability.

TNC itself can be seen as an improved version of traditional network access control (NAC) approaches. NAC approaches allow to limit the access a client gets when it accesses the network. This is done by leveraging approaches like the IEEE 802.1X [114] which provides a port-based access control. Furthermore, the access decision is usually based up on a policy defining which client can access which service. In oder to distinguish between different clients, an user authentication is usually performed. In case of the 802.1X protocol, the extensible authentication protocol (EAP, [115]) is responsible for performing the actual authentication. In addition to this, there are three roles defined: the so-called supplicant, an authenticator as well as an authentication server. The term may be slightly different for another actual NAC implementation, although the role itself is the same. The client, which requests access to the network takes the role of the supplicant. A switch is commonly used as port-based access device, thus taking the role of the authenticator. The third component, often depicted by an AAA server takes the role of the authentication server. If the supplicant now requests access to the network, it communicates this request to the authentication server. This is done through the authenticator, which only allows this kind of traffic from the supplicant. Practically, the protocol used for this kind of communication is EAP-based from the supplicant to the authenticator and RADIUS [116] based from the authenticator to the authentication server. In detail, the EAP messages are encapsulated into the RADIUS protocol appropriately. As already mentioned, a user authentication is commonly carried out within this process. That is, the authentication server first checks the user's credentials and based on the result of this check and the network's policy makes a decision. This decision is then communicated to the authenticator and consists of the information (a) whether the supplicant is allowed to access the network and if it is allowed (b) which kind of access is granted. The authenticator carries out the decision appropriately, thus connecting the supplicant to the network or denying access.

As easy to see, the access decision made by the authentication server is solely based on the user of the supplicant, not on the supplicant's properties itself. This may be a problem, if the access decision should also be based on particular features of the supplicant. For example, security related features like the supplicants patch level or certain anti virus countermeasures may be one of the aspects which form a decision. To allow for a check of those, some NAC approaches allow to measure the supplicant's system. That is, these approaches are able to determine the system's integrity and base decisions on this integrity. A commonly used concept to achieve this, is to use an agent on the supplicant which is responsible for performing the measurements and to communicate them to the authentication server. A problem arises when compromising this agent: it may send false measurements of the system, thus making the measurements useless as they are not trustworthy (cf. [117, 118]). Due to this, there is the general question about trusting the measurements done on the supplicants side. Trusted Network Connect now provides a way of putting trust in the received measurements by leveraging the capabilities of a trusted platform. TNC can therefore to some extent be seen as an enhanced NAC approach.

TNC provides the following capabilities, which are a mixture of common NAC abilities and exclusively trusted computing-based solutions.

Platform Authentication A Platform Authentication ensures two things: (1) the authenticity of the platform itself and (2) the integrity of the platform. Authenticity of the platforms means, that the platform which is communicating with the networks is exactly that platform which was expected to do this. The second point, the integrity (2) is measured on the platform and evaluated on the server's side. If it is as expected and the authenticity (1) was ensured, the overall process is considered successful. While the integrity may also be considered in an insecure way by other NAC approaches, the platform's authenticity is introduced by TNC.

Endpoint Policy Compliance The authorisation process for a particular platform is based on the integrity measurements done on that platform and a policy. This policy specifies the different cases in which a platform may or may not get access. For example, the policy could hold information about the operating system or the platform's patch level.

Access Policy In addition to platform-specific decision making, the user of the platform should also be taken into account. This may be done by using standard approaches (i.e. 802.1X) to perform an additional user authentication.

Assessment, Isolation and Remediation If the platform is not allowed to access the network due to not complying with the network's policy, the platform is to be isolated. Isolation can be distinguished into two cases. The first case denies every access to the network, thus effectively interrupting all communication possibilities between the network and the platform. In the second case, the platform is placed within a special quarantine network segment. Within this segment, the platform has the possibility to solve the integrity related problems which lead to its access denial. This is for example done by providing access to particular update servers allowing to platform to update certain applications or the OS. After resolving has taken place, the so-called remediation can be carried out. Remediation allows to re-assess the platform's properties again and if the platform is now policy compliant (i.e. the platform authentication is successful), access can be granted.

Architecture

The TCG provides an architecture which allows to realise the features explained above. The most overall version of this architecture is depicted in figure 3.4. The architecture consists of two conceptual parts. The left side, which provides the functions like platform authentication which was explained in section 3.2.3 and the right side which contains

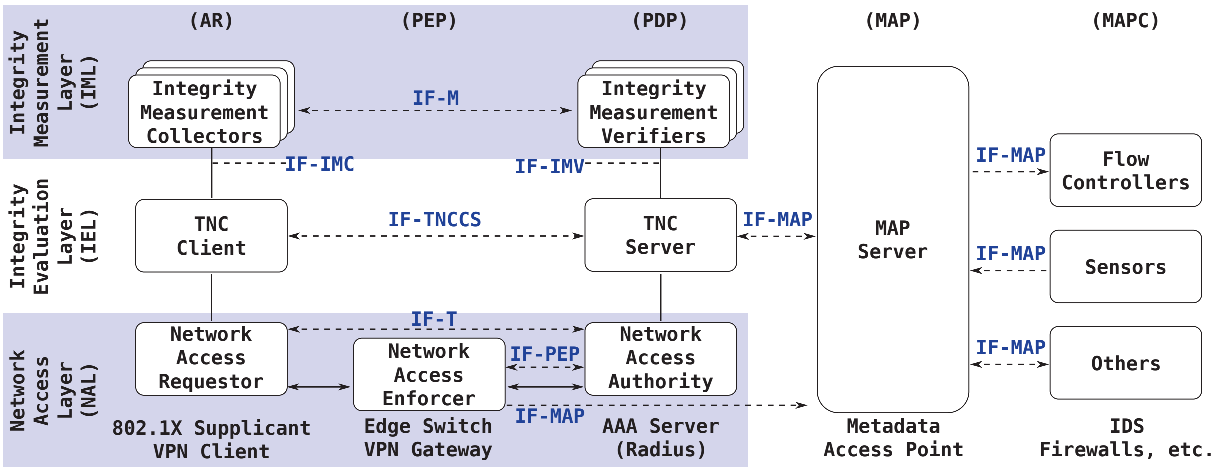
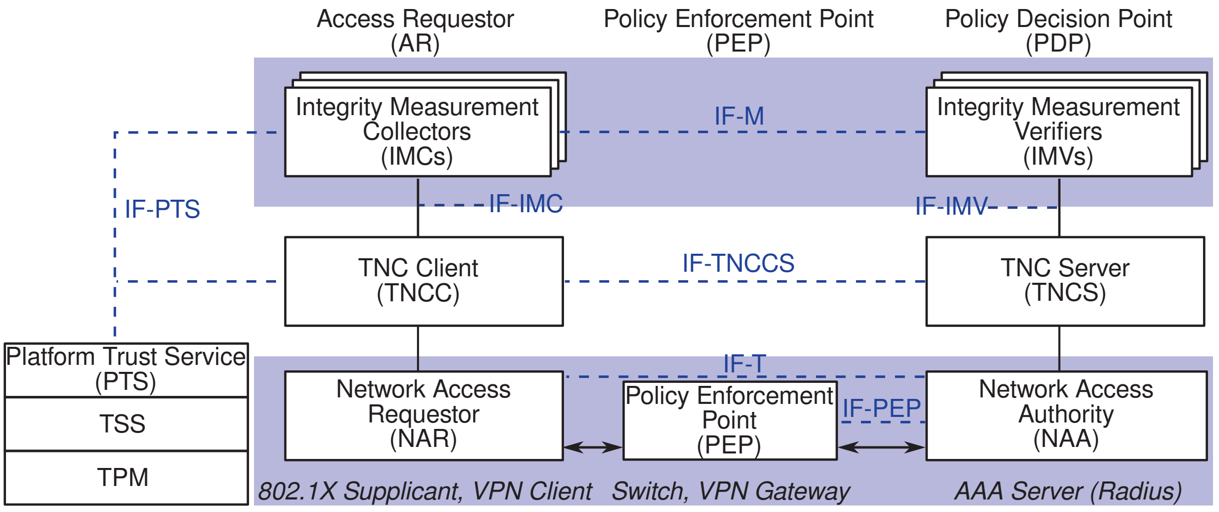


Figure 3.4: Overall TNC architecture [96].

IF-MAP specific extensions. This section first deals with the left side, while the right side is explained later in section 3.2.3. A more detailed version of the left side is shown in figure 3.5. The remainder of this section is based on this architectural version. The



Access Requestor (AR)
Policy Enforcement Point (PEP)
Policy Decision Point (PDP)
Integrity Measurement Collectors (IMCs)
IF-M
Integrity Measurement Verifiers (IMVs)
IF-PTS
IF-IMC
IF-IMV
TNC Client (TNCC)
IF-TNCCS
TNC Server (TNCS)
Platform Trust Service (PTS)
TSS
TPM
IF-T
Network Access Requestor (NAR)
Policy Enforcement Point (PEP)
IF-PEP
Network Access Authority (NAA)
802.1X Supplicant, VPN Client
Switch, VPN Gateway
AAA Server (Radius)

Figure 3.5: Detailed TNC architecture without IF-MAP [119].

architecture consists of logical entities, denoted by blocks and standardised interfaces, denoted by the prefix IF. Communication either via a network or by processes are shown by lines, in some cases with arrows indicating the flow of the communication. There are three components defined by the architecture, which map the roles that were explained for general NAC approaches. They are realised as vertical sections in the architecture.

The Access Requestor (AR) takes the role of the supplicant, thus requesting to access the network. It is usually an endpoint device like a laptop or a smartphone. The Policy Enforcement Point (PEP) is responsible for providing or denying this requested access. It therefore acts as the authenticator. Commonly, a switch which is capable of performing port-based access control is used as component for this. The third component, which is the Policy Enforcement Point (PDP) is responsible for evaluating the AR's measurement and comparing it against a policy, thus making a decision about whether the AR should gain access or not. It is therefore in the role of the authentication server. The three components and their communicational patterns are divided horizontally into three different layers. Beginning with the lowest layer, the so-called Network Access Layer, via the Integrity Evaluation Layer and ending on top in the Integrity Measurement Layer.

A detailed explanation of the TNC architecture is given in 7.3.

Policies

Three different logical policies can be identified, whereas logical refers to the policies being different in their means while they may be physical located within one file. The three types are the IMV policies, the TNCS policy and the NAA policy. The IMV policies control the evaluation of the particular measurement values. They are only valid for the appropriate IMV and there may be more than one of them, each for every IMV. These kind of policies control for example which patch level of the system is valid. The second type, the TNCS policy controls the decision making of the TNCS. This is necessary as the TNCS receives a suggestion from each IMV, thus it may receive an arbitrary high amount of different suggestions. The TNCS policy controls which kind of decision is made based upon the several suggestions, thus it effectively controls the weighting of all different aspects measured. By the use of this policy and the suggestions, the TNCS can perform a decision making process. Although this decision is the final action from the TNC's concept point of view, the Network Access Authority has the overall decision. To control the behaviour of the NAA, another policy is used. This NAA policy holds aspects like the different users and different groups which may be treated differently. All of these policies are needed in order to define allowed states for a client.

In addition to these PDP-specific policies, there may be a fourth kind of policy. As the AR is located on the client device, there is one problem in terms of privacy: the measurements which are performed may contain arbitrary data about the client. This leads to a situation, where the user of the client has to trust the infrastructure about not collecting the user's information. As smartphones may contain sensitive information, which the user don't want to share, it is necessary to provide a way to at least inform the user about what is collected and give her the choice to decide. Although this is not directly addressed in the TNC specifications itself, there is some research-based work which provides a solution for this and allows a wider use of TNC.

Bente et al. introduce a concept called client side policy in [120]. It allows the user of an Access Requestor to be actively part of the TNC handshake by accepting or denying a measurement request. This is done by defining a so-called client side policy (cf. figure 7.5 and listing 7.1 for an example), which is based on the IF-M specification [121] and allows to define which attributes may be measured or not. In addition, the concept also limits the measurement abilities to the defined attributes. This limits the server in requesting arbitrary things from the client. In detail, the policy is placed by enhancing the architecture of TNC, shown in figure 3.6 and thus also extending the message flow (see figure 7.6). As it easy to see, the Access Requestor is enhanced by a Policy Manager (PM), which

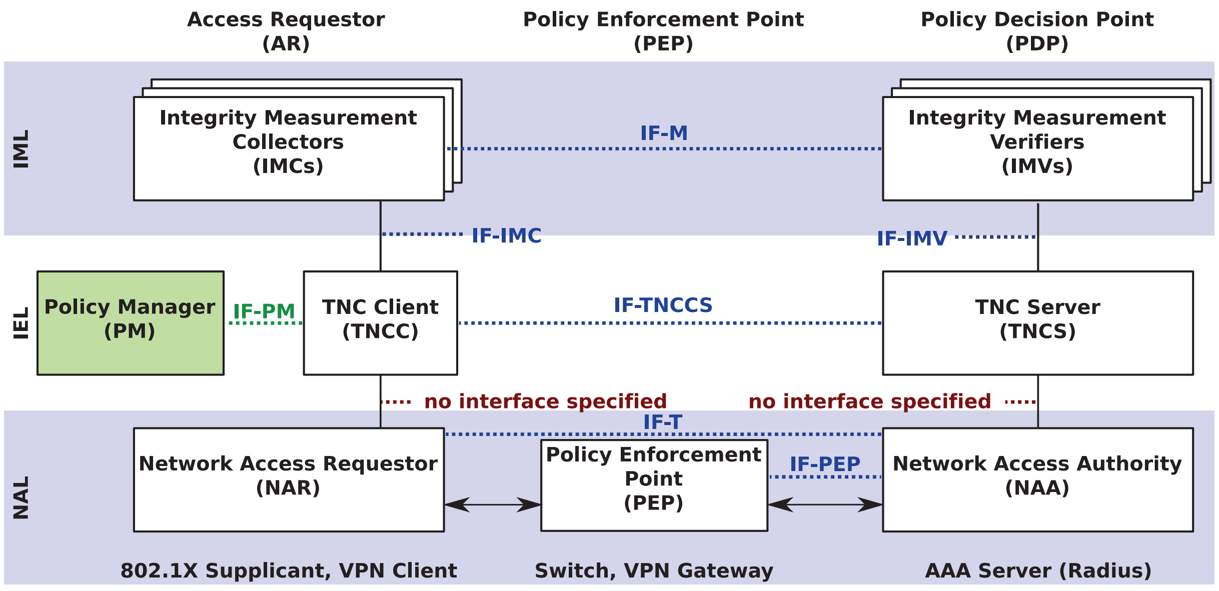

Access Requestor (AR)
Policy Enforcement Point (PEP)
Policy Decision Point (PDP)
IML
IEL
NAL
Integrity Measurement Collectors (IMCs)
IF-M
Integrity Measurement Verifiers (IMVs)
IF-IMC
IF-IMV
Policy Manager (PM)
IF-PM
TNC Client (TNCC)
IF-TNCCS
TNC Server (TNCS)
no interface specified
no interface specified
IF-T
Network Access Requestor (NAR)
Policy Enforcement Point (PEP)
IF-PEP
Network Access Authority (NAA)
802.1X Supplicant, VPN Client
Switch, VPN Gateway
AAA Server (Radius)

Figure 3.6: Extended TNC architecture [120].

handles the necessary policy interruptions. As long as the TNCS accesses properties and attributes that are defined as allowed, the whole process is the same as in standard TNC.

If the TNCS wants to access arbitrary, undefined properties or disallowed attributes this is denied and an empty message is returned to the TNCS. Although this approach ensures the privacy to some extent, it depends on the TNCS's policy used on the PDP. That is, if one of the attributes which has been denied by the client is considered highly valuable, the PDP certainly won't allow access without it. Nevertheless, the approach allows to deploy the TNC architecture in fields that demand privacy considerations, as already stated like in smartphone enhanced environments.

Remote Attestation

Going back to the features trusted computing provides, remote attestation is the most valuable one in terms of deriving trust (R1). Given all the techniques and concepts from the previous sections, like the root of trust or TNC as communicational framework, remote attestation can be explained in the following. Furthermore, there are several research approaches which allow for more sophisticated attestation approaches. Some of them are also shown and evaluated.

Remote attestation describes the actual process of receiving integrity related information from the client which describe the platform's state and which can be considered trustworthy under some circumstances. As already stated this is done by measuring the platform's properties and extending them into the TPM's PCR(s) effectively building a measurement chain. When verifying the measurements, the final value is retrieved from the TPM. This is done by performing a quote operation for the appropriate PCR(s). Within this quote operation the values are signed by the TPM with a key, allowing it to verify the validity of the TPM. This is necessary as the TPM is one of the root of trusts for a trusted platform. The key which is used to sign the quoted values are one of the already described AIKs. The connection of this AIK is given due to the AIK being derived from the TPM's EK and by establishing a relationship of the AIK and EK. This is done by an external party, a so-called privacy CA, which signs the AIK and attests that it belongs to a valid TPM. Also this process is rather straightforward, it demands that the privacy CA acts trustworthy. This is necessary as if the privacy CA provides its information to the verifiers of the signed PCR values, the verifiers would be able to uniquely identify the platform each time an interaction with this platform takes place. To circumvent this problem, Brickell et al. propose the so-called direct anonymous attestation approach (DAA) in [122] and [123]. The DAA introduces a tailored protocol, which allows to derive the validity of an AIK without the use of a privacy CA. It is a rather complex protocol but allows to avoid the privacy limitations if using the privacy CA approach. Independently of using the DAA or privacy CA approach, the verifier is able to reconstruct the measurement chain. This is due to the platform not only sending the quoted values of the PCRs but also a log, the so-called stored measurement log (SML), which includes all information about the measurement steps that were carried out on the platform. Given the quoted PCR values and the SML, the verifier can rebuild the measurement chain on its own and, if everything is as expected, by performing each measurement step described in the SML on its own should come up with the same value that was received from the platform. The SML itself is not secured at all as this is not necessary. If there are differences between the SML and the actual measurement chain, e.g. in case the SML was compromised, the verifier won't reach the final value, thus it recognises that something is not as expected.

The process of measuring each component of the platform based on hashing of the executables of this component is the TCG's proposed standard approach for attestation. It is called binary attestation, and relies conceptual solely on the measurement of binaries on the systems. This imposes some questions, with some of them already being mentioned. The main problem of binary attestation is the scalability of this approach (cf. [124, 125]). To get a complete picture of the system, every component (i.e. every binary) needs to be measured. Given an actual system with a common operating system like a Linux- or Windows-based one, there is a rather high amount of components which need to be measured. Even if it would be possible to measure all components in a reasonable way, for example in a reasonable amount of time as hashing takes some time per file, the approach does still not scale very well. This is due to the fact, that each time the system reboots things are changed on the platform. Even without the user doing something, like storing new programs or files, the system also changes. In particular, every time the system updates itself, the changes are even of higher impact. This leads to a situation, where each reboot would require new reference values, which is practically very inefficient. To counter this, the system can be divided in several distinct compartments [126, 127, 128]. One of these compartments may be interpreted as security critical and be set to a read only state which prevents changes to it, even when rebooting. Although this idea provides some more usability, the problem of changing system configurations through updates and the high amount of possible system configurations remains.

To perform an actual remote attestation, the TNC framework may be used. TNC provides a full set of methods to perform a remote attestation in a interoperable manner (R1, R5 and R6). There are no problems if performing this while the client (i.e. the AR) is directly connected to the network, not using some kind of VPN approach. If the access is provided through a VPN tunnel, the TNC framework proposes several means to allow for an attestation. First of all, the VPN gateway takes over the role of the Policy Enforcement Point, thus being responsible for enforcing the access to the network. Furthermore, the approach strongly relies on the VPN being established using IPSec [129]. This is due to the fact, that the TNC handshake is proposed as being part of the Internet Key Exchange performed within the IPSec negotiation phase. In detail, TNC demands version 2 of the Internet Key Exchange Protocol (IKEv2, [130]) as only this version if capable of encapsulating the appropriate TNC-specific packets. This can be seen as a limitation, as there are several other technologies for establishing VPN connections (e.g. [131]). Addressing this limitation, there are approaches which allow to perform a remote attestation in VPN environments.

Schulz et al. propose an approach in [132, 133] which also extends the IKEv2 protocol in order to support a remote attestation. The aim of this work is rather in the area of simplicity and efficiency to support arbitrary attestation approaches. In detail, the approach negotiates the exact version of which kind of attestation is carried out in the phase of negotiating the IPSec's security association. The attestation negotiation is therefore done by introducing a new structure which allows to communicate about the attestation method. The actual attestation process is done after the IKEv2 part is nearly finished. In this phase, attestation data is communicated between the parties and a final connection is only established if this phase is successful. The approach itself does not rely on TNC, but provides effectively the same functionality with the improvement of supporting arbitrary attestation techniques. As with the TNC approach, the VPN implementation itself has to be changed in order to support the IKEv2 and attestation data enhancements, which renders this method only interoperable in a very limited way.

Besides this approach, Baiardi and Sgandurra propose another approach in [134]. Although their work is mainly about performing a remote attestation within a so-called overlay network, they do also rely on a VPN environment. In difference to the previous approaches, this environment is based on OpenVPN [131].In order to perform a remote attestation, the authors enhance OpenVPN itself. In detail, a particular plug in is implemented adding remote attestation to the OpenVPN components. Comparing this to Schulz's approach mentioned above, the concepts are similar to a certain extent. Both approaches use the possibility to customise and extend the VPN components. Besides that, the approach of Baiardi et al. only allows for a conventional binary-based attestation. Another common point for both approaches is that they do not aim at interoperability as both require changes to the software components providing the VPN environment. Another approach based on TNC aims to provide this interoperability.

In [135] Bente et al. propose an idea to perform a TNC-based attestation without the need to change components responsible for the VPN connection. It is based on the second version of the IF-T [136] protocol, which provides a method for communicating TNC messages using an already existing ip-based connection. In detail, a TLS secured tunnel is established which allows to perform a remote attestation. This is done by allowing the AR in its initial state to communicate with only the PDP. A packet filter is used to enforce this communication and operates on the VPN gateway so no other communication is possible for the AR. After the handshake is finished, the filter may open up the network. Although software is unchanged if using this approaches, some special components are added to the architecture (figure 7.7 for details). There needs to be the packet filter, which acts as Policy Enforcement Point and to provide real interoperability a so-called PEPd[1] which provides IF-PEP functionality. Figure 3.7 depicts the changed flow of messages. The steps which are carried out by using this approach are as follows.

1. First of all, the VPN connection is established. This is done in a regular way, i.e. without any TNC or attestation-specific means. Due to this, there are no changes to the VPN software necessary. The VPN tunnel is established after this step.

2. Next, the TNC handshake starts. The necessary components on the AR perform the required measurements and the results are communicated via the IF-T protocol through the established VPN tunnel.

3. The PDP receives these results, as the packet filter allows the AR to communicate with the PDP. It does not allow any other communication at this time. The PDP can therefore perform the evaluation of the received results.

[1]The d indicating this as a daemon component.

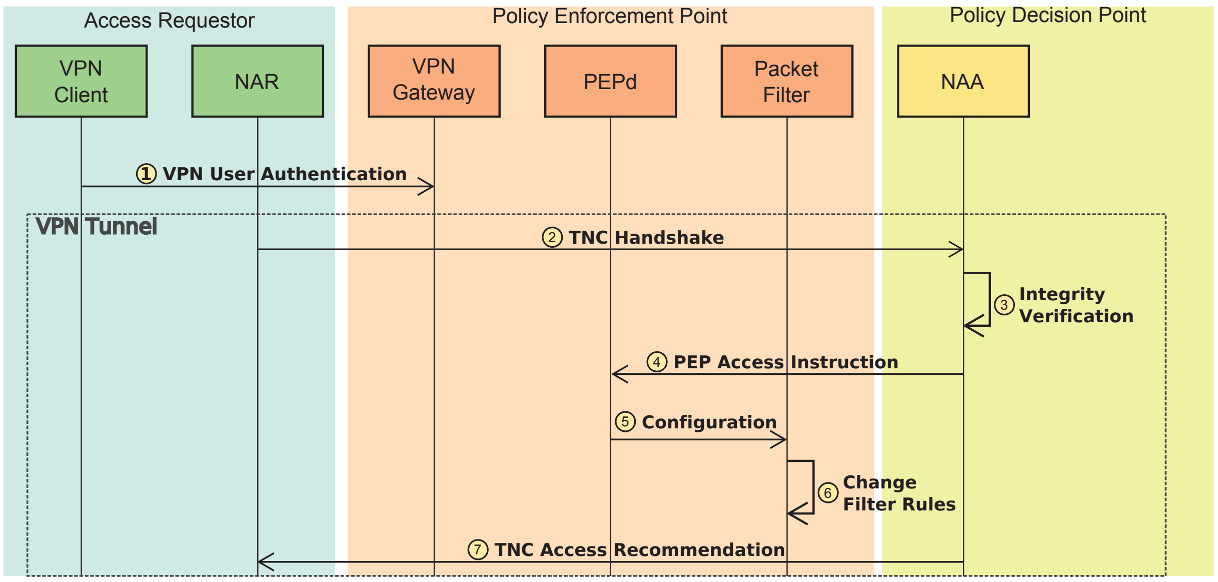


Figure 3.7: Changed message flow for VPN [135].

4. If the evaluation was successful, the PDP sends the enforcement instruction back to the PEP. The PEPd is now responsible for receiving this message, as the packet filter is usually not able to understand IF-PEP protocol-based messages.

5. Due to this, the PEPd is also responsible for triggering a rule change of the packet filter. This rule change intends to allow network access for the AR.

6. The actual process of changing the rules is then performed by the packet filter itself. In detail, it simply updates the rule set in order to allow the network access.

7. Finally, the access recommendation is communicated to the AR. Although this step is depicted last, it may happen in parallel to the previous steps, as soon as the PDP has sent the access recommendation to the PEPd. Finishing this, the AR gets access to the network.

The approach provides also means to isolate an AR if necessary. This is done by instructing the packet filter with a specialised rule set which only allows access to a particular isolation network. Furthermore, the approach not only provides interoperability due to it's straightforward integration but also support for arbitrary attestation methods. Such other methods are shown in the following.

As already stated, binary-based attestation has several drawbacks which are mainly based on the poor scalability of the approach. Furthermore, there are also environments which provide other, more tailored, means to attest the integrity. The approach of an attestation based on properties rather than on binaries presented by Sadeghi et al. in [92] was already explained in section 3.1.3. The idea of this approach is to define a platform's properties, like security relevant properties, and use a third part to translate the binary measurements into property-based measurements, thus effectively attesting the properties. The problem within this approach is the use of a third party which maps actual configurations to properties. As with the privacy CA-based approach, this introduces privacy related problems, as the trusted third part might spread identifiable information. In [137] Chen et al. introduce a protocol which solves this problem. This is done by using specialised signatures types, which allow to map configurations on properties without the use of a third party. Using this refined version, property-based attestation provides a promising way of deriving trust without the scalability problems introduced by plain binary attestation. Some more views at property-based attestation are given in [138].

Another approach using properties is presented in [139]. It does not define properties in the same way as the property-based attestation approach does, as it aims on dynamically measuring a platform's state. Thus, properties are defined as constraints a program[2] must fulfil in order to be considered trustworthy. These constraints are based on pre defined behaviour profiles programs possess when executed. At runtime, these properties are measured, thus providing a dynamic version of the remote attestation approach.

Moving forward from trusted computing-based approaches, another attestation approach aiming at embedded devices is introduced in [140]. This approach, called SoftWare-based Attestation for Embedded Devices (SWATT) is being able to verify the memory contents of a device from an external party. This is done by providing a special verification algorithm to the embedded device, which is able to generate memory-based checksums. If an external party needs to verify memory contents, it sends a challenge to the device. The device then calculates a checksum in a well defined manner, including some time critical constraints, and responds to the verifier. Like in the TCG's approach, the verifier is able to do the same with the reference memory content and can therefore proof the answer correct. If the answer is unexpected or the calculation time differs, the device's memory is interpreted as compromised. In difference to the TCG's attestation, SWATT

[2]I.e., an executable and its instantiation at runtime.

only uses a software-based root of trust (cf. [141]). This is possible for this area as an embedded device has a defined architecture, as well as defined components and software. While not directly addressing the field of problems the TCG's approach aims for, this approach can be used to attest embedded devices and thus provides a possibility to derive trust for such devices. An improvement of the pseudorandom memory traversal concept SWATT is based on is given in [142]. In this paper, the authors propose to use a SWATT like approach to determine the trustworthiness of a node in a sensor network. It is done by taking an amount of distributed nodes to determine the integrity state of another node. Results of simulations show, that the approach allows not only to detect one single compromised node but also works for more than one node.

Going back to the core attestation scheme proposed by the TCG, there is another approach which tries to circumvent the problems of both, the TCG's binary approach and the property-based approach. This approach, called behaviour-based attestation, is presented by Li et al. in [143]. It is based on two conceptual parts: a trustworthy bootstrapping of the system to initialise the necessary components and second the monitoring of the system's behaviour. The authors propose to use the techniques introduced in [144] to achieve a secure bootstrapping procedure. They do not propose other, in particular new means for this area. The second part, the behaviour monitoring is based on a policy. This policy defines expected behaviour on the verifiers side. On the client's side a special module is responsible for monitoring the behaviour and measuring it appropriately. This module is secured by the bootstrapping of step one. The actual attestation is based on the measurements of the behaviour which are communicated to the verifier. The verifier is able to rebuild this behaviour and can compare it against the policy which defines expected behaviour, thus trustworthy behaviour. Although behaviour can be considered a rather good indicator for the trustworthiness, it is unfortunately not the only one (cf. direct vs. indirect trust). Due this, the authors approach is only able to record and attest the results of untrustworthy actions. It is not able, unlike the other approaches, to compare a certain state of the system. Nevertheless, there might be applications, where this kind of trustworthiness may be considered enough in order to derive trust.

Naumann et al. propose another behavioural-based attestation approach in [145]. In difference to the approach presented above, it is based on the low level binary measurements which are mapped to a behavioural policy. That is, a pre defined behavioural model is used to describe a policy. This policy is used as a reference for future attestations. Given the platform of the client, a framework introduced by the authors is used to abstract low level binary measurements in a high level behavioural form. That is, the behavioural model of the actual platform is derived by a defined set of binary and property-based measurements which allow to map a specific configuration to a specific behavioural pattern. This pattern is used to perform the actual attestation. Based on the pre defined behavioural model, the pattern is matched against the policy. The policy is capable of defining expected behaviour as well as unexpected behaviour. This allows to include all platform relevant properties, which effectively reduce the problems of the approach presented prior to this one.

Another approach which uses the behaviour of a certain piece of code as attestation base is introduced by Haldar et al. in [146]. It is called semantic attestation and uses a special architecture, which includes a trustworthy virtual machine running on the platform. Programs which are intended to run on the platform are loaded into this virtual machine. A policy on the verifiers side defines which behaviour is considered trustworthy. The program runs within the virtual machine, which makes it possible to monitor it's behaviour in a sandbox like environment. The behaviour defined in the policy is then evaluated against the behaviour that was monitored in the virtual machine. This process renders the actual attestation, as the verifier decides based on the monitoring results. The approach demands for a secure virtual machine environment as the verifier puts trust into the observations made by the virtual machine. Due to this, it is also necessary to enable a secure bootstrapping of the client. Furthermore, although the approach might be used for arbitrary types of programs, the authors describe it explicitly for the use of java executables running in a trusted java virtual machine.

Given the area of mobile security, Naumann proposes an approach to perform a remote attestation for the Android platform in [147]. It is based on the usage of the Integrity Measurement Architecture [148] in order to start a Chain of Trust from the kernel level. However, the authors do not rely on existing open source software that provides the necessary functions. Instead, they implemented a mini TPM emulator as well as a mini TSS with a rather limited functionality. Furthermore, the paper proposes two attestation methods: so-called application level attestation and so-called class level attestation. Application level attestation measures installed apps prior to their execution. A problem arises as the established chain of trust cannot be considered complete. This is due to the fact, that the approach does not measure the Android Java framework constituting of a set of core libraries. To overcome this limitation, class level attestation is used in the overall approach. In detail, each single Java class is measured after it is loaded but before it is executed, thus fixing the otherwise broken chain of trust. While it can be considered as a complete attestation scheme, the approach solely relies on standard binary attestation, thus not covering the specific permission model of the Android platform and thus only partially fulfilling R1. Furthermore, the class level attestation approach increases the number of taken measurements linear to the number of the app's Java classes, which makes the poor scalability of binary attestation even worse. This poor scalability directly affects the fulfilment of R6 as it is difficult to integrate within an arbitrary amount of devices. Furthermore, R5 must also be considered not fulfilled as the approach does only rely on the binary-based attestation mechanism.

An approach which includes the special properties of the Android platform is presented by Bente et al. in [149]. The authors propose an attestation scheme which relies on the permissions given to the apps installed on that platform. That is, as permissions control the behaviour of the apps by limiting their access rights, this approach can be seen as a very special kind of a behavioural-based attestation. The approach is not only based on simple evaluation of permissions given to apps but also on complex conjunctions of them (see figure 7.8 and 7.9 for examples). Furthermore, the approach relies on a custom rom, which allows to attest the Android platform's basic systems, such like the kernel and the virtual Java environment. This is done by leveraging the IMA [148] system which performs a binary measurement of all static components, up to the application layer. At the point of the application layer, the permissions for each installed app are attested (R1). That is, the verifier receives two parts: a proof of the basic system based on a binary measurement and the measured permissions for each app. A policy defines which permission combinations are trustworthy. More information, in particular about the detailed message flow and the performance of the approach can be found in section 7.7.

Bente et al. propose another approach for a Google Chrome (cf. [150]) based environment in [151]. It can, to some extent, be seen as another version of the permission-based approach presented above. However, it aims on performing a remote attestation for the Chrome OS platform, which has some features allowing also to base the attestation upon the installed apps. In detail, Chrome OS provides a well defined basic system, including the kernel and everything up to the Chrome browser. This basic system is secured inherently by a specialised version of a trusted boot. That is, the Chrome platform verifies the integrity state of the system and only boots the system if its state is as expected. Apps provide application level functionality and are run inside the Chrome browser's sandboxed environment. Due to this, the approach provides two things: first it attests that the measurements made are originated from a valid Chrome platform and second, measurement results of the apps installed inside the browser. Proving the platform to be a Chrome device is enough, as the trusted boot ensures the system's integrity. If the proof can be given, it must have been a valid booted Chrome system answering the proof request. This puts the root of trust to some extent on the side of Google as the approach relies on their trusted boot. The actual measurements are taken from the applications installed on that platform, thus residing in the Chrome browsers sandbox. They are based on several properties, for example the manifest defining such an app or the actual binaries. More details of the approach are shown in section 7.8. Using this approach, an effective remote attestation can be performed for Chrome OS-based systems. In difference to the TCG's binary attestation, the approach is able to measure at a higher level by using the properties of the particular apps.

Given all these different approaches, it can be concluded that there are a lot of possibilities of deriving the actual trustworthiness of a device, thus providing basic methods for fulfilling R1. Furthermore, it is noticeable that there are well suited approaches for most of the relevant device classes, in particular for Android-based smartphones. These approaches can be used in order to fulfil parts of the requirements. Besides that, it is also clearly to see that none of the approach provides an all together solution which is able to fulfil all requirements, in particular R2 and R3 are not considered at all. This is due to the specific scenarios the approaches are intended for in addition to their fundamental research-based characteristics.

IF-MAP

As written in section 3.2.3, TNC consist of two conceptual parts. Figure 3.4 depicts both parts, with the concept responsible for providing attestation possibilities shown on the left side of the figure. The other part, which consists of the specification for the so-called Metadata Access Point (MAP) concept forms the right side of the figure. The standardised interfaces (IF-MAP) are mainly specified in [152] by the TCG.

IF-MAP can be seen as a special version of a content-based publish subscribe system (cf. [153]). That is, there are entities which provide information to an authority and other entities which subscribe to this authority based on a specific content and consume the appropriate information. More in detail, there are two logical components defined in IF-MAP. The first one, which is called MAP Client (MAPC), is responsible for (a) publishing information and on the other side for (b) consuming information. There may be an arbitrary amount of such MAPCs in an IF-MAP enabled network. The second component, the so-called MAP Server (MAPS) is the central authority responsible for receiving published information from the MAPCs and handling the appropriate subscriptions for other MAPCs. Examples for MAPCs are flow controllers of the network or simple clients providing logging information. Using the provided information, the MAPS creates and maintains an internal graph structure depicting the information and to some extent their relationship. The graph is based on so-called identifiers and so-called metadata. Identifiers are nodes which represent some kind of a key element making a set of information identifiable while metadata are data which describes this key element. A device identifier may for example possess certain capability metadata expressing certain abilities of the device. An example graph is depicted in figure 3.8, the grey elements represent identifier while the green and red elements show metadata. Furthermore, green depicts infrastructure related metadata while red shows smartphone metadata. The example itself shows a possible graph for an environment consisting of the CADS system, one or more smartphones, infrastructural components like a DHCP server and a TNC-based PDP (e.g. [154]).

The basic specification of IF-MAP does not define which metadata should be used. It only defines the abstract relationship between identifiers and their metadata. The particular metadata is either specified in addition (cf. [155, 156]) to the basic specification or must be specified in a vendor-specific way for the particular environment. That is, in order to actually use IF-MAP, all MAPCs and MAPSs must be able to understand the data exchanged. Interoperability can be achieved if the standardised metadata is being used as each of the components may not be from the same vendor but all must implement the standard specifications.

The CADS approach which was explained in section 3.1.1 is also, from the implementations point of view, based on the IF-MAP standard. The concept of the implementation consists of two parts: the mapping of the appropriate components like the Feature Collector to the IF-MAP roles as well as the definition of appropriate metadata. Both is given in

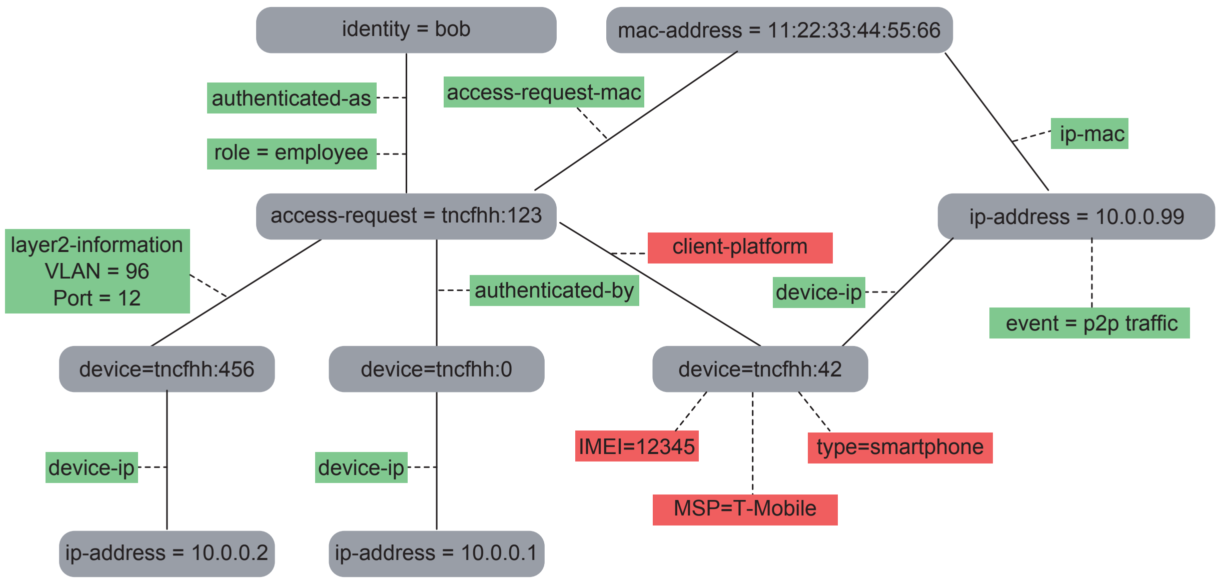


Figure 3.8: Example graph.

detail by Bente in [2]. The CADS architecture is formed by four logical components: the Feature Collector responsible for measuring Features, the Feature Provider which handles the storage of the Features, the Correlation Engine which evaluates Features and the Feature Consumers which somehow process Features. Each of these four components are mapped to the explained IF-MAP roles of MAPC and MAPS. Figure 3.9 depicts this particular mapping of the IF-MAP roles within the CADS implementation. As it is easy to see, the Feature Collector, the Correlation Engine and the Feature Consumer are mapped as MAPCs. The only component being a MAPS is represented by the Feature Provider. This is rather straightforward, as the Feature Provider is responsible for collecting and providing Features from and to all other components. Furthermore, CADS demands that there is only one Feature Provider and one Correlation Engine. This corresponds at least for the Feature Provider with the IF-MAP approach, as there is usually also only one single MAPS. In addition to this, there may be an arbitrary amount of Feature Collectors and Feature Consumers, which is also as supposed as they act as MAPCs.

IF-MAP defines the communicational means only for exchanging data in a publish subscribe based manner. The semantics of the data itself is defined in additional specifications or must be self (i.e. vendor-specific) defined. In case of the CADS system, the IF-MAP provided specifications were insufficient to handle the requirements the CADS system de-

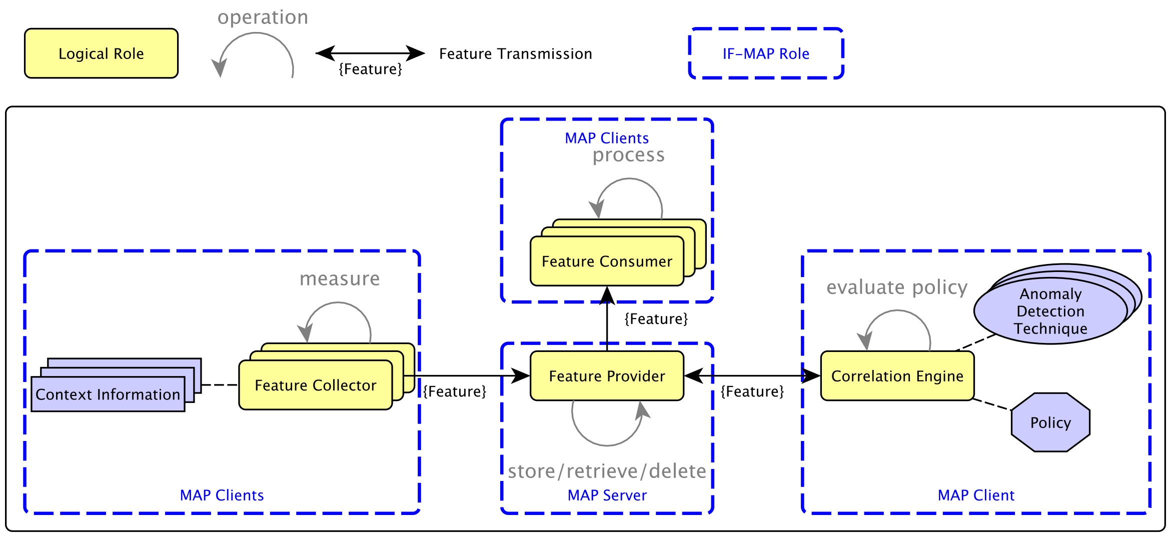


Figure 3.9: IF-MAP mapped CADS architecture [2].

mands (cf. [2]). Due to this, CADS uses its own vocabulary of metadata, which is strongly based on the Feature concept of CADS. This vocabulary is mainly based on Features and Categories. While Features, as already explained, encapsulate measurement values in conjunction with contextual information, Categories are used to structure Features. In detail, for each smartphone which is measured, a Feature tree, consisting of Features in its leafs and Categories in its branches is constructed. This Feature tree is stored on the Feature Provider, i.e. on the MAPS besides the standard IF-MAP graph, thus being an extension to the standard graph. Figure 3.10 depicts such an example graph which results from the use of the CADS system in an IF-MAP environment. The graph consists of two logical areas: the upper area which represents standard IF-MAP elements and the lower which is CADS-specific. The standard IF-MAP part in the upper area is constructed through the use of other IF-MAP based components. Such components can be for example an IF-MAP enabled flow controller or a sensor capable of publishing IF-MAP metadata. CADS components are only able to operate on CADS-specific data (lower area), as they only encapsulate the necessary information, like contextual information for the Features. This means, to use standard metadata, a special MAPC would be required in order to translate these standard metadata in the CADS-specific Feature format. An easier approach to do this, is to simply extend the appropriate MAPC with the functionality to directly

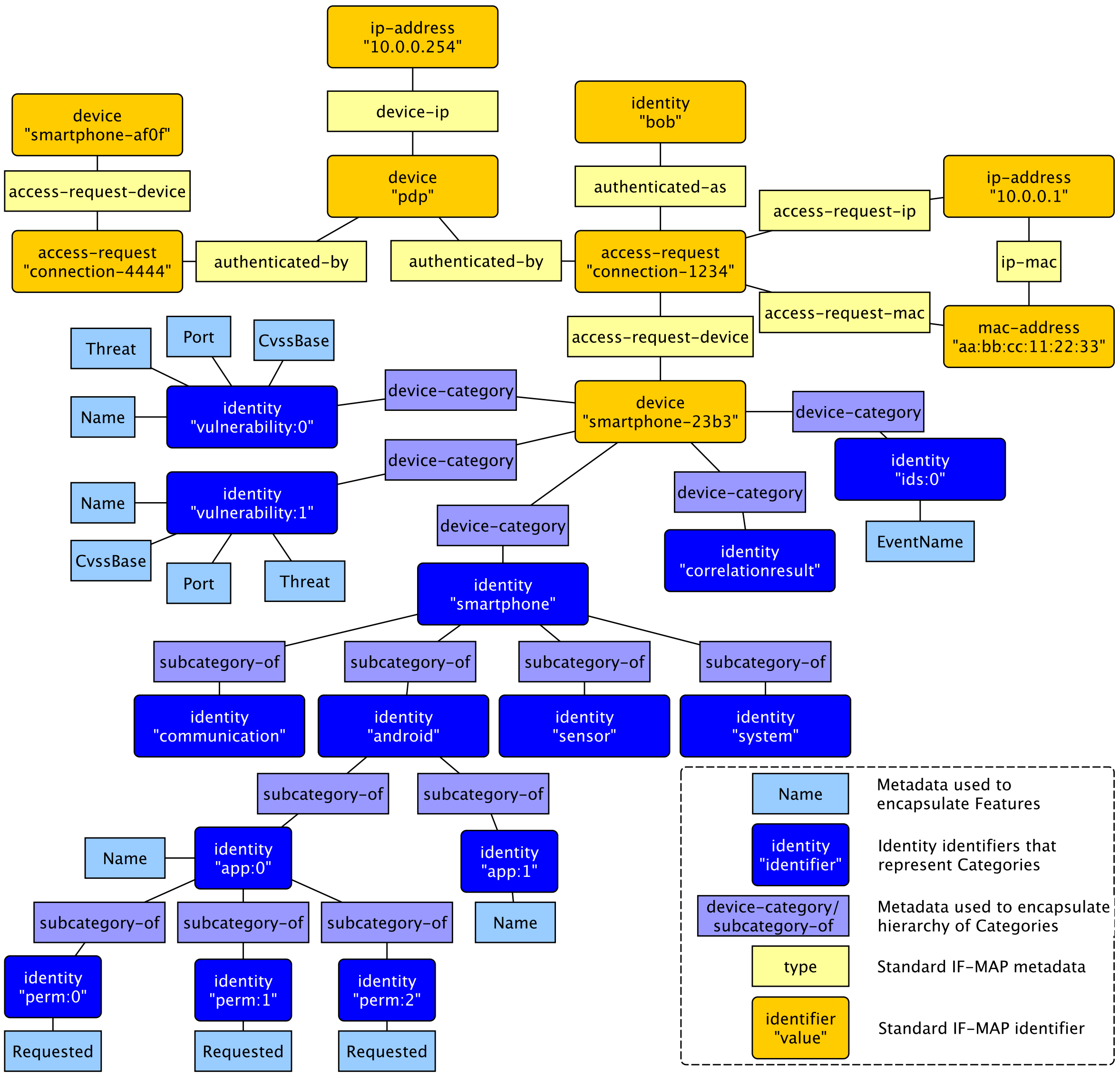


Figure 3.10: IF-MAP example graph with CADS elements [2].

publish Feature-based metadata. Although CADS can only operate on its own elements, the Correlation Engine is capable of publishing back standard metadata under some circumstances. Due to this, CADS may also act as a decision engine providing decisions for standard IF-MAP enabled components. Furthermore, the graph being created by the use of the CADS components is only usable by components which are able to understand the Feature-based semantic. This is only given for logical components which are either a

Correlation Engine or a Feature Consumer. That is, if another component is intended to be included in the CADS graph, with the term included referring to the ability to use the Feature-based metadata, it needs to be either defined as Correlation Engine or as Feature Consumer, whereas Feature Consumer is more likely.

Given that, the communicational flow created by the CADS component can be shown in a more revised version. Figure 3.11 depicts this flow with the steps being explained in the following. The components which are involved are two Feature Collectors (FCol-1/2), the Feature Provider (FP), the Correlation Engine (CE) and one Feature Consumer (FCon).

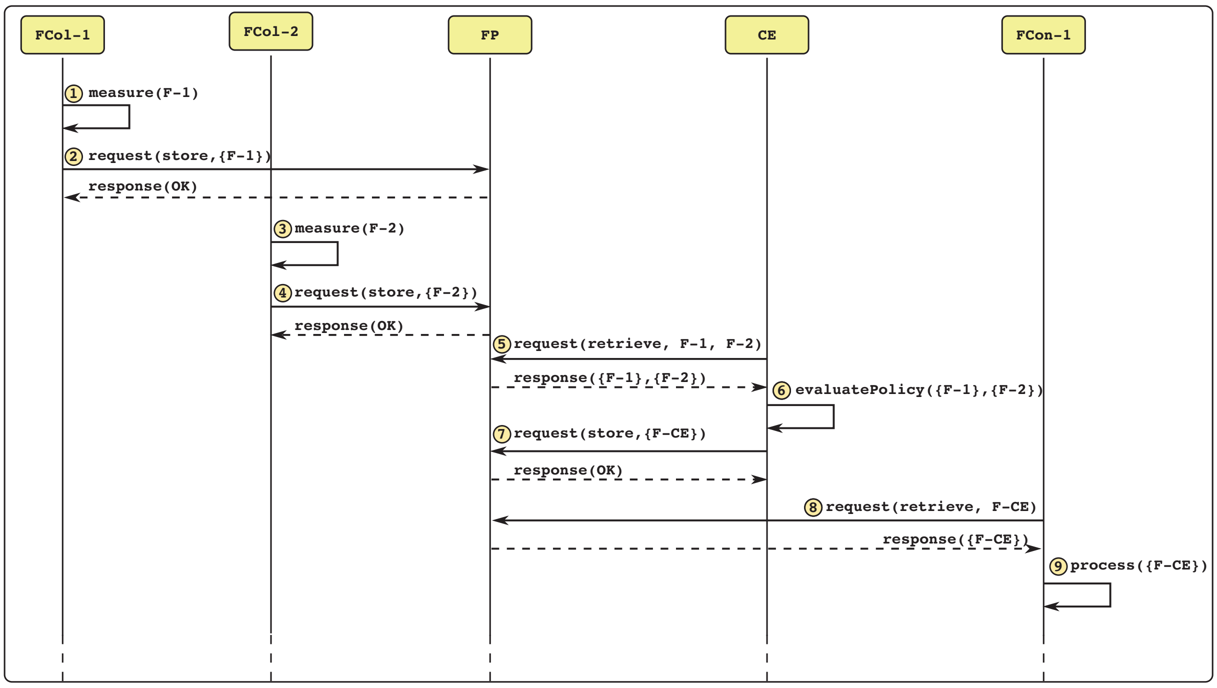


Figure 3.11: CADS communication flow [2].

1. The first Feature Collector starts by measuring a certain property of the device and creates a Feature for this measurement.

2. After that, the Feature Collector request storage for this Feature on the Feature Provider. In detail, this is done by publishing this Feature as IF-MAP metadata and providing it to the MAPS which includes it in the CADS-specific sub graph.

3. The second Feature Collector continues by measuring another Feature.

4. This Feature Collector also requests the Feature Provider to store the previously measured Feature. As with the first Feature Collector, this is done leveraging IF-MAP and performing a publish into the appropriate sub graph on the MAPS.

5. The Correlation Engine retrieves both stored Features from the Feature Provider. Seen from IF-MAP this is done due to the use of a subscription from the Correlation Engine for CADS-specific metadata. That is, each time CADS metadata (Features) are published by a Feature Collector, the Correlation Engine is triggered and is provided with the Features from the MAPS.

6. Using this data, the Correlation Engine is now able to evaluate the Features against the policy. This policy defines actions based on certain measurement values and their combination, additionally including contextual information. Based on this evaluation, a new Feature which expresses the Correlation Engine's decision is created.

7. This decision holding Feature is then again stored (i.e. published to and stored) by the Feature Provider.

8. A Feature Consumer, which has an appropriate subscription for this kind of Feature-based decision metadata is now able to retrieve this decision.

9. Using this received Feature for an arbitrary purpose, the final step is finished.

As these steps and the CADS mapped architecture shows, IF-MAP provides a rather straightforward way of using the CADS concept. By providing the necessary mapping for the appropriate components and the definition of the Feature-based metadata format, the IF-MAP based implementation approach can be easily extended. This can be used to fulfil R5 and R6 when providing a proof of concept implementation.

3.2.4 Summary

Trusted Computing provides two basic abilities: the trust derivation approaches and the framework-based approaches which are conceptional as well as implementation centric. The trust derivation is given by the concept of a trusted platform possessing several capabilities which allow to put trust into this platform. Furthermore, trust is not only derived from software but founded by several hardware-based parts forming a rather strong root of trust. Besides the TCG standardised approaches, there has been some work in this area, not only for the root of trust itself but also in the area of attestation. This work shows several promising aspects, which may be leveraged in order to establish a mechanism for evaluating trust, thus fulfilling R1. Unfortunately, neither the TCG's standardised approach nor the research-based approaches provide a way to perform a trust evaluation in the reference environment out of the box. Furthermore, none of the approaches fulfils the other trust-related requirements R2 and R3. A policy (R4) is given and explained by some of the approaches. However, all of the approaches tailor their policy for the selected use case.

The framework-based approaches provided by Trusted Computing are used to allow for a high level of interoperability, thus aiming in the direction of R5 and R6. Although not directly related to the trust-based requirements, it is rather promising as it allows to implement systems which can be integrated into existing architectures. This is due to the fact, that the new component simply needs to implement this standard as well if the environment the component should be included is based on the standardised approach. While not directly useful for the conceptual approach, this benefit needs to be considered when actually building and implementing the system. In detail, the current implementation of the CADS system is based on such a standard and therefore provides a solution R6.

3.3 Assessment

Based on the findings made for the related work in both parts: research- and the technology-based approaches, this section summarises the most important points for each of the requirements stated in section 2. First of all, an overview table shows both, research- and technology-related approaches and their particular fulfilment of the requirements. Based on this, a detailed discussion of each requirement summarises the chapter.

The following table (3.1) summarises the fulfilment of the requirements for the most important approaches shown in the previous sections. The approach is either named or referenced by the authors who proposed it. Approaches are aggregated into one single entry if there are more than one version of it, i.e. the entry summarises all features of the particular versions of this approach. The TCG-based approaches are categorised by either they are TNC-related or not. A — indicates a non fulfilled requirement, a O indicates a partly fulfilment and a + a complete requirement fulfilment. It is clearly to see, that there are a lot of useful approaches which address some of the requirements. However, there is no approach which is able to fulfil all of the requirements.

Approach	*Trust evaluation (R1)*	*Trust history (R2)*	*Trust correlation (R3)*	*Policy extension (R4)*	*Extensibility (R5)*	*Integrability (R6)*
CADS [2]	—	O	O	O	O	+
TIDS [71]	O	—	O	O	—	O
SABER [52]	—	—	O	O	O	O
Yeom et al.[72]	—	—	—	—	O	+
Sun et al. [84]	O	O	—	O	—	O
HADOF [85]	O	O	O	—	—	—
Bhargava et al. [81]	O	—	—	—	—	O
Hailes et al. [59]	O	+	—	—	O	—
SOCBox [41]	—	O	O	O	O	—
DSOC[42]	—	O	O	O	O	—
Duma et al.[54]	O	O	O	—	—	—
CRIM [47]	—	O	O	O	O	—
Zhou et al.[45]	—	O	O	O	—	—
Papalilo et al. [82]	O	—	—	O	—	—
Bao et al. [86]	O	O	—	O	—	O
Fung et al.[58]	O	O	—	—	—	—
Liu et al. [75]	O	—	O	—	—	—
Chang et al. [90]	O	O	—	O	—	O
Azzedin et al. [77]	O	O	—	O	—	—
STDEM [91]	O	O	—	—	—	O
Ramachandran et al.[43]	—	O	O	—	—	—
PBA [92]	+	—	—	O	O	O
Vijayakumar et al. [101]	+	—	—	O	—	O
IDMTM [89]	O	—	—	O	—	O
TCG's Trusted Platform	+	O	—	—	O	—
TCG's Trusted Network Connect	—	—	—	O	O	+

Table 3.1: Fulfilment of the requirements (R1-R6) by the most important approaches presented in this chapter.

In general, it can be easily recognised, that the CADS approach already provides means to include smartphones within a company's infrastructure. Besides that, it also fulfils some of the requirements partially or provides a very good basis for their solution. Although it does not incorporate trust, it provides a powerful decision making system and was designed with reusability and integrability already in mind. Furthermore, it can be summarised that there is no single approach fulfilling all of the requirements in a satisfying way. Given this, it is necessary to develop an own approach which is capable of fulfilling the requirements. A detailed assessment against the requirements is given in the following. It does not only emphasise the problems with the shown approaches but does also present aspects which may be used in the own approach which needs to be developed.

R1: Trust specification, calculation and evaluation In order to fulfil this requirement, an approach must be able to allow a specification, a derivation and a calculation of trust as well as provide an evaluation method for this trust. Given the approaches presented in this chapter, it can be summarised that one group does not address this problem while the other group provides means to evaluate the trustworthiness of components. As the first group does not address this problem, because their information collecting components are treated inherently trustworthy, it cannot provide a solution to fulfil this requirement. The second group cares about trustworthiness of their sensory components. This is mostly demanded by the scenarios of the single approaches. However, they do not directly address the trustworthiness of the collected data but the trustworthiness of the components collecting this data. Given this requirement, it is necessary to address trustworthiness for the collected data and not only the component as this kind of view does not address the communicational part which is responsible for exchanging the collected data. Besides that, there are some reasonable approaches about deriving trustworthiness for a component. They may be included as part of an overall solution. The TCG-based approaches offer the same benefits: a reliable method for deriving trust for a certain component (i.e. a system being part of a network). Furthermore, the trust derivation is based on a hardware secured root of trust, which can be considered very reliable. However, as with the other approaches, the communication channel itself is not taken into account. While the trust derivation in form of a remote attestation itself is secured by signing means, which is necessary to provide the proof for a valid TPM, a measurement of a sensor of such a system is not directly addressed. As it is the same for the research-based approaches, a rather useful method for deriving system and component-based trust is given while there is no way to cover the whole data exchange process. Given all these points, it can be summarised, that while there are promising approaches to allow for a component-based trust evaluation, there is no approach combining both component and communication-based trust. Due to this, a solution to fulfil this requirement allowing an effective trust evaluation may be based on some of the approaches but must incorporate more sophisticated means in order to completely fulfil this requirement.

R2: Trust history To fulfil this requirement, an approach needs to be able to (1) keep a history of the collected data (i.e. the measurements taken by the particular sensors) and (2) a history of the trust assigned to this values. Nearly all of the research centric approaches presented in section 3.1.3 do not provide a direct method to fulfil this requirement. While some of them are able to keep track of updates of measurement values (1), thus effectively allowing to establish a history, they do not allow to keep track of the change in trust. This is due to two limitations: either the approach does not incorporate trust or trust is only derived for the currently stored measurement values. However, as table 3.1 and the particular explanations show, there are two approaches which provide a basic history function including trust. Unfortunately, these two approaches lack in fulfilling other requirements, in particular they are either not integrable or not extensible. The approaches presented by the Trusted Computing concept and the related research ideas do also not address this. This is because both ideas do not aim at recording measurement values but more on attesting a particular state of a system at a certain time, without keeping track of the state change of that system. Summarising this, to fulfil this requirement, the solution must be able to support both: the trivial case of keeping track of data changes (i.e. new measurement values) as well as the more sophisticated case of keeping track of this data's trust changes.

R3: Trust-based correlation To allow the decision making component provided by the IDS in the reference infrastructure a sophisticated process of decision making, the IDS itself need to be able to directly use the trustworthiness for their calculation. That is, the IDS should not only be able to distinguish between trustworthy and untrustworthy sensor data but also being able to use the actual value of trustworthiness for their decision making process. Given the research-based approaches, there are some which support this kind of correlation. In detail, approaches used for intrusion detection in distributed networks that depend on trustworthiness of their nodes (e.g. in sensor networks), the nodes trustworthiness is indeed used to derive intrusion detection decisions. These approaches therefore calculate a node's trustworthiness and correlate upon this trustworthiness. This provides a way of fulfilling this requirement. However, all of these approaches have several constraints, like the particular environments or the very limited trust derivation. Due to this, they can render a promising base for building a solution fulfilling this requirement. Besides that, the TCG-based approach provides no way of correlating values directly. This is due to the fact, that their main goal is to provide a mean to judge about the integrity state of a particular platform. As they do not provide a correlation method, they cannot be used as a solution to fulfil this requirement. However, it can be summarised that there are approaches which allow to provide means for a trust-based correlation. They need to be tailored into an overall solution.

R4: Extending policies with trust This requirement summarises the possibility to actually use the trust related functions in a policy-based way. That is, by fulfilling this requirement, it is possible to define arbitrary trust related constraints within one or more policies. As the reference infrastructure used by the scenarios already provides a policy which controls the details of its decision making process (i.e. the IDS' policy), it must be possible to also address the trust issues within this policy or at least in some way to influence the decision making process. Most of the approaches presented in the related work section support such a policy in a more or less effective form. However, this policy is always tailored for the specific scenario given in that approach. There is no generalised version to be found in the approaches. Although that means that there must be a particular new solution to fulfil this requirement, the ideas which are described in the approaches may be used. Given the TCG-based approaches it is a rather similar situation. Although there are means of defining constraints and expressing rules in a policy defined, they do not directly address this requirement here. This is not only due to the particular scenario of the Trusted Computing approach but also due to the properties which should be expressible in that policy. As the TCG approach does not aim onto the part of decision making but only on attesting the trustworthiness of a platform, it is not sufficient to simply use the TCG's policy approach. The same limitation affects the attestation-based approaches: most of them use somehow a policy, but this policy is insufficient for expressing a trust-based reaction. It is therefore necessary to define a rather tailored policy which incorporates the trust-specific functions.

R5: Extensibility in terms of used data and trust calculation methods As sensors should be placed on arbitrary systems in the network which can have arbitrary properties, the trust derivation must be very flexible. That is, it must be able to define it also in an arbitrary way to be able to use sensors from different, unlimited in terms of their type, systems. Due to this, the method used for calculation must also not be defined statically. Without this, it would be impossible to switch from one trust deriving method to another, as the algorithm responsible for actually calculating the trustworthiness would be unable to work on the changed method. Due to this, it is necessary to provide means for using flexible data sources to derive trust and to be flexible in terms of the actual trust calculation. Taking these requirements and evaluating it against the research-based approaches, it is very clearly to see, that most of the approaches are rather limited in terms of extensibility. Although some are designed with extensibility in mind, they miss the trust consideration, thus only providing extensibility in terms of the data of sensors. The approaches which keep trust in mind allow for adding a limited set of means of deriving trust, in particular through adding reputation trust. Due to this, there is no approach which is able to define the trust derivation factors in an arbitrary way. This limitation is even stronger when evaluating the calculation methods used by the approaches. Nearly all approaches stick to one single algorithm, which may be rather complex but not interchangeable. This leads to the fact, that some approaches come in three or more versions which differ just by small changes in the calculation algorithm. To circumvent this, a flexible and freely definition of this algorithm is necessary, also if considering the effectiveness of such an algorithm. As research results show, most algorithms are improved over time. Given the attestation-based approaches, most of them are very freely in their definition of trust deriving sources. This is due to the fact, that the attestation approach is not bound to a particular set of properties but rather on measuring everything valuable on the platform. Due to this, the properties used for deriving the trustworthiness of this platform may be defined by combining components that should be measured. This really allows for the use of arbitrary system properties. Due to this, the TCG-based approaches, mainly the attestation approaches, provide a good building block to construct a solution that fulfils this requirement, although they lack the definition of an actual trust calculation algorithm.

R6: Ability of seamless integration To allow an integration into an existing infrastructure, in particular into the reference infrastructure, this requirement must be fulfilled. Looking first at the research approaches, nearly none of them aims to provide a integrable solution. This is due to the fact, as they aim on a particular problem for a particular environment, which is very common for this kind of contribution. All of these approaches would need to be tailored and generalised in a rather complex manner in order to make them integrable. Given the TCG approaches, this is a little bit different. Due to the TCG approaches addressing the requirement of interoperability, especially when evaluating the framework-based approaches provided by the TNC specification, they are integrable by design.[3] As the current CADS implementation is already based upon a TCG defined specification, it can be summarised that these specifications fulfil the requirement of a seamless integration to a high level. Due to this, the TCG's approach provide a good technological building block in order to fulfil this requirement.

Given the evaluation and table 3.1, it is clearly to see that there is no approach known which fulfils all requirements in a satisfying manner. However, there are some concepts, which provide a good start and seem very promising for the inclusion into an overall concept. In particular, the Trusted Computing-based approaches provide reasonable mechanisms in terms of trust specification and derivation (R1) as they introduce the concept of remote attestation which allows to measure a platform's state, thus providing a mean of trust derivation for this platform. Besides that, approaches like Property-based Attestation ([92], cf. [101]) provide a more refined version of the remote attestation approach, particularly a more flexible trust derivation (R1), which will be considered within the concept. Furthermore, the only approach which provides already a smartphone centric

[3]With the integrability based on the requirements given by the TCG.

view combined with a decision making system and a high level of integrability as well as extensibility is the CADS approach (cf. 3.1). As explained, it was designed to provide means to transparently use smartphones in business environments, however without the consideration of trust. While not addressing R1, CADS fulfils R2 to R5 partially as well as R6 completely. Due to this, CADS provides some basic concepts which can be considered very helpful in order to completely fulfil all of the requirements named here. Given that, CADS is chosen as the basis for a domain-specific solution which fulfils the requirements. This does not imply, that the concept presented in the next chapter is only usable with the CADS approach. On the contrary, CADS is only one possibility of a domain-specific mapping. That is, differently to the approaches named in this chapter, the overall concept aims on fulfilling all requirements. Doing so, the reference infrastructure is enhanced in a way, that it is able to deal with trust in a sophisticated manner. It is not only able to evaluate the trustworthiness of collected data binary-based but also to correlate on this trust keeping track of value changes. Furthermore, it allows to express trust within its policies and provides the same integrability the CADS system does. A concept which provides all these possibilities is developed in the following chapter.

4 A Concept for Trustworthy Smartphone Integration

Contents

The following chapter offers a solution to the problems shown in the previous chapters. The solution itself is divided into three parts: (1) the definition of a generic model, (2) the definition of the trust model relying on the previously defined model and finally (3) the domain-specific mapping.

4.1 Generic Model

This section defines a generic role model and associated operations which can be carried out by the roles. Furthermore, a mapping process of the generic elements is given.

4.1.1 Role and Operational Model

The following section describes the generalised model, which allows a decoupling of actual components and their actions. Figure 4.1 shows the model in detail. As it is easy to see,

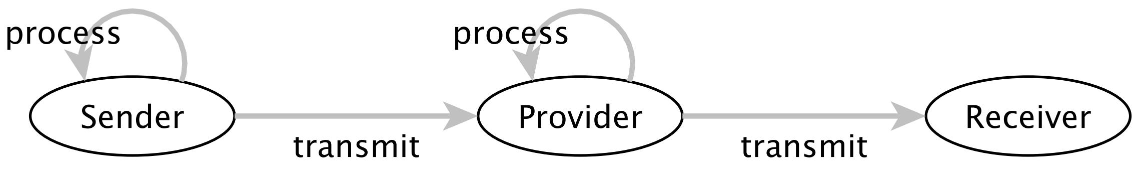


Figure 4.1: Generic Model.

the model operates not on particular components but on generic roles. These generic roles are case-specific assigned to each relevant component. Besides the roles, there are also two types of generic operations defined. These operations are responsible for the actual part of working on the data collected. The collected data represents the device's state, i.e. it is data from a sensor monitoring the device. It is therefore defined as so-called Feature. The definition of the term Feature was already given in 3.1.1, based on the definition made in [2]. The term is used for the remainder of this thesis.

Roles

In order to decouple the components from actions, the model defines three distinct roles: (1) the sender, (2) the provider and (3) the receiver role. They are explained in detail in the following.

Sender The role of the sender is mapped to components which are responsible of the initial creation and propagation of a Feature. In detail the sender performs three actual tasks.

Feature Creation This is the first sub-step of the overall model. It takes place when a Feature is initially created, whereas created means measuring a Feature or aggregating several Features into one. Measuring refers to the process of receiving properties of a given system and putting them into the appropriate Feature type. Aggregating refers to the process of using already existing Features, which may have been measured before, and combining their values into a single one inside a newly created Feature.

Feature Preparation The preparation sub-step takes place after the Feature was successfully created. It is necessary to allow a further processing of the Feature. In detail, the created Feature will be prepared for the transmission within this step. This process consists of making the Feature available until the actual transmission step will be initiated and if the transmission step is about to start creating an actual transmittable representation of the Feature. This representation, i.e. the message which will be communicated later, may consist of more than one single Feature.

Feature Transmission The final sub-step the sender is responsible for is the actual transmission of the prepared message consisting of the Feature. To perform this step, the sender simply selects the provider the message should be transmitted to and performs the task of communicating the message via a network-based connection.

Given these steps, the role of the sender is mapped to each actual component which is responsible for measuring and collecting Features.

Provider The second role is the role of the Provider. A Provider is responsible for receiving Features from arbitrary Senders and preparing them for further processing. To perform this task, the Provider distinguishes it work into three sub-steps.

Receipt of Feature After the Sender has started the Feature transmission, this sub-step can be performed on the Provider. The transmitted message will be received and one or more Features which are marshalled within the message can be extracted. The communication which needs to be carried out in this step is not limited to the sending and receiving process described above, but can also be carried out in a more complex way, including an arbitrary amount of communication steps.

Feature Storage and Preparation This step takes responsibility for storing the extracted Feature in an appropriate way and to reload and prepare the Feature if requested. Storage is simply performed by putting the Feature into an appropriate container whilst reloading takes the Feature out of this container. After reloading, the Feature is prepared into a message equalling the message type that was used by the Sender in its preparation step.

Feature Transmission The prepared message, which may again consist of several Features, can be transmitted to the receiver. This is done in this last sub-step.

Receiver The last role assigned to the component which finally receives the Feature and uses it is called Receiver. The working process of the receiver is divided into two separate sub-steps.

Feature Receipt This step equals the receiving sub-step carried out on the Provider. That is, the Receiver needs to unmarshall the message which was received from the Provider thus extracting the contained Features. After the Features are extracted, the Features can be further processed which is done in the next sub-step.

Feature Processing The Receiver uses the Features that were extracted in the last sub-step to perform arbitrary tasks. The exact process the Features are used to, depends on the actual Receiver. The model itself does not limit the types of processing which may be done within this step.

Having these three roles, it is possible to express specific behaviour within one model. Specific behaviour means, that in one case a Feature may be sent from component A via component B to its destination which may be component C. Another case with the same components may be that a Feature flows from C via B to A. Using the generic model, the role-based flow is always the same independently from the actual components. This allows to base further definitions upon this model without the need to distinguish between the particular cases. Although the roles are sufficient to map the case-specific components, the actual operations which are performed between the components (or in general between the roles) need to be defined as well.

Operations

Figure 4.1 depicts the roles as well as the operations which are carried out between and by them. There are two kind of generalised, i.e. abstract, operations: (1) the so-called process operation and (2) the so-called transmit operation. The properties of these operations are explained in the following.

Process The process operation is carried out by and on a particular component. It describes the action of processing a Feature. Processing can be distinguished into the following types.

- Creation of the Feature itself. This can, as already described, be realised by measuring certain properties and generating a new Feature or by aggregating already existing Features into a new one.
- Storing a Feature for further actions and retrieving a stored Feature out of a storage container.
- Using the Feature for arbitrary actions.

Although a generic process operation must be of one of these types, it is important, that it can only be of one and the same type to a certain point of time. That is, it is not possible to combine several types into one single process operation. Besides this, it is easy to see, that these types of the process operation summarise a subset of the role-specific steps described in section 4.1.1. The missing steps are assigned to the second type of generic operation.

Transmit The transmit operation defines the communication-based transmission of a Feature. In contrast to the process operation, two components (i.e. roles) are part of this operation. It consists of the following steps.

- The preparation of the Feature which is intended to be transmitted. This step summarises the task of taking one or more Features and marshalling them together into one, transmittable message. As a precondition of this step, a process operation may have happened.
- The actual transmission step. Based on an arbitrary communication protocol, the created message is sent from the first component to the second one.
- The second component receives the message and unmarshalls the appropriate Features out of the message. A process operation may happen after this step.

In contrast to the process operation, which is of one actual type, the transmit operation always consists of this three steps. That is, all three steps in this exact order need to be carried out.

Combining these process and transmit operations allows to describe the handling procedure of a Feature within a particular system like CADS in an abstract manner. All possible cases of the Feature flow can be expressed by using the operations defined. Using the roles and the operations together, an abstract system can be formulated which is able to represent the actual system used.

4.1.2 Operational Flow

The generalised flow, which expresses all distinct Feature flow cases, consists of the following steps.

1. The creation or aggregation of the Feature and its additional preparation is the first step. It is performed within the $process_{Sender}$ operation running on the Sender.

2. After the Feature is prepared for the communication process, it will be communicated by the Sender to the Provider. The Provider receives the Feature and prepares it for further usage. The $transmit_{Sender}$ operation is responsible for this task.

3. After successfully receiving the Feature, storing and reloading the Feature on the Provider is handled within the $process_{Provider}$ operation.

4. Finally, the communication between the Provider and the Receiver is performed within the $transmit_{Provider}$ operation. It is responsible for transmitting the Feature from the Provider to the Receiver.

These steps summarise the flow of operations within the model.

Formal Representation

The representation consist of three different parts: (1) the roles and (2) operations which where explained above as well as (3) a formal representation of the Feature. The roles (1) are simply represented by their appropriate name, thus the Sender is represented as $Sender$, the Provider is assigned $Provider$ and the Receiver is represented as $Receiver$. As already used, the operations are defined as their abstract version mapped onto a specific role. That is, the operations (2) are defined as $process_{Sender}$, $process_{Provider}$, $transmit_{Sender}$, $transmit_{Provider}$. Following the explanation made above, there is no final process operation running on the Receiver defined. The Feature itself (3) is represented by a ζ, following the notion used in [2]. The formal representation assumes, that all operations are able to work directly upon and with the Feature. This means, that the actual method used for communication (i.e. the protocol and the protocol data) is not part of this representation and needs to be addressed when mapping this representation to the particular case or scenario. Besides this limitation, the flow of operations can now be expressed in the following manner by using this representation.

$$process_{Sender} \xrightarrow{\zeta} transmit_{Sender} \xrightarrow{\zeta} process_{Provider} \xrightarrow{\zeta} transmit_{Provider}$$

This representation is used later in this thesis to show the particular trust related extensions to the flow. As it is easy to recognise, all operations take only one argument. While this is fully correct for the process operations as they are working on one component (or on a role) it is an abbreviation for the transmit operation. The representation of the transmit operation only takes the sending source as argument. This is possible because the model defines the destination of the transmit operation uniquely according to the transmission source.

This formal representation allows to extend the model easily, for example by simply adding more operations or roles. Furthermore, it is a necessary building block for the following chapters.

4.1.3 CADS-specific Feature handling

As explained in section 3.3, the CADS approach is well tailored for smartphones and provides a basic decision making system which can be used in order to provide a domain-specific extension. To allow this extension, the model presented here must be applicable to the CADS approach. This means, the CADS system must be expressible using the generic model. This section provides a mapping to not only show this possibility of mapping but only to provide a foundation for the domain-specific extension which is addressed later in this chapter. Furthermore, it can be considered as an example on how to map a particular system using the generic model.

Within the CADS architecture (section 3.1.1), there are two distinct cases of Feature handling, thus there are two cases how a Feature may flow through the system. The first case describes the situation where a Feature Collector is sending data to the Feature Provider and finally to the Correlation Engine. The second case takes place if the Correlation Engine is sending a Feature to a specific Feature Consumer.

Case I

The first case describes the situation, where a Feature Collector sends data to the Correlation Engine. This for example could happen, if a Feature Collector located on a smartphone measures certain elements of the device and wants to send this measurements to the Correlation Engine. After the Correlation Engine receives this measurements, it can perform arbitrary tasks upon this data.

The case itself is divided into several independent steps, which are carried out in a serial order. That is, the ordering of these steps is always the same.

- The first step is the creation of the Feature on the Feature Collector's device. This can be done by simply measuring a certain property of the device using the Feature Collector or by aggregating other Features which have already been measured. After finishing this step, the Feature which is to be sent to the Correlation Engine is prepared for the next step.

- The Feature Collector is now able to send the Feature to the Feature Provider. In detail, this is done by using an appropriate communication method in order to transmit the Feature from the Collector to the Feature Provider. Although it is not defined which type of communication is used as this is a domain-specific property, it is defined that this step is a logical step. That is, the transmission itself must not necessarily be performed by using a real network-based communication method but can also be done by using an inter process-based communication type for example.

  As a result of this step, the Feature Provider now possesses the Feature.

- Having the Feature successfully received, the Feature Provider now stores the Feature until the Correlation Engine requests the Feature for further processing. If it is requested, the Feature Provider retrieves it from its database and prepares it for the next step.

- The next step is the transmission of the Feature from the Feature Provider to the Correlation Engine. This step equals the first transmission step, where the Feature Collector sends the Feature to the Feature Provider.

- Finally, after the Correlation Engine received the Feature, it can process the Feature further.

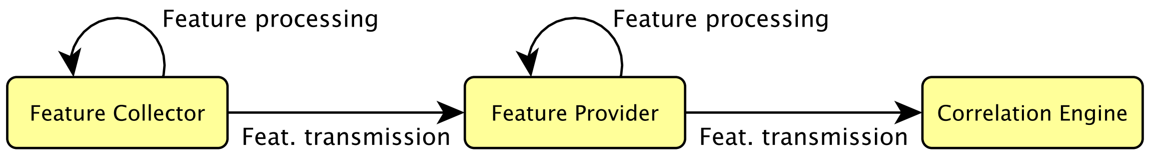


Figure 4.2: CADS Case I.

This process of collecting, preparing and transmitting the Feature is summarised in figure 4.2.

Case II

Beside the situation where one or more Feature Collectors are sending their collected Features to the Correlation Engine, there might be the case of sending a Feature originating from the Correlation Engine. This could for example happen after the Correlation Engine has evaluated some rules and one of the rules is triggering an enforcement, thus creating an enforcement Feature which is sent out to a device capable of performing the actual enforcement.

Details of this case are as follows.

- At first, the Correlation Engine needs to create the appropriate Feature. This can be done by simply generating a completely new Feature, or by aggregating already existing Features into another Feature. After the Feature is created, it is being prepared for the next step.

- The Correlation Engine sends out the Feature to the Feature Provider. Besides a different sending source, this step equals the transmission steps of the first case. That is, after this step has been carried out, the Feature is ready for further processing on the Feature Provider.

- As the Feature Provider successfully received the Feature, it can be stored and kept ready for transmission to the next component.

- When the Feature is requested by the Feature Consumer, the Feature Provider can now send it to the Consumer. As all transmission steps, besides changing sources and destinations, it is equal to the other transmission steps.

- After the Feature was received by the Feature Consumer, it may be further processed and used.

The overall flow of these steps is summarised in figure 4.3. As the generic steps which have to be carried out are equal in both cases, the flow can be generalised.

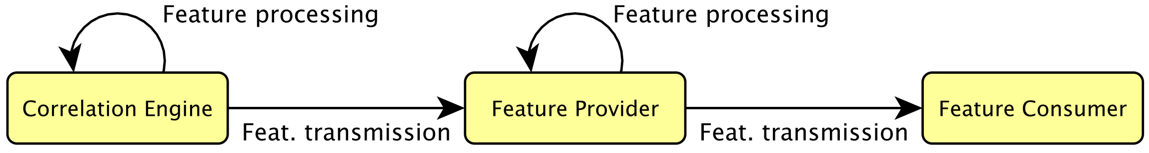


Figure 4.3: CADS Case II.

Generalised Flow

As stated above, the steps and their associated operations are to some extent equal in both cases. Due to this, the overall process (i.e. the steps necessary) of Feature creation/processing and Feature transmission can be expressed in a more generic way:

- The first step takes responsibility for creating the Feature on the appropriate component. This is done by either aggregating already measured Features or by creating a new Feature. Besides the creation of the Features, this step is also responsible for taking the Feature in a transmission-ready state.
- The second step describes the transmission process between the first component and the Feature Provider. That is, after the Feature was created and prepared for transmission within the first step, it actually gets transmitted within this step.
- As third, the Feature is stored on the Feature Provider and after some time again prepared for the second transmission to the receiving component.
- Finally, the Feature is sent by the Feature Provider to the receiving component.

In contrast to Case I and II, there is no final step of Feature processing on the receiving component. This is due to the fact, that (1) a Feature processing may not happen on this component and (2) that the processing itself is out of scope.

While this generalised flow offers a possibility to express the Feature processing with one notation, it is not possible to use generalised components. Due to this, it must still be distinguished between the specific component used at this point. For example, it has to be distinguished between the Correlation Engine and the Feature Collector as sending component. As defining a Trust Model would also imply to distinguish between those components and thus limit the usability and flexibility of the model, a further generalisation is necessary at this point. This generalisation was already given by the abstract model above. Due to this, the next section shows how to map this generic model to the particular CADS cases, thus showing the process of instantiation.

Role and Operation-specific mapping

To allow definitions based on the generic model, the roles and operations need to be mapped upon the CADS-specific cases.

Mapping roles to components Using the abstract model, the components defined within the CADS system can be mapped to roles as follows.

- The component Feature Collector always takes the role of a Sender, thus it is not important which actual case is handled by the Collector.
- As already outlined, the Feature Provider is always in the role of the Provider. Equal to the Feature Collector's mapping, this role never changes throughout different cases.
- The Correlation Engine cannot be mapped to only one role in a static manner. That is, the Correlation Engine is either a Receiver or a Sender. It is a Receiver if and only if it is used within a case I specific scenario. If it used in a case II-specific scenario, it operates as a Sender.
- The last component, the Feature Consumer, always takes the role of a Receiver. Like every other component but excluding the Correlation Engine, this assignment is independent from the actual case.

Having these mappings, definitions describing the CADS system can now be formulated in a generic manner. As this mapping only reflects the relationship between components and roles but not their operations, an operation-specific mapping needs to be defined.

Operational Mapping To map the actual CADS cases, the model defines four particular operations based on the abstract process and transmit operations: two process operations and two transmit operations. The process operations are bound to the Sender and the

Provider, while transmit is carried out between the Sender and the Provider as well as between the Provider and the Receiver. They are defined as the following.

- The measurement or creation of a Feature, i.e. the measurement operation which the actual Feature Collector performs in its role as Sender, is called $process_{Sender}$.
- The operation transmitting the Feature from the Sender to the Provider, which may be for example the communication between the Feature Collector and the Feature Provider, is called $transmit_{Sender}$.
- As next, the operation used is the $process_{Provider}$ operation, which is performed after the Feature was received on the Provider. This operation is usually carried out on the Feature Provider.
- Similar to the first transmission operation, the operation which is used by the Receiver to retrieve the Feature from the Provider is called $transmit_{Provider}$. This happens for example, if the Feature Consumer receives a Feature from the Feature Provider.

By using these operations and including them in the model, thus gathering roles and their operations, it is possible to express the CADS-specific cases in an abstract manner. Furthermore, this model may be used to add new cases to the CADS system. Besides that, it is now possible to express a generalised flow which is based upon the roles as acting parties and their operations as actions which will be performed.

As easy to see, there is no final process operation defined for the Receiver. Due to this, the model does not define the final step of Feature processing. This is necessary to allow arbitrary processing methods to be included within the model to not limit the applicable cases for the model. It does not define that there is no final process operation allowed but defines that the operation performed may be arbitrary in any form.

4.1.4 Model Summary

To represent arbitrary cases of Feature flows, the model developed above can be used. It consists of two generalised parts: an abstract role definition where roles can be mapped to particular components in a case-specific manner. And, as the second part, two abstract types of operations: process and transmit. Putting the roles and the operations together, a generalised flow can be formulated. The developed flow is able to represent particular specific cases without the need to address the change of the actual components used. That is, it is possible to develop further extensions in terms of trustworthiness based on this model without the need to address each case separately. Besides that, the model defines also a more formal representation.

Giving an example of the instantiation process of the model, the two CADS-specific cases can be mapped into a generic representation. This allows it to define further mechanisms, in particular the trust model, up on the generic model with the effect of the definition being applicable to the particular specific system which would be CADS in this example. The case-specific mapping necessary to consider a trust extension of the CADS system in a more general way is only one area of use for the model. It may also be used as a representation for other systems which can be mapped. To express another system by using the model, the applicability of the model needs to be analysed. That is, it is necessary to define the exact mapping of operations and roles onto the system's specific properties. If it is possible to map a system, all developments which are based on the model may be assignable to the system as well.

To allow the assignment of the model to different systems, the following section gives a short guideline on how to map roles and operations onto arbitrary systems.

Instantiating

First of all, the Feature needs to be mapped into the system's environment. As the Feature represents the informational unit, with information being defined as data attached with semantic knowledge (cf. [157, 2]), it has to be instantiated for the type of information used within the system. An actual example is given later in this section. Second, the roles have to be appropriately mapped to existing components. This can be done by categorising the components based on if they are an informational source, an informational sink or a neutral component in terms of information. Sources of information should be considered as Senders, while sinks should be considered as Receivers. Neutral components need to be investigated further, based on the logical way the information flows within the system. Due to this, it is necessary to analyse this logical flow within the next step. To do this, like in the CADS system, cases have to be defined expressing the system's operations. In theory, it is necessary that all possible cases defined, thus covering the whole potential of the system. For a practical instantiation, it could be enough if those cases are defined, which should be expressed in the abstract model. Having those cases and therefore the logical flow of information, the elements (i.e. components) which are part of the flow and neither a source nor a sink can be treated as Providers. This should result in the fully defined mapping of the three roles allowing to take care about the operational mapping. This is also the direct next step, the mapping of operations. This is done in two phases: the mapping of the process operations and the mapping of the transmit operations. The process operations are defined to be carried out by the Sender and the Provider. Thus, the characteristics of the components which were identified as Sender and Provider need to be analysed. This analysis is done in terms of their impact on the information handling. By taking the explanation of the operations made in section 4.1.1 into account, it is clear that the process operation of the sender must express the creation or aggregation of the information as well as the preparation. On the Provider's side, storing and reloading needs to be defined for mapping the process operation. Completing this step, the process operations should be well defined for the system allowing to address the transmit operations. This is done in a similar manner, which means that the statements of section 4.1.1 need to be fulfilled. Transmit must be defined as the logical transmission of the information between the two components connected to this operation. It is important to understand, that transmit expresses the logical communication. This means, that if the communication of the information runs across several components, which do not alter or access the information but only communicate it, these components will not be given a specific role. Having defined all the operations, the mapping is completed and may be verified on the system.

A summarised form of the steps necessary for instantiating the model are given in the following.

1. Definition of the Feature as informational unit: it has to be defined which information is communicated throughout the system.

2. Categorisation of the system's components into (a) information sources, (b) information sinks and (c) neutral components. Components which are type (a) or (b) will be assigned the Sender and Receiver role appropriately. Components of (c)-type need to be investigated further.

3. Analysis of the informational flow within the system to isolate cases. As a result, (c)-type components can be either treated as Providers or treated as irrelevant for the model to system mapping.

4. Mapping of the process operations based on the model's definition for this operation type. Equally, mapping of the transmit operation based on their abstract definition and the flow which has already been analysed to isolate the cases.

If one ore more of these steps cannot be successfully finished, the system might not be applicable for an abstract expression based on the above defined model. In this case, there needs to be checked if the model needs to be extended to cover the system.

Example

A syslog (cf. [158])-based logging system is used as a simple example. The system consists of three components: the syslog-client (SYC) producing log messages, the syslog server (SYS) which stores and holds the logging information and a program (EMA) that takes the syslog messages and generates email-based alerts. Taking this, the system is structured as $SYC - SYS - EMA$. To map the abstract model, the first step is the Feature definition. As the system communicates syslog messages expressing certain system states, the Feature is being defined as such a syslog message. The next step is the categorisation of the components into information sources and sinks. As the system's type of information (i.e. Features) are syslog messages, it needs to be categorised appropriately: the SYC is a source while the EMA is a sink. Due to this, the SYC can be considered as Sender while the EMA can be considered as Receiver. Although the SYS may be considered a sink, it is none as the final destination is the EMA. As the system has only one relevant case in which the SYC creates a log message, sends it to the server from where the EMA takes it and generates an email, the analysis of the flow to isolate cases is rather simple. Furthermore, as there is only one (c)-type component, the SYS is given the Provider role. The mapping of the operations can also be done based on the analysis of the single case. Process on the Sender defines the creation of the syslog message by an arbitrary daemon on the SYC. Process on the SYS defines the successful receipt (not in terms of communication) and storage of the message until the EMA accesses the message to generate an alert. Transmit is defined as using the syslog protocol to communicate the message between the SYC and the SYS, which is the first transmit operation, and the communication between the SYS and the EMA, forming the second transmit operation. As it is easy to see, the second transmit does not actually map network-based communication but logical communication between the SYS and the EMA. The actual communication could be based on a pipe for example. Using this mapping, the system can be expressed as

$$process_{SYC} \xrightarrow{syslog} transmit_{SYC} \xrightarrow{syslog} process_{SYS} \xrightarrow{message} transmit_{SYS}$$

. It is easy to see, that the last transmit operation no longer operates on the syslog protocol but on the actual message (e.g., text). As explained above, this is due to the EMA using not syslog to access the message. Concluding the example, by taking this mapping, all assumptions and extension which are made on the abstract model can be applied for the syslog system as well.

Given the abstract model, which allows to express the system not longer based on their specific cases but in a more abstract manner, it is now possible to develop a trust model.

4.2 Trust Model

Using the abstract model defined in the last section, it is not possible to make trust-based assumptions in such a system. This section develops the second part of the overall solution: the trust model enhancing the abstract model. To do so, a trust definition is given first. This definition acts as a building block for all further developments as it defines uniquely what can be considered trustworthy. After this definition, the source of trust is introduced. That is, which parts of the model are used as building block to derive trust fulfilling the trust definition. With having the properties to base the trust up on, a method for combining these and to actually derive trust in a measurable manner is developed. Continuing the overall development of the trust model, special properties are addressed and followed by the definition of the process of trust derivation. As all these approaches are based on the abstract model, this model is extended in terms of the Feature flow as well as in terms of the underlying data model. That is, a new data model which is able to represent the trust model is also developed. Finally, the trust model is instantiated for the CADS system, also showing the extended CADS-specific cases.

The aim of this section is to introduce an extended version of the abstract model defined above. This version allows a reasoning about the trustworthiness of Features. It aims to fulfil the requirements given by the scenarios. The first area that needs to be addressed in order to develop the trust model is a basic trust definition. This is done in the following section.

4.2.1 Definition of Trust

To allow a reasoning about trust which is valid throughout the whole system as well as for each Feature, a trust definition needs to be made. This creates the foundation not only to allow the reasoning and evaluation itself but also the base to derive and calculate the trust. There are some definitions of trust already (cf. section 3.1.3, 3.2 and [59, 85, 87, 88, 89, 92, 101]), which can be used and extended to suite the needs as part of the approach. All of them are based more or less on the expected behaviour. Given that, the definition which is used for the trust model presented here is based upon the TCG's definition [96] which has already been explained in section 3.2. It is used here, as the TCG's definition summarises the aspect of basing trust on expected behaviour into a consolidated form. It can be considered consolidated, as the TCG itself is driven by different stakeholders that agreed upon this definition. Due to this, the definition can be considered as widely accepted. It is based on the behaviour of components, thus is only applicable for components and expresses that if a component behaves as expected it can be considered trustworthy. More in detail, as long as a component does what it is intended to, it is trustworthy, negligible if the task itself may be considered as being malicious. By using this definition, it is possible to reason about the trustworthiness of components. In this case, reasoning describes the process of taking a measurement of the trustworthiness and evaluating it against an expected value. That is, if a component acts as it is supposed to, due to the TCG's trust definition, the reasoning of the component's trustworthiness should be positive. As it is easy to see, this definition is only applicable for component-based systems, as the TCG's trusted computing environment is. For using this definition in the approach presented here, it has to be extended. This extension is done in two steps: the first step is to define what can be reasoned about as components are not applicable here. The second step is to define what is used as a measurement base for trustworthiness. While the first definition is given in the following, the second part is postponed until further elements of the trust model have been developed.

The TCG's definition reasons about the trustworthiness of components by measuring their behaviour and comparing it to the expected behaviour. As the approach only defines roles which can be mapped to actual components when instantiating the model, the trust could only be defined up on these abstract roles. This is rather difficult, as the roles do not imply a unique behaviour of the mapped components. I.e. differently behaving components might be mapped to the same role making it difficult to define their behaviour simple and uniquely. Furthermore, the TCG's definition aims to isolate the components as critical actors. That is, a compromised and untrustworthy component has always an actual influence on the system. Within the approach, this is different: a component might be compromised but Features sent out by this component could still be valid under certain circumstances. This fact shows the main difference in terms of trustworthiness of the two systems: while the components are the most important part in the TCG's system, the Feature is most important here. Due to this, the definition of trustworthiness must be based on the validity of the Feature itself. It is therefore defined, that the measurement of trust expresses the integrity of a Feature not of the components or roles. This integrity can be defined by using more properties of the model. When analysing the formalised flow $process_{Sender} \xrightarrow{\zeta} transmit_{Sender} \xrightarrow{\zeta} process_{Provider} \xrightarrow{\zeta} transmit_{Provider}$, it is easy to see that the operations are an important part in the overall Feature creation and transmission process. Due to this, the basic trust definition can be based on these operations and the TCG's behaviour-based definition: as long as the operations act as expected, the integrity of the Feature which the operations are working on does not change. Based on this, trust is defined as expected behaviour of the particular operations. Combining this with the exchanged Features, the following definition is used:
A Feature is considered trustworthy if and only if all operations handling this Feature perform as expected.
This basic definition would allow a reasoning about trust if there are means to judge about the operations behaviour in terms of handling the Feature. This is addressed in the next section.

4.2.2 Security Properties

Besides the definition of the trustworthiness itself, the most important part is the source for deriving trust. This section introduces special properties allowing this and thus forming the basic layer of the trust model. Furthermore, the question how trust is derived and where from it is being derived is addressed.

As already stated, trustworthiness describes a Feature's integrity and is derived from the behaviour of the operations which work on this Feature. To reason about the trustworthiness it is therefore necessary to introduce actual means to judge about the operations behaviour. This judgement must fulfil two criteria: (1) it must be able to express the overall behaviour of the operation and (2) it must be able to rate the behaviour. This is necessary to not only distinguish between benign and malicious behaviour but to distinguish between different grades of the particular behaviour.

The solution to fulfil criteria (1) is the introduction of so-called security properties (cf. [92, 101]). These security properties are given to each operation defined within the model. That is, there are security properties for $process_{Sender}$, $transmit_{Sender}$, $process_{Provider}$ and $transmit_{Provider}$. A single Security Property (SP) is defined as property characterising the behaviour of a particular operation. An arbitrary amount of such SPs may be assigned to a single operation. The properties are not limited and can be defined and used freely, thus allowing to define them case and instance-specific. When defining them, they have to be attached to the appropriate operation, meaning that there may be different properties for each of the operations. Examples for such security properties are the usage of Transport Layer Security [159] for the transmit operations or a platform secured by a trusted or secure boot (cf. 3.2 and [97]) for process operations. Using these properties, expected behaviour can be expressed for the operations although there is yet no way to classify this behaviour. To allow this classification and fulfil the second criteria, so-called ratings are used.

Ratings

Ratings (ω) are measurements which allow a scoring of particular security properties. That is, a rating allows to classify the behaviour expressed by a security property. In order to provide maximum flexibility, the approach does not define the codomain of the ratings nor any kind of rating type as this definition must be specifically made for the domain and the instance the system will be used within. In general, each of the defined SP is assigned an appropriate ω. That is, by using this rating, the SP is weighted in terms of it's trustworthiness. Although a codomain is not defined, ratings are allowed to have negative values, thus allowing to have SPs which express untrustworthy properties.

The relationship of the security properties and their appropriate ratings can be expressed as $SP := \omega$. This means, that each security property must have a rating attached to it. Using the examples of TLS and tboot given above, these properties could for example be characterised in the following way: $(usingTLS)_{transmit} := 5$ and $(tbootSuccess)_{process} := 10$, with 5 and 10 being the appropriate rating.

Security Property Map

The conceptual component holding all assignments of security property and rating is called Security Property Map (SPM). It is necessary to allow the usage of dynamic ratings per SP. Dynamic ratings are another basic building block of the trust model. Currently, the mapping between a security property and their rating can only be made in static manner. That is, once the rating is set and the system is running it cannot be changed. As this is very limiting and prevents changeability of ratings at system runtime, dynamic ratings and their assignment are introduced. Dynamic ratings are defined as ratings which can be changed at each time, thus they are changeable when the system is running. This allows to react on events which have influence on one or more operation's SPs. If for example, the transmit operation is secured by TLS, as shown above, the rating could be initially high. If, at some point an error of the TLS-enabling library is recognised, the rating could be dynamically lowered to indicate this event. Furthermore, the SPM holds the association of the SPs to their appropriate operations. Each property defined within is attached to one or more operation. The TLS property which was given as example, would be attached to a transmit operation while the trusted boot property could be attached to a process operation (which was already implicitly done when showing the example).

The actual structure of the SPM is divided into two parts: the static part which defines the security properties and their operational attachment and the dynamic part holding the ratings for the properties. Static must not be mixed up with final in this case, it only expresses that entries made in this section cannot be altered. Although there may be new entries added, the ones which are already defined can only be deleted. That is, a change of entries within this section is still possible, but only in the way of deleting the old entry and adding a new one. This is necessary as this section acts as key for referencing the dynamic

parts. The static part using the example introduced in the section above is defined as $SP_{PropertyIndex} : (PropertyName)_{operation} := \omega_{RatingIndex}$.
Using the example from above, the static part would expressed like the following.

$$\begin{array}{lll}
SP_1 : & (usesTLS)_{transmit_{Sender}} & := & \omega_1 \\
SP_2 : & (tbootSuccess)_{process_{Sender}} & := & \omega_2 \\
... & & & \\
SP_n : & (PropertName)_{operation} & := & \omega_{RatingIndex}
\end{array}$$

That is, it simply holds the properties and their associated operations. The rating is referenced from the dynamic part of the SPM and defined as $\omega_{RatingIndex} = RatingValue$, thus the dynamic part would look like the following when using the examples from above. Note that entries in the dynamic part might change.

$$\begin{array}{ll}
\omega_1 = & 5 \\
\omega_2 = & 10 \\
... & \\
\omega_n = & RatingValue
\end{array}$$

As it is easy to see, the references used in the static part are simply based on the index of the rating. That is, if a rating needs to be changed a lookup procedure must be performed in order to change the correct rating. The lookup is responsible for discovering the reverse mapping of the rating as the SPM defines a forward reference only. This means, that there is a direct path from a property to its rating by using the index while there is no direct path backwards. Due to this, the lookup procedure needs to traverse the properties the rating is to be changed for and select the correct ratings appropriately.

Security Property Records

By using both, the security property and its dynamic rating per operation, the basis for calculating the trustworthiness of the Feature handled by these operations is given. Unfortunately, this approach still has some limitations: it only allows to assign one specific property and its rating per operation, resulting in only four properties characterising the whole system. This might be enough for very basic instances but is very limiting. To overcome this limitations, a further extension encapsulating the SPs need to be introduced. This extension is called Security Property Record (SPR) and simply represents a set of multiple security properties. In detail, there may be one security property record per operation, thus resulting in four SPRs. Each SPR itself is unlimited in terms of the amount of security properties attached to it. It is simply defined as $SPR_{operation} = \{SP_1, SP_2, ..., SP_n\}$. The SPR does not encapsulate the actual ratings for the security properties as this link is established by accessing the security property map. This allows to change ratings although the property the ratings is defined for is already part of a SPR. Enhancing this, the SPR may be re-evaluated at any time and incorporates the rating which is currently defined for the properties that are part of the record. The assignment and lookup of the ratings is performed by another (i.e the second besides the SPM) conceptual component.

Security Property Record Manager

The Security Property Record Manager ($SPRM$) is the conceptual component responsible for assigning (1) the appropriate property to an operation and (2) the correct rating to that property at any time it is requested. Furthermore, (3) the $SPRM$ takes care of changing ratings for SP's dynamically.

The assignment of the properties (1) is performed in several sub-steps: first of all, when an operation is carried out, the $SPRM$ is able to identify the exact type of operation. That is, it can distinguish between the four operation types defined above. With having this information, it accesses the SPM in the next step and performs a lookup which property is assigned to that operation. It then takes the property (or more properties if there are more defined for this operation) and puts them into the operations SPR, finally attaching this SPR to the operation. The algorithm used for this task is shown in the following.

Algorithm 1: SPR construction.

```
Data: $SPM$, operation
Result: $SPR_{operation}$
type = getOperationType(operation);
accessSPM();
foreach $SP$ : mapSP in $SPM$ do
    spType = getSPType(mapSP);
    if type matches spType then
        addPropertyToSPR(mapSP);
    end
end
```

The second responsibility of the $SPRM$ is the mapping of the appropriate ratings for an operation's SPR. As with the forward mapping of the properties to an operation's SPR this is also done in several steps. As a precondition, there must be some kind of request which needs to access the actual ratings of the properties stored within a security property record. This could be for example an event, where it is necessary to calculate an actual trustworthiness of the operation. After the request has been made, the first actual step done by the $SPRM$ is to retrieve the SPR of the operation where the ratings are requested for. Having this SPR, the $SPRM$ can now perform and inverse lookup on the security property map. In particular, for each property stored in the operation's SPR, the $SPRM$ searches the appropriate SP entry in the static part of the SPM. This entry is then used to access the actual rating in the dynamic part of the SPM. The SPR's data structure is then expanded with the retrieved ratings, thus adding the rating for each SP in the SPR. To allow this addition of ratings, the so-called Rated Security Property Record ($RSPR$) is used. The format of the $RSPR$ is defined as $RSPR_{operation} = \{(SP_1, \omega_{SP_1}), (SP_2, \omega_{SP_2}), ..., (SP_n, \omega_{SP_n})\}$. It could equally be expressed as $RSPR_{operation} = \{(SP, \omega)_1, (SP, \omega)_2, ..., (SP, \omega)_n\}$. In addition to this, the process of looking up all ratings of a SPR's properties and the construction of the appropriate $RSPR$ must be performed as one single, i.e. atomic, step. This is necessary to prevent the construction of an invalid $RSPR$ in case of a rating change: if ratings are changed while the $RSPR$ is constructed, it would hold new (valid) as well as old (invalid) ratings in a mixed form. The summary of the algorithm is shown below.

Algorithm 2: $RSPR$ construction : getRSPR($SPR_{operation}$).

```
Data: $SPR_{operation}$
Result: $RSPR_{operation}$
receivedRSPRRequest();
foreach $SP : opSP\ in\ SPR_{operation}$ do
    lockSPM();
    accessSPM();
    foreach $SP : mapSP\ in\ SPM$ do
        if $opSP\ matches\ mapSP$ then
            rating = retrieveRatingByIndex(mapSP);
            addRatingForToRSPR(rating, opSP);
        end
    end
    unlockSPM();
end
```

It is easy to see, that the algorithm waits until an actual request is received. Furthermore, it also locks the SPM access exclusively. This is done in order to prevent the problems with updating certain ratings while simultaneously constructing the $RSPR$. The process of changing the ratings in a dynamic fashion (3) is also performed by the $SPRM$. To perform the actual rating update based on a certain property, the $SPRM$ must receive the new rating. It therefore takes the tuple $(SP_{operation}, \omega)$ as input. Besides the update process, this tuple can also be used to initially set the ratings and/or the security properties itself. As this process equals the update procedure, it is not explained additionally but as part of the update algorithm. After the $SPRM$ received this tuple, it accesses the SPM as soon as possible. That is, if a $RSPR$ construction is performed at the same time, the SPM is locked until the $RSPR$ has been constructed. In this case, the $SPRM$ first finishes the construction before starting the update procedure. If it is able to access the SPM it looks up the property given in the tuple. If existing, the appropriate rating is changed by navigating to the correct entry via the index. If there is no property entry, a new one is created in the static part of the SPM. Moreover, the index-based link between the rating is created and the rating itself is set. The update procedure itself must run exclusively in order to prevent problems with inconsistent ratings or properties. Due to this, the algorithm also locks the SPM and the `lockSPM()` as well as `unlockSPM()` functions only return if there is currently no lock on the SPM. The complete update algorithm is summarised below.

Algorithm 3: Rating and Property update.

```
Data: $SPM_{old}$, $(SP_{operation}, \omega)$
Result: $SPM_{new}$
lockSPM();
accessSPM();
spFound = false;
foreach $SP$ : $mapSP$ in $SPM_{old}$ do
    if $SP_{operation}$ matches $mapSP$ then
        updateRatingByIndex(mapSP, $\omega$);
        spFound = true;
    end
end
if $spFound == false$ then
    int idx = addSecurityProperty($SP_{operation}$);
    addRatingByIndex(idx, $\omega$);
end
unlockSPM();
```

While the tuple (property per operation and assigned rating) the $SPRM$ uses for this update procedure is defined, the process of its construction is not. The construction must be performed by external components which possess information about how to map operations and their ratings appropriately. To give an example, this could be done by performing a remote attestation (section 3.2.3) on the component the rated operation will be assigned to. In detail, the presented attestation approaches like a permission-based attestation [149] for an Android-based smartphone, can now be leveraged for this task. It could also be achieved by using expertise about the components or the overall system.

Deployment of Conceptual Components

Besides the procedures the $SPRM$ performs on the SPM, the deployment of both components needs to be defined. This is necessary as the underlying component (i.e. where the $SPRM$ and the SPM will be deployed at) need to fulfil certain conditions in terms of its trustworthiness. In addition, the conceptual deployment itself hast to be based on the abstract roles defined above. There is no way of defining the deployment based on actual components as the approach does not specify these components. That is, the $SPRM$ and the SPM need to be assigned to one of the roles and have to be mapped to the actual component when instantiating the approach, as well as the role itself . To select a role, it must be first defined if the $SPRM$ and the SPM can be deployed as one logical component. As there is no need for having both components in a standalone manner, they can be deployed as one single combined ($SPRM/SPM$) component for simplification reasons. Furthermore there is only one role, the Provider, which is applicable to hold the combined components. Besides other which will be explained later, this is due to the fact, that the Provider is the central role within the approach's architecture, thus having connections to both other roles. To actually deploy the combined $SPRM/SPM$ the Provider has to be trustworthy under any circumstances. That is, if it becomes compromised, the stored SPM as well as the processing $SPRM$ might be compromised. As they both form the basis for deriving trust, the overall trust calculation would be compromised. Due to this, when instantiating the model and implementing it, there need to be special measures to ensure the integrity of the Provider itself.

Property Summary

The security property approach introduces three main conceptual parts: the properties (SP) and their set (SPR) itself, the SPM and the $SPRM$. Security properties are defined as arbitrary characteristics which express the trustworthiness of a certain operation. To measure this trustworthiness they are determined by ratings expressing the actual trustworthiness of the property itself. security property records encapsulate an arbitrary amount of SPs for one operation. If an actual rating is attached, a Rated SPR is generated. The properties and their ratings, which may change in a dynamic fashion, are stored within the security property map. The map itself is used by the security property record manager which takes care of changing the map and creating the SPR as well as the $RSPR$. Both the SPM and the $SPRM$ are located on the Provider. Based on these components, the actual approach for calculating the trust can be introduced.

4.2.3 Trust Calculation

This section introduces the calculation of the actual trust. It therefore introduces the so-called Trust Level (TL) expressing this calculated trustworthiness. Furthermore, there are some intermediary steps which have to be carried out in order to calculate the overall TL. These steps are also explained within this section.

The previous section introduced the abstract role model as $process_{Sender} \xrightarrow{\zeta} transmit_{Sender} \xrightarrow{\zeta} process_{Provider} \xrightarrow{\zeta} transmit_{Provider}$. Additionally, the first part of the trust model added security properties encapsulated in a security property record and weighted in terms of trust by a rating. Therefore it has been defined, that each of the four operations has a SPR attached to it. This SPR is being used to (1) retrieve the current ratings for the SPs stored within it (i.e. generating a $RSPR$) and (2) to calculate the trustworthiness of this operation. Based on this, the overall trust level which combines the trustworthiness of all four operations into one measurement can be calculated. This TL does not express an operation's trustworthiness but the trustworthiness of the Feature that was handled by these operations. The first step, the creation of the $RSPR$ is performed by using the $SPRM$ in the manner shown above. As with the $SPRM$ itself, the trust calculation is performed as part of the Provider. That is, each time the Provider is in charge of handling a Feature, the TL for this Feature is being calculated. The calculation of an operation's trustworthiness (2) is done by calculating the so-called Security Level (SL) for this operation.

Security Levels

The SL combines the ratings of an operation's SPR into a single value. That is, to evaluate the SPR (i.e. the $RSPR$) of an operation, a generic rating Function (rF) which composes an arbitrary amount of ratings together into the SL is used. The function is defined in the following way.

$$rF(\{\omega\}) = \omega_1 \circ \omega_2 \circ ... \circ \omega_n$$

In detail, to calculate the SL of an operation, the Provider takes the SPR for this operation and retrieves the assigned rating (using the $SPRM$) for each SP of the SPR. These ratings are then composed by using the defined rF to calculate the SL. While the codomain of the rating itself is not defined, the codomain of the SL is defined as value of $[-1, 1]$ allowing a scoring (cf. [35]) of the SL values. Using the calculations parts which are currently defined together, the following expression can be formulated.

$$SL_{operation} = rF\Bigg(\omega_{SP_1}, \omega_{SP_2}, ..., \omega_{SP_n}\Bigg) \text{ with } SP \in SPR_{operation}$$

That is, the SL of a certain operation is calculated by combining each rating of the operation's security properties using the generic rating Function. Due to the fact that there are four operations defined by the model, four different SLs can be calculated. The algorithm used by the Provider is depicted in the following. It uses the $RSPR$ generated by the $SPRM$ as input and calculates the appropriate SL.

Algorithm 4: Simple calculation of the SL for an operation : calculateSL($RSPR_{operation}$).

```
Data: $RSPR_{operation}$
Result: $SL_{operation}$
for $i < RSPR_{operation}.size$ do
    $SL_{operation}$ = ratingFunction($SL_{operation}$, $RSPR_{operation}$.getRatingByIndex(i));
    i++;
end
```

As a single SL only expresses a trust measurement of a particular operation, there needs to be another mechanism to combine the SLs of all four operations into a single value. Besides that, the SL already allows a reasoning about trustworthiness of one operation which may be useful under certain circumstances. The construct to aggregate the single SLs into one value is introduced in the next section.

Trust Levels

As already shown, the trust model uses so-called trust levels (TL) to express a Feature's trustworthiness. That is, the TL combines the previously calculated SLs into a single value. In order to explain the calculation, the trust definition made in section 4.2.1 needs to be applied to the concept of trust levels. It has been defined that the model allows to reason about a Feature's trustworthiness by evaluating the trustworthiness of the operations used to handle this Feature. That is, operations are rated in terms of their expected result. This is done by defining properties and appropriate ratings for the operations. Combining these ratings, an overall measurement for the operations trustworthiness can be calculated (i.e. the SL). As there are four operations carried out when handling a Feature, the combination of these four SLs build the overall TL. This TL is the overall trust measurement of the Feature. It is based on the ratings of all operations properties. That is, the trust level expresses the trustworthiness of a Feature based on the trustworthiness of the operations that have been used to handle this Features. The term trustworthiness is defined as the behaviour of an operation in terms of Feature handling. Given this trust definition of section 4.2.1, a more refined version which is based on this trust level can be formulated: *A Feature is considered trustworthy if and only if its trust level is considered trustworthy.* That is, if the calculated trust level for the Feature lies within a trustworthy range (this range may vary depending on the particular values used), the Feature itself is considered trustworthy.

The trust level is defined as composition of all SPs which are assigned to the four operations and allows a reasoning based on the assigned properties (i.e. their ratings). Another function, the so-called Trust Function (tF), which expresses this composition is therefore simply defined as combination of all four SLs.

$$tF(\{SL\}) = SL_{process_{Sender}} \circ SL_{transmit_{Sender}} \circ SL_{process_{Provider}} \circ SL_{transmit_{Provider}}$$

The particular method used for the combination itself depends on the actual domain the model is instantiated for, thus needs to be specified for instantiation. Besides that, due to the codomain definition of rF ($[-1,1]$) the domain of tF is also defined as value of $[-1,1]$. By using tF, the overall trust level for a Feature can be calculated in the following way:

$$TL_{\zeta} = tF\Big(\underbrace{rF(\{\omega\}_{process_{Sender}})}_{SL_{process_{Sender}}}, \underbrace{rF(\{\omega\}_{transmit_{Sender}})}_{SL_{transmit_{Sender}}}, \underbrace{rF(\{\omega\}_{process_{Provider}})}_{SL_{process_{Provider}}}, \underbrace{rF(\{\omega\}_{transmitProvider})}_{SL_{transmit_{Provider}}}\Big)$$

Summarising this, the overall process of calculating the trust level for a certain Feature consist of three phases: (1) the assignment of security properties (SPs) and their appropriate ratings (ω), (2) the calculation of four SLs with one for each operation by using the rF and (3) finally the calculation of the TL by using the tF and the previous calculated SLs. This process can be expressed in the following simple manner.

$$\{SPR, \{\omega\} \xrightarrow{rF} SL\}_{operation} \xrightarrow{tF} TL_\zeta$$

As it easy to see, the first steps are already defined as algorithms shown above. The missing final algorithm to put all parts together and forming the overall TL is shown below. It simply takes the four SLs and combines them using the tF into one single TL.

Algorithm 5: Calculation of the final TL for a Feature.

```
**Data**: $SL_{process_{Sender}}, SL_{transmit_{Sender}}, SL_{process_{Provider}}, SL_{transmit_{Provider}}$
**Result**: $TL_\zeta$
$TL_\zeta =$
trustFunction($SL_{process_{Sender}}, SL_{transmit_{Sender}}, SL_{process_{Provider}}, SL_{transmit_{Provider}}$);
```

Due to the SLs encapsulating most of the previously made calculations, the combination of them into one single value is rather straightforward. Using the result of this algorithm, the final TL, a reasoning about a Feature's trustworthiness can be done. Due to the Provider being responsible for performing this algorithm and thus calculating the TL_ζ, there is one problem: it needs to know the last $SPR_{transmit_{FP}}$. As this step represents the feature request of a Receiver, the actual properties of the transmit operation are only known after the Provider received the request and can determine the SPR by using the $SPRM$. This may not be a problem if only one single Receiver is used as the operation could be predefined but if there are more Receivers, the last SPR is unknown until the request was received by the Provider. To address this problem, phases are introduced within the next section.

4.2.4 Phase Extension

To solve the problem stated above, the TL calculation is being divided into two phases and thus there are two actual types of trust level. The first type, being calculated when a Feature was processed on the Provider is called Phase 1 Trust Level (TL_ζ^{P1}). Equally to that, the second type is called Phase 2 Trust Level (TL_ζ^{P2}). The previously defined trust level without a specific phase is still used in an abstract manner where phases are not relevant. The phases which define either the phase 1 or phase 2 trust level are defined and explained in the following.

Phase 1

Phase 1 describes the combination of the first three SLs into one single TL_{ζ}^{P1}. It therefore consists of the operations $process_{Sender} \xrightarrow{\zeta} transmit_{Sender} \xrightarrow{\zeta} process_{Provider}$. The phase 1 trust level is due to this simply defined as part of the overall TL, shown in the following.

$$TL_{\zeta}^{P1} = tF\left(\underbrace{rF(\{\omega\}_{process_{Sender}})}_{SL_{process_{Sender}}}, \underbrace{rF(\{\omega\}_{transmit_{Sender}})}_{SL_{transmit_{Sender}}}, \underbrace{rF(\{\omega\}_{process_{Provider}})}_{SL_{process_{Provider}}}\right)$$

This means, after a Feature was received and processed by the Provider, the TL_{ζ}^{P1} is calculated. It expresses only the trustworthiness of the Feature based on the first three operations. The algorithm used to calculate it is a limited version of the complete algorithm which was shown above.

Algorithm 6: Calculation of the TL_{ζ}^{P1} for a Feature : calculateP1TL($SL_{process_{Sender}}, SL_{transmit_{Sender}}, SL_{process_{Provider}}$).

```
**Data**: $SL_{process_{Sender}}, SL_{transmit_{Sender}}, SL_{process_{Provider}}$
**Result**: $TL_{\zeta}$
$TL_{\zeta}^{P1}$ = trustFunction($SL_{process_{Sender}}, SL_{transmit_{Sender}}, SL_{process_{Provider}}$);
```

It is easy to see, that the tF takes only the first three SLs as input, thus only using the first three SPRs and their properties to calculate the TL.

As already stated, the phase 1 trust level can be calculated each time a Feature was received and processed by the Provider. The TL is then stored in special structure, which will be explained later. If the Provider receives a request for the Feature from a Receiver, it can initiate the procedure to calculate the second phase.

Phase 2

This phase consists of the calculations made in the first phase as well as the trustworthiness of the last operation (transmit from the Provider to the Receiver). This is due to the fact, that the $SPRM$ can only determine the final SPs for the $transmit_{Provider}$ operation if it knows the actual Receiver the Feature should be sent to. For example, if there are security properties expressing a communication channels integrity (e.g. if it is encrypted), the exact property can only be assigned if the communication channel is known at assignment time.

As with phase 1, the second phase describes the combination of all four SLs into one single trust level. This trust level is then called phase 2 trust level, short TL_{ζ}^{P2}. It consist of the complete operational flow $process_{Sender} \xrightarrow{\zeta} transmit_{Sender} \xrightarrow{\zeta} process_{Provider} \xrightarrow{\zeta} transmit_{Provider}$, thus in difference to the TL_{ζ}^{P1} it also encapsulates the last operation. The algorithm used has already been shown as Algorithm 5, this is due to using all four SPRs again. The only difference is that the algorithm is no longer referred to as `calculateTL(...)` but as `calculateP2TL(...)` where the arguments are all four SLs.

While the phase 1 trust level is globally valid, the TL_{ζ}^{P2} is only applicable for the requesting Receiver. That is due to the TL being calculated exactly for this Receiver as it takes SPs from the last transmit operation to this Receiver.

Phase-based Flow

The overall flow when handling Features and calculating trustworthiness consist of the two phases added above. That is the following steps are carried out.

1. Processing of the Feature on the Sender. As already explained this consists of creating and/or aggregating the Feature and preparing it.

2. Transmission of the Feature to the Provider using a protocol capable of transferring the Feature appropriately.

3. Processing of the Feature on the Provider, thus storing and retrieving the Feature.

4. Calculation of the phase 1 trust level by the Provider. It uses the $SPRM$ and SPM: the SPs of the first three operations are taken to calculate the TL. The TL is then stored and kept ready for further use. Every time the Feature is being re-received by the Provider, the TL will be recalculated using the last values set in the SPM.

5. If a Receiver wants to access the Feature, the Provider calculates the TL_{ζ}^{P2} for the last transmit operation as it knows the Receiver at this time. It attaches the calculated TL to the Feature.

6. The Feature and the TL_{ζ}^{P2} are transmitted to the Receiver which was requesting the Feature.

In addition to the Provider being trustworthy as it calculates the TLs, the transmission of the TL_{ζ}^{P2} needs to be secured as well. If it is done via an unsecured channel, the calculated TL might be compromised. Although the concept does not enforce it, it is strongly recommended to sign the TL_{ζ}^{P2} in order to make sure that the Receiver can determine the source of the TL_{ζ}^{P2}. In contrast to the evaluation of the last transmit operation, signing the TL_{ζ}^{P2} is enough as the Provider is considered trustworthy. Furthermore, it is also possible to use another communication channel than the last transmit to propagate the TL_{ζ}^{P2}.

4.2.5 Feature Handling on the Provider

As already outlined, the Provider takes care of calculating the trust level of a Feature on certain occasions. This section defines which occasions it is responsible for. Mainly, there are three types of handling processes the Provider can perform. The first one is the creation of a Feature, the second one encapsulates the process of updating an already created Feature and the last one handles the deletion of a Feature. Figure 4.4 depicts the life cycle formed by these three types.

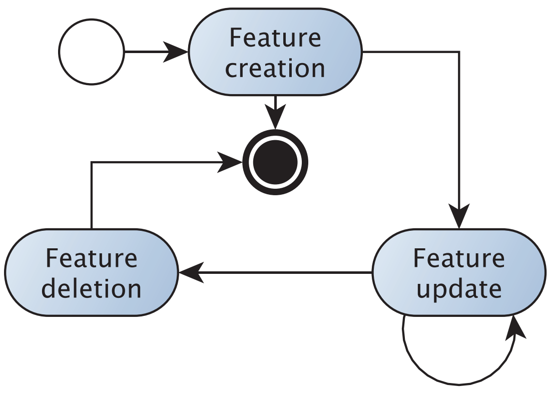


Figure 4.4: Feature life cycle on the Provider.

Feature Creation

If the Provider receives a Feature from the Sender it has never received before, it creates a new entry for this Feature and stores it locally. That is, the term creation refers not to the process which is done on the Sender but describes the type of process operation performed by the Provider. In addition, the Provider calculates the TL, first the TL_{ζ}^{P1} and if it is requested the TL_{ζ}^{P2}. The Feature creation can only be performed as long as the Feature has not already been created.

Feature Update

If the Provider receives a Feature from the Sender which has already been received it updates this Feature. It is not important if the Feature is received the second time (thus the previous handling operation was Feature creation) or n time (previous operation was update). Besides the update of the Feature, the Feature's TL is also updated. That is, the Provider simply performs a recalculation of the TL thus including potentially updated ratings. This recalculation applies for both types of the TL considering that the TL_{ζ}^{P2} is only recalculated after the Feature was requested by a Receiver.

Feature Deletion

A special kind of handling is the deletion of a Feature. While the other two handling procedures receive a Feature from the Sender which should be stored, this type receives a Feature which is flagged for deletion. This indicates, that the Feature should be deleted what is then done by the Provider. In addition, the TL_{ζ}^{P1} for the Feature is also deleted.

The overall algorithm realising these three types of Feature handling by the Provider is shown in the following. The Sender is only capable of telling the Provider to receive or delete a Feature, it is not able to express a Feature update. This is due to the fact, that another Sender may have already sent the same Feature.

Algorithm 7: Feature handling on the Provider.

```
Data: Sender request type $type$ ,may be either add or delete
Result: Feature processed $\zeta_{processed}$
$\zeta_{received}$ = receiveFeature();
if $type == add$ then
    update = false;
    foreach $\zeta$ : storedFeature in ProviderStorage do
        if storedFeature == $\zeta_{received}$ then
            updateFeature(storedFeature, $\zeta_{received}$);
            update = true;
        end
    end
    if update == false then
        addFeature($\zeta_{received}$);
    end
else
    deleteFeature($\zeta_{received}$);
end
```

While the first two types are straightforward, the deletion introduces a problem: there is no way to indicate the trust level for the Feature deletion. That is, the current model is unable to express the trustworthiness of the deleted Feature and of the operations used to delete the Feature. This is problematic as there is now way to recognise if a compromised component has deleted a Feature indicating this component's compromise. Due to this, another conceptual part is needed which is introduced in the following section.

4.2.6 Snapshots

While the current conceptual parts allow to store the TL that is currently valid for a Feature that is available on the Provider, it does not store TLs for deleted Features as well as old TLs for updated Features. This leads to the problem shown above and in addition to the problem of recognising changes of the TL when updating a Feature. To circumvent this, the trust model uses so-called snapshots. These snapshot preserve the actual state of a Feature including the Feature itself as well as the Features trust information. The Trust information includes the aggregated TL as well as the $RSPR$ which consists of the SPs and their appropriate ratings. That is, the snapshot stores the ratings which where assigned to the properties at the time the TL was calculated. By using these snapshots, it is possible to evaluate the change of the TL or its parts (SP, ratings) and to compare older values against the actual value.

There are three types of snapshots which correspond to the Feature's handling phases on the Provider. Due to this, the types are a so-called Create Shot, an Update Shot and a Delete Shot. These snapshots are always generated when the appropriate handling phase is carried out. The last snapshot which has been created always represents the current state. That is, if a Receiver accesses a Feature and its Trust information, the snapshot that has been created as last (thus being the newest snapshot) is given to render the Feature's actual state in terms of trust. This implies, that every time a Feature handling on the Provider takes place, a snapshots is being created.

Snapshots are only generated for TL_{ζ}^{P1}. That is, the trust information stored in the snapshot does not include the $RSPR$ for the final transmit operation. Due to this, the TL_{ζ}^{P2} is only valid for the most recent snapshot and cannot be applied to older snapshots. This is necessary, as there is no way to determine the last transmit operation as this determination relies on the properties of the actual Receiver. These properties are unknown before a Receiver's actual access request is being received.

Create Shot

The Create shot is generated when a new Feature is received on the Provider. The term new refers to the Providers view on the Feature and means that the Provider has never received this Feature before. This snapshot type therefore stores the trust information which is valid if a Feature is newly created on the Provider. Analysing the overall Feature life cycle on the Provider, it is clear that there can be only one Create Shot for a Feature. This is due to the Feature only being created once. If the Feature never gets updated nor deleted, this type of snapshot remains the most current all the time.

Update Shot

The Update snapshot is created each time a Feature is updated, which occurs if the Provider receives a Feature again. Due to this, there may be an unlimited amount of Update Shots, which is contrary to both other types. As with the other type, each Update Shot stores the Feature's value and the associated trust information that were valid at the time of updating. If the Feature is updated but not deleted, the lastly created Updated Shot expresses the Feature's current trust state.

Delete Shot

If a Feature is intended to be deleted, a Delete Shot is being created. It consists of the Feature's last value prior to its deletion and the trust information of the update which instructed the deletion. Due to this, the Delete Shot holds information about the SPs of the operations that where performed to delete the Feature making it possible to evaluate this deletion. As a Feature can only be deleted once, there is only one Delete Shot. This Delete Shot forms the final state of the Feature as well of the Feature's trust information. If the Feature is updated again, a new Create Shot is generated. This behaviour corresponds to the Provider's Feature handling model. This type of snapshot can be used to determine the state of the Feature prior to its deletion and, more importantly, to recognise which Sender (including this Sender's SPR) triggered the deletion of the Feature.

The following algorithm summarises the snapshot extended handling of Features on the Provider.

Algorithm 8: Snapshot extended Feature handling on the Provider : handleSnapshot(*type*, ζ).

```
Data: Sender request type type ,may be either add or delete
Feature received $\zeta_{received}$
Result: Feature processed $\zeta_{processed}$ and the Snapshot
if $type == add$ then
    update = false;
    foreach $\zeta$ : storedFeature in ProviderStorage do
        if $storedFeature == \zeta_{received}$ then
            Snapshot = generateUpdateShot($\zeta_{received}$);
            updateFeature(storedFeature, $\zeta_{received}$);
            update = true;
        end
    end
    if $update == false$ then
        Snapshot = generateCreateShot($\zeta_{received}$);
        addFeature($\zeta_{received}$);
    end
else
    Snapshot = generateDeleteShot($\zeta_{received}$);
    deleteFeature($\zeta_{received}$);
end
```

Summary

The actual trust calculation used in the approach consists of three main parts: the definition of properties and their ratings, the calculation of the trust level and the extension of this Level into phases and snapshots. The properties, ratings and appropriate components where already defined in the last section. This section introduced the trust level which is being calculated in two sub steps: first the calculation of a SL based on the operation's $RPRS$. Second the combination of these SLs into one single trust level. As the SL of the last transmit operation is only known at certain times, the trust level is divided into two types. While the TL_{ζ}^{P1} expresses the Feature's trustworthiness only based on the first three operations, the TL_{ζ}^{P2} includes also the fourth operation. Due to this, the TL_{ζ}^{P1} is globally valid for a Feature while the TL_{ζ}^{P2} is only valid for a particular Receiver. To allow reasoning about deletion of a Feature and the change of trust, snapshots are used. There are three types of snapshots, each representing one phase of Feature handling on the Provider.

By using this parts of the trust model, the approach allows to reason about a Feature's trustworthiness. Furthermore it allows to build a history in terms of trust change of a Feature and to distinguish between the trustworthiness of a Feature stored on the Provider or already placed on the Receiver.

4.3 Data Model

To actually use the parts of the model and their appropriate trust building components, a data model encapsulating Features and their trust information is needed. Figure 4.5 depicts the overall architecture which is based on the so-called TrustLog. Besides the TrustLog and the Feature stored in it, it consists of several sub elements mapping the components of the model. These elements are assigned to three layers which are explained in the following in a bottom up manner. That is, Layer 1 forms the basic building blocks while Layer 3 forms the top.

4.3.1 Security Property Layer

The lowest layer consists of the elements mapping the security properties and their appropriate records. It is depicted in figure 4.6.

SecurityProperty

The SecurityProperty element encapsulates one single security property and its rating. It is not bound to a specific operation, thus the same element may be used for several operations. The rating itself is actually stored to allow an evaluation of new rating values (if the rating was changed meanwhile) versus the stored value. The SecurityProperty element is the most basic part of the data model and does not use other subclasses. When

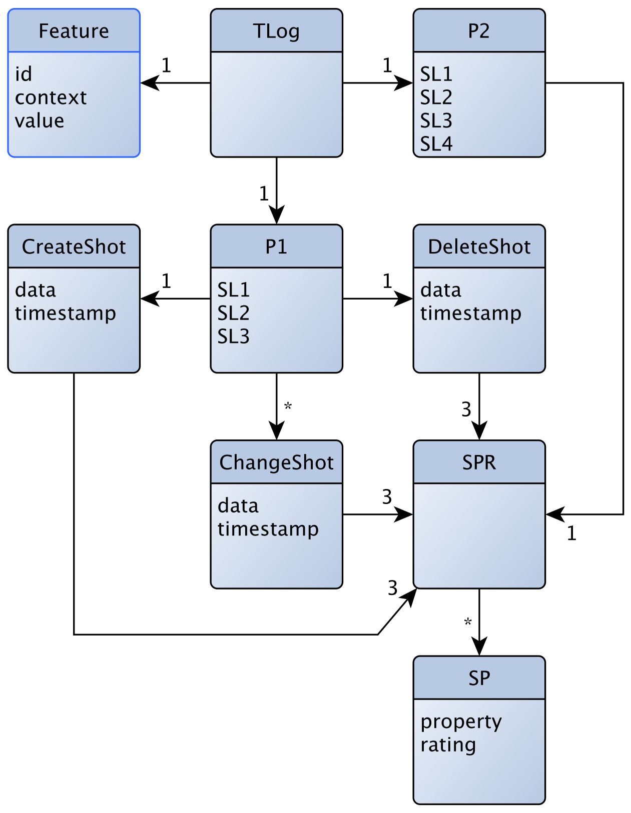


Figure 4.5: Complete TrustLog data model, including all layers. The Feature (marked by a blue frame), is given by the CADS model (see [2] for more details of the Feature data model).

the overall TrustLog is created, the Provider generates a SecurityProperty element for each security property that is part of the Feature's trust calculation. If more than one operation uses the same SP, there may be only one SecurityProperty object for this SP which is being used several times. This is possible as long as the operation references exactly the same SP.

SecurityPropertyRecord

The SecurityPropertyElement allows to map an operation's actual SPR into the data model. That is, for each operation which was part of the Feature handling the TrustLog is being created for, a SecurityPropertyElement will be produced. The element consists of an amount of n SecurityProperty elements which represent the SPs that are stored within a

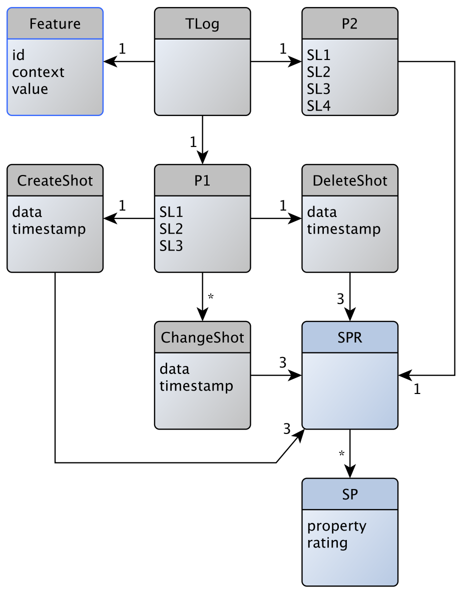


Figure 4.6: Security Property Layer of the TrustLog data model, whereas the coloured elements depict the appropriate classes of the layer. The relationship between the classes are given be the arrows which include the appropriate cardinality.

SPR. Because of the data model's SecurityProperty element actually storing the ratings besides the SPs, the SecurityPropertyElement maps the $RSPR$ rather than the SPR only. This is due to the SPR being unable to hold ratings. The SecurityPropertyRecord is being used by the layer above.

4.3.2 Phase and Snapshot Layer

The second layer maps the conceptual parts of the phases and the snapshots as part of the data model. It therefore consists of elements which represent the snapshot types and elements which represent the two different phases. The layer is depicted in figure 4.7.

Snapshot Elements

There are three snapshot elements mapping the Create Shot, Update Shot and Delete Shot. They are named equally to the conceptual components: CreateShot, UpdateShot and DeleteShot and are created according to the algorithm shown above. That is, every time a new Feature is stored and thus a new TrustLog is created, the first snapshot is

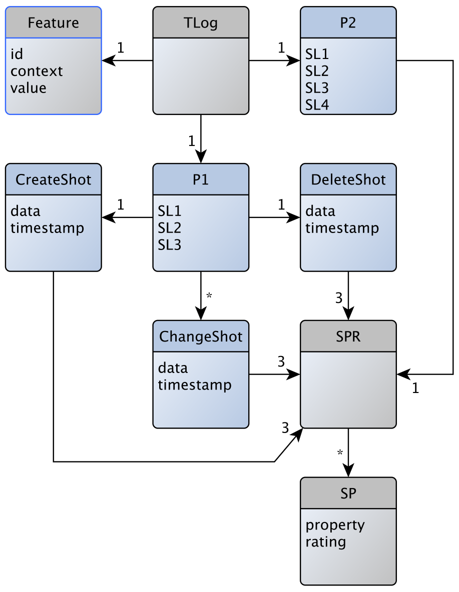


Figure 4.7: Phase and Snapshot layer of the data model with the appropriate elements shown as coloured classes. Cardinality and class relationship is indicated by the appropriate arrows.

created as CreateShot. Every time an update for an already stored Feature is handled by the Provider, a new UpdateShot is generated and associated with the appropriate TrustLog. Finally, if a Feature is going to be deleted, the appropriate DeleteShot is created and stored.

The Snapshot elements themselves consist of the Feature's stored value, i.e. not the whole Feature element but only the value of it. Furthermore a timestamp is stored allowing to track when the snapshot was created. Besides that, three SecurityPropertyRecord elements are attached to the Snapshot. That is, there are only three as snapshots are only created for phase one. The SecurityPropertyRecord elements may be different but can also be the same if different Snapshots reference exactly the same SecurityPropertyRecord element.

Phase Elements

There are two elements which map the appropriate phases onto the data model, they are simply called Phase1 and Phase2. Both types consist of the calculated SL with the difference, that Phase1 holds only three SLs while Phase2 holds four SLs. This is equally to the amount of operations which are considered by the appropriate phase. Besides the calculated SLs, the phase elements also hold the SecurityPropertyRecord for each SL. While Phase2 references four SecurityPropertyRecord elements directly, Phase1 references Snapshots. There are exactly one CreateShot and one DeleteShot as well as an arbitrary amount of UpdateShots referenced. The creation of these Snapshots is done according to the method mentioned in 4.3.2. That is, the SecurityPropertyRecord elements are not directly referenced by the Phase1 element but indirectly through the Snapshot elements. Phase2 can reference the SecurityPropertyRecord element directly, as there are no snapshots in the phase 2 flow of the model. This is due to the calculation of the TL_{ς}^{P2} only being valid for on single Receiver as well as being carried out just before the final transmit operation is performed. The layer 2 elements can now be used in order to create to overall TrustLog element.

4.3.3 Feature Layer

The upper layer, depicted in figure 4.8, consists only of two elements: the Feature itself and the TrustLog element which references the Feature. The root used within this data model is also rendered by the TrustLog, i.e. the Provider stores TrustLog entries for each Feature it is handling or it has handled.

TustLog

The TrustLog itself consist of the already mentioned reference to the actual Feature and to the appropriate phase elements. Furthermore, it holds the calculated trust levels, in detail the TL_{ς}^{P1} as well as the TL_{ς}^{P2}. The trust levels represent the particular valid TL, i.e. the TL_{ς}^{P1} that is based on the last snapshot and the TL_{ς}^{P2} which was just calculated for a certain Receiver. They are refreshed each time a TL is recalculated which would be in case of another Receiver for the TL_{ς}^{P2} and in case of update or delete for the TL_{ς}^{P1}. Besides this direct storage of the TLs, it references the two phase elements already explained. It is therefore possible to access every single part of the overall trust calculation by accessing the TrustLog.

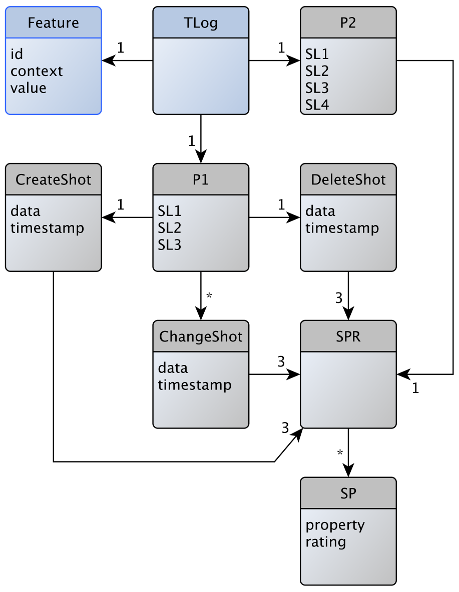


Figure 4.8: The Feature layer of the data model shown by coloured elements. Arrows indicated the relationship between the classes including their cardinality.

Feature

The Feature element itself simply holds the actual Feature and is referenced by the Trust-Log. It consists of the unique Feature id, the value of the Feature, the type of the Feature and the appropriate contextual information. It is described in detail in [2].

Summary

The data model introduced in this section allows to store the necessary parts of the trust model on the Provider. It consists of three different parts, each one depicting the appropriate area of the model. The main element of the data model is used as storage root for Features and their attached trust information on the Provider. That is, for each Feature handled a TrustLog data structure is created and used for further actions.

4.3.4 Provider State Machine

The previous sections introduced the trust measurement as well as their calculation along with a data model allowing to adopt the overall concept. As it was already stated in those sections, the Provider is responsible for performing all necessary steps, thus also for answering requests from other roles in the model. While the sections introduced the necessary parts to perform the trust calculation on the Provider, there are still two parts missing: (1) the types of requests the Provider is able to answer and to handle and (2) the detailed operations of those requests the Provider has to perform. These missing parts are introduced within this section and finalise the conceptual model.

Given the trust concept above, it is easy to see, that there are different type of states in it. That is, the trust calculation is always separated into several sub steps that need to be performed. Furthermore these steps need to be performed if a certain type of request is received by the Provider. Based on this, three different types of requests can be defined.

Sender Requests The so-called Sender Requests are the request which map the action if a Sender transmits a new or updated Feature to the Provider. As the name states, the request is bound to the Sender role and thus can only be triggered by a message from the Sender.

Receiver Request The other role-specific request type is the so-called Receiver Request. Inversely to the Sender Request, this type is triggered by the Receiver, namely in the case when a Receiver needs access to a Feature stored on the Provider. Furthermore, this type is also bound to the specific role thus can only be triggered by a Receiver.

Reasoning Request The last type, which is different to the first two types in terms of the accessing component, is the so-called Reasoning Request. This request is used to access one or more TrustLogs directly on the Provider. That is, it is not bound to a specific role but can be triggered by every one. Even more, it can be triggered by components which have not been directly mapped to roles. The request itself is needed for an overall trust reasoning of the collected Features. For example, by using this type it is possible to gain an overview about the trust situation of the complete environment (given that it is fully integrated).

To handle these types of requests in a defined manner, the Provider uses an internal state machine. This state machine realises each type of request and encapsulates the necessary sub steps to fulfil the request. Furthermore, it defines the transitions between the sub steps and determines also when a request type is finished. This renders the Provider ready to process another request. The overall state machine is depicted in figure 4.9 and will be explained by each request type in detail in the following sections. By using and adopting

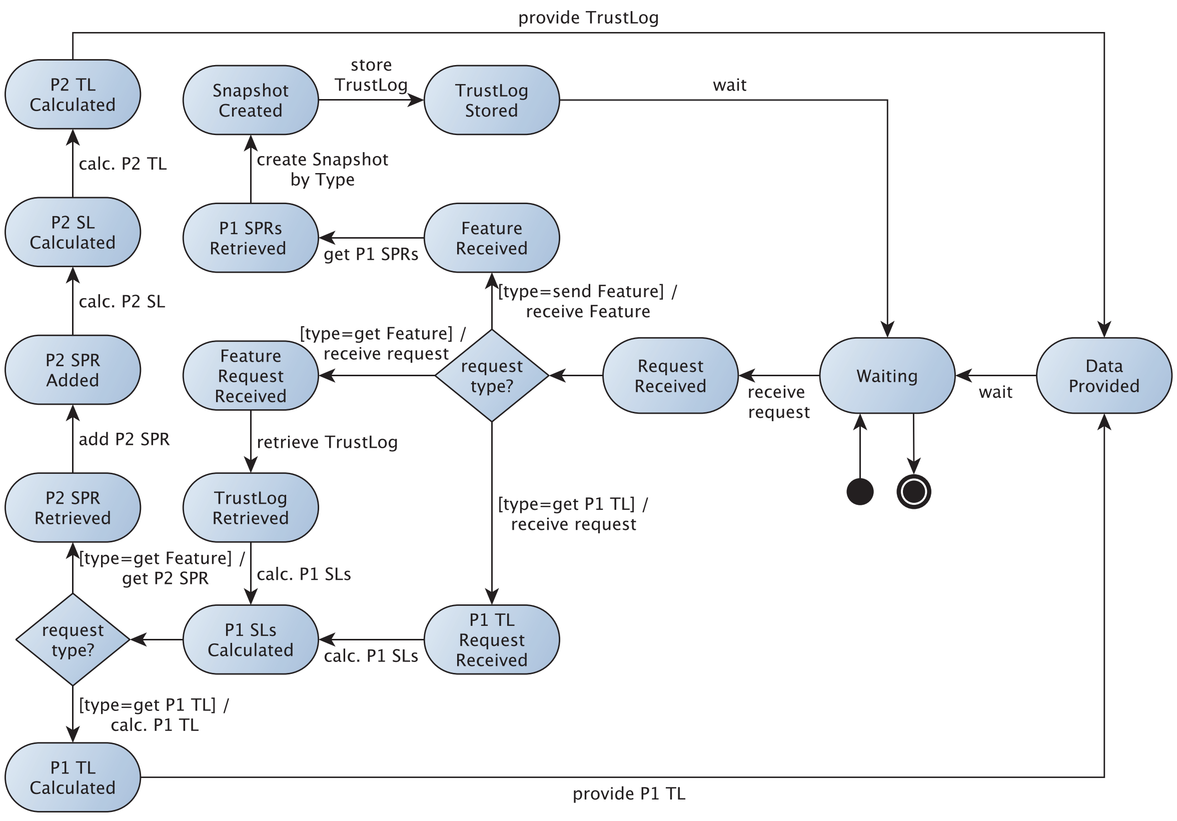


Figure 4.9: Provider State Machine including all relevant paths.

this state machine, the Provider is able to process the request types shown above completely. Furthermore it is able to perform the Feature handling as it was described in the abstract model part. To fully use the concepts defined here, it is necessary to implement this state machine. The general transitions through the states in the Provider can be divided into three parts: (1) the receipt of the appropriate request, (2) the evaluation of the request itself to gain the request type and (3) finally the actual handling of the request. Although the general flow is always the same, there are three distinct paths inside the state machine, one for each request type.

Sender Requests

As already explained, the Sender Request expresses those types of requests which update or create a Feature on the Provider. They are triggered by components mapped to the Sender role. As the type does not define if the request itself consist s of an update or a creation, the Provider needs to determine this. This is partially already handled by algorithm 8, which decides about the type of snapshot creation and thus needs to know if the Feature is already known. The path the request is handled within the state machine is shown in figure 4.10. As depicted, there are six different states the Provider can have if

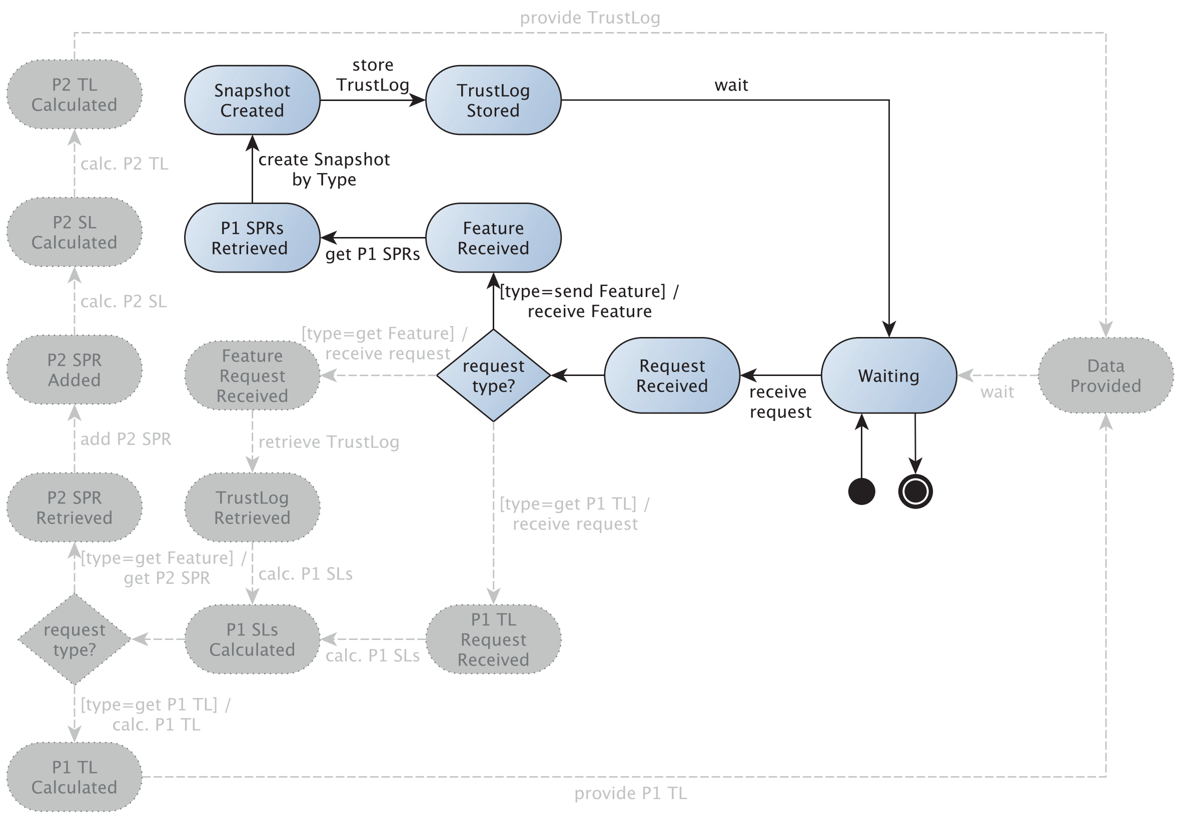


Figure 4.10: Sender Request path of the state machine, indicated by coloured elements.

handling this type of request. Two of these states are applicable for every request type: the state of waiting and the request received state. These states match the above describe part (1). Within the waiting state, the Provider does not perform any action but waits for an incoming request. If this request is inbound, the Provider receives it and changes its state into the request received state. At this time, all the data which belong to this request (but not the Feature) are on the Provider's side. Based on this information, the Provider decides about the request's type and changes into the appropriate path. In this case it steps into the Sender Request path shown in the figure. With stepping into this path, the Provider receives the actual Feature and changes it state to Feature Received. When analysing the defined operations in the model, the state change from Request Type to Feature Received occurs exactly when the $transmit_{Sender}$ operation is finished in this type of request. Due to the Provider possessing all necessary information to fulfil the request, it can now advance further. As this request type is only applicable for phase 1, the Provider's next step is to receive all necessary $RSPR$s. This is done by querying the $SPRM$ in a reasonable way which responds with three $RSPR$s, that are based on the operations that have been carried out to handle the Feature up to this point. When the Provider has successfully gained the appropriate $RSPR$s, it changes its state into SPR Received. In the next step, the Provider changes into the state Snapshot Created. It is reached when the Provider has successfully generated the appropriate snapshot for the Feature it has received. This may be either a CreateShot, an UpdateShot or a DeleteShot. Having reached this state and thus created the snapshot, the Provider continues the trust calculation by calculating the three security levels as well as the Phase1 element storing those SLs. Furthermore, it creates the overall TrustLog structure which combines all elements into one single entity. After it has created the TrustLog, it is stored appropriately inside the Provider's storage area. The Provider changes its state to Log Stored, thus finishing the request handling. As the Provider needs to be able to handle further incoming requests, the last step is to wait and thus changing back into the Waiting state where it was before it starting handling this request.

Although the Provider calculates the SLs and the Phase1 elements in this type of request, it is not necessary by all means. That is, the calculation of the SLs as well as the creation of the phase elements are only done, considering the data model, because the Provider needs to attach the snapshot anywhere. The abstract flow of this request type does not enforce the calculation of the SLs. This is due to the fact, that the final trust level is only calculated when one of the other request types is being handled. In order to provide always trust levels of highest possible freshness it is unavoidable to calculate the TL just in time when it is being requested. If a more complex data model is being used, which is able to support snapshots that are not directly bound to a phase element, it is possible to store the TrustLog and the snapshots only, without taking care of the SLs and the phase elements. In this case, the Provider would not request the $RSPR$s but the SPRs only to assign them properly.

The following algorithm summarises the handling process for this type of request without considering the drawbacks of the data model. That is, the SLs are not calculated and the algorithm assumes, that snapshots may be directly attached to the TrustLog. Furthermore, it makes use of algorithm 8 and 1 to create the actual snapshot and determining the correct type.

Algorithm 9: Handling of Sender Requests : handleSenderRequest(*type*).

```
Data: type type of the request (add or delete)
Result: Stored TrustLog for the Feature received in the request
$\zeta_{received}$ = receiveFeature();
Snapshot = handleSnapshot(type, $\zeta_{received}$);
update = false;
foreach $\zeta$ : storedFeature in ProviderStorage do
    if storedFeature == $\zeta_{received}$ then
        TrustLog = getTrustLogByFeatureId($\zeta_{received}$);
        updateTrustLog($\zeta_{received}$, Snapshot);
        update = true;
    end
end
if update == false then
    createTrustLogByFeature($\zeta_{received}$);
    updateTrustLog($\zeta_{received}$, Snapshot);
end
```

As already shown, after the Provider has finished handling the request, it enters the Waiting state again and is ready to process another request.

Receiver Requests

The second request type the Provider handles is the so-called Receiver Request. It is triggered by a Receiver which requires to access, thus receives, a Feature from the Provider. The path which is used within the state machine is shown in figure 4.11. The main parts of this request type are the receiving of the request itself, the retrieval of the stored TrustLog out of the Provider's storage and finally the calculation of both trust level types and the providing of the Feature and the TLs to the Receiver. Starting in the Waiting state again,

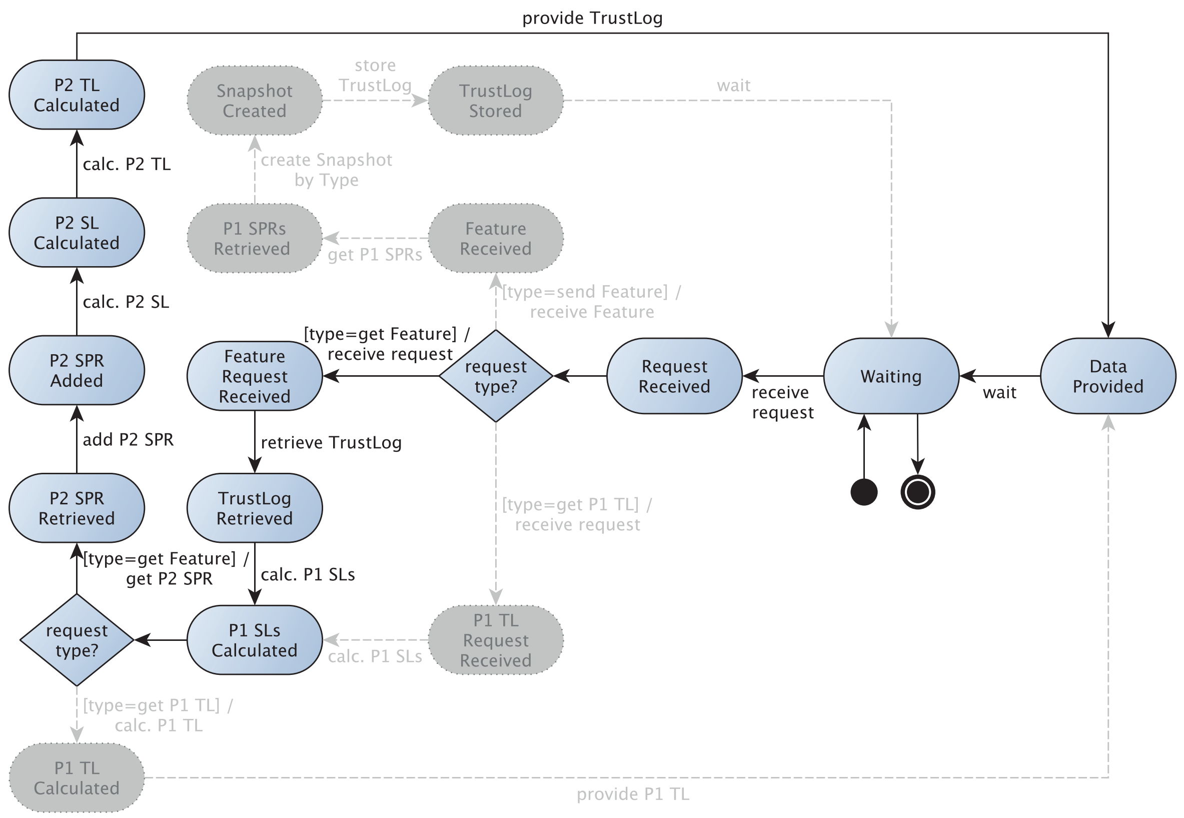


Figure 4.11: Receiver Request path of the state machine depicted as coloured elements.

the Provider first receives the request including its type and enters the Request Received state. It then has to decide, based on the type, which path it takes, as by receiving this request type it enters the Feature request branch. That is, the first two states as well as the decision transition are the same as for the other request types. With entering the appropriate branch, the Provider enters the state Feature Request Received by actually receiving the Feature id from the Receiver. To reach the next state Log Retrieved, the Provider accesses its storage and retrieves the appropriate TrustLog structure for the Feature id received in the request. If there is no TrustLog existing for the Feature id given, the Provider responds with an empty message to the Receiver, thus indicating that the request Feature was not found. For the ease of understanding, this part is not directly shown in the state machine's figure but should be taken into account when implementing the Provider's logic. Given that the TrustLog could be retrieved, the Provider is now in the state Log Retrieved. It can now proceed further by starting the actual trust calculation based on the TrustLog's elements. The first step is to calculate the three security levels for phase one. This is done by taking the last snapshot stored in the TrustLog and thus getting the stored SPR for each operation. Those three SPRs are then used to retrieve the three $RSPR$s, which include the actual rating of the SPs by querying the $SPRM$. The $SPRM$ accesses the SPM and retrieves the currently valid ratings as already shown above. Using these ratings, the SL for each of the three operations can be computed and additionally stored in the appropriate phase element of the TrustLog. Finishing this step, the Provider enters the state P1 SLs Calculated. This state is not only used exclusively by this type of request but also by the reasoning request type. Due to this, the Provider re-checks the request type to proceed into the correct branch: In case of this request type it needs to proceed with the phase 2 calculation. This is done by retrieving the SPR and based on this building the $RSPR$ for the final transmit operation. It is possible to get these values at this time, as the Provider knows which Receiver requires to access the Feature by simply checking which Receiver was the source of the request. Based on this information, the $SPRM$ is able to determine the correct security properties for this operation and thus attaching the correct ratings into the $RSPR$. After this has been done, the Provider is in the P2 SPR Retrieved state from where it continues by adding the SPR just retrieved into the phase 2 element. Finishing this, the Provider is in the P2 SPR Added state which indicates that all necessary information for the next few steps are available. This makes it possible to start the actual calculation process. As it has already calculated the appropriate SLs for phase one, it only needs to calculate the last SL for phase two. It therefore takes the $RSPR$ gained in the last step and performs the SL calculation, finishing it by putting the SL into the correct phase element as part of the TrustLog. By doing so, it reaches the state P2 SL calculated where it holds all four SLs in their appropriate phase elements. It is now able to process to the next state by calculating the final TL_{ζ}^{P2}. This is done using the appropriate function to combine the SLs into one single value. This value is then written into the TrustLog. After this has been carried out, the Provider enters the P2 TL Calculated state which represents the final state of this request's calculation part. The only step remaining is the actual providing of the TrustLog. This is done in the next step: the Provider transmits the TrustLog (and the attached Feature) to the requesting Receiver, thus entering the state Data Provided. This finishes the handling of this request type. As already explained in the first request type, the Provider enters the state Waiting again in order to continue with further requests. The algorithm to handle this request type is shown in the following. It combines the already shown algorithm 2 and algorithm 4.

Algorithm 10: Handling of Receiver Requests : handleReceiverRequest(FeatureId).

```
Data: FeatureId identifying the requested Feature
Result: Transmitted TrustLog including the Feature for the id
TrustLog = getTrustLogByFeatureId(FeatureId);
Snapshot = getLastSnapshot(TrustLog);
$SPR$[3] = getAllSPRs(Snapshot);
$RSPR_1$ = getRSPR($SPR$[0]);
$RSPR_2$ = getRSPR($SPR$[1]);
$RSPR_3$ = getRSPR($SPR$[2]);
$SL_1$ = calculateSL($RSPR_1$);
$SL_2$ = calculateSL($RSPR_2$);
$SL_3$ = calculateSL($RSPR_3$);
$RSPR_4$ = getFinalTransmitRSPR();
$SL_4$ = calculateSL($RSPR_4$);
calculateP2TL($SL_1$, $SL_2$, $SL_3$, $SL_4$);
```

The algorithm itself consists mainly of calls to already defined functions, thus it combines the parts shown above into one.

Reasoning

The last request type the Provider is able to handle is the so-called reasoning request. The branch inside the state machine is shown in figure 4.12.

Aim of this type is to provide a possibility to retrieve the TL_{ζ}^{P1} for each Feature stored on the Provider. This allows to perform a trust-based reasoning on top of the trust values of all known Feature which could be for example used to gain an overall impression of the environments trust situation. The reasoning request branch of the Provider's state machine consists of two main parts: the retrieval of the appropriate TrustLogs and the

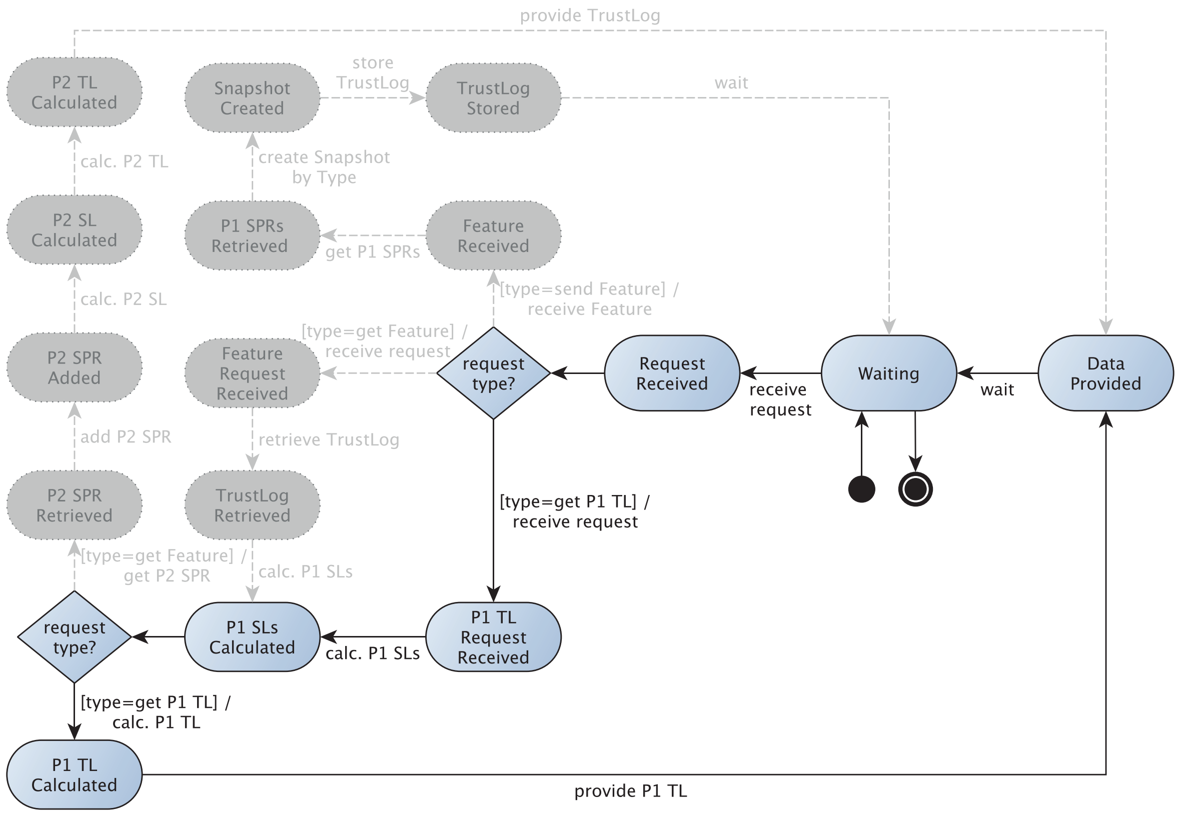


Figure 4.12: Reasoning path inside the state machine (shown as coloured elements).

calculation of all necessary TL_{ς}^{P1}. In detail, the reasoning request itself consists of a list of Feature ids or nothing but the request itself. If the request is received without any Feature id, it indicates that the Provide should answer with all trust levels for all Features it has stored at this time. The handling process of this request type begins as usual with the Provider receiving the request itself and thus entering the state Request Received from the former state Waiting. It then decides about the request type, which is in this case a reasoning request and enters the appropriate branch by reaching the state P1 TL Request Received. If there are Feature ids given, it retrieves only these TrustLogs or if there are no ids given it retrieves all stored TrustLogs to continue. The next part equals the phase one SL calculation part of the receiver request type with only one difference: it is done multiple time. That is, the calculation of the appropriate SLs is done for each Feature id received or for each Feature stored on the Provider. After this was finished, the Provider is in the P1

SLs Calculated state. This is the same state it reaches when handling a receiver request, thus the Provider has to check the request type again for further proceeding. In this case, the type indicates, that a reasoning is requested and thus the TL_ς^{P1} for each Feature given is required. Due to this, the Provider takes the calculated SLs for each TrustLog (i.e for each Feature) and combines them by using the $tF()$ into the TL_ς^{P1}. As a result of this, the Provider reaches the state P1 TL Calculated where it holds a list of all calculated TL_ς^{P1}s. To finish this request type, the Provider needs to transmit the calculated trust levels to the requesting component. The particular method for transmission is not defined within this model although the model demands that the channel used for transmission must be secured to avoid compromise of the calculated values. This could be done for example by signing the trust levels appropriately. After the transmission is finished, the Provider enters the state Data Provided which renders the final stage for this request type. As with the other request types, in order to be able to handle further requests, the Provider waits again and thus finishes in the state Waiting.

The algorithm used for this request type is shown in algorithm 11. It makes use of the SL calculation (algorithm 4) as wells as the P1 trust level calculation (algorithm 6). For simplicity purposes, the retrieval of the $RSPR$s for each SL and their connected operations are encapsulated into `getSL3ByFeatureId(id)`. The detailed algorithm used to perform these intermediary steps are shown in the algorithm below.

As described above, the algorithm decides if it should process all stored Features or only a limited list and then calculates all necessary parts. Based on these parts, the trust level is calculated and stored into another list which is then sent to the component which was requesting the reasoning.

Summary

The state machine of the Provider is responsible for the correct handling of the request types supported. There are three of these types: the Sender Request, the Receiver Request and the Reasoning Request. The Sender Request is triggered by a component possessing the Sender's role and consist of the sub types add or delete. Add indicates, that the sender wants to update or add a new Feature to the Provider's storage. The Provider takes the Feature in this case and stores it along with the Feature's attached security properties. In case of a delete subtype, the Provider tags the appropriate Feature as deleted in its

Algorithm 11: Handling of Reasoning Requests : handleReasoningRequest(FeatureId).

```
Data: Feature id list fids
Result: Transmitted list of P1 TLs
if $fids.size > 0$ then
    $P1TL[fids.size]$;
    for $i = 0; i < fids.size; i++$ do
        $SL[3]$ = getSL3ByFeatureId(id);
        $P1TL[i]$ = calculateP1TL($SL[0], SL[0], SL[0]$);
    end
else
    $P1TL[ProviderStorage.FeatureCount]$; i = 0;
    foreach $\zeta$ : storedFeature in ProviderStorage do
        id = storedFeature.getId();
        $SL[3]$ = getSL3ByFeatureId(id);
        $P1TL[i]$ = calculateP1TL($SL[0], SL[0], SL[0]$);
    end
end
```

storage. The second type, the Receiver Request, is triggered by a Receiver which wants to access (i.e. retrieve) a Feature. It tells the Provider therefore the Feature id. Based on this id, the Provider retrieves the Feature out of its storage and calculates the trust level (phase 2) for this Feature. The calculation is based on the currently valid rating values of the attached properties. After the TL has been calculated the Feature including its Trust properties is sent to the requesting Receiver. The third type of requests is the Reasoning Request. It used to retrieve an arbitrary amount of phase one trust levels from the Provider. This list can be used for further tasks, e.g. for evaluating the overall security situation of the environment. If the Provider receives this type of request, it calculates the appropriate TL (phase one) for each Feature requested or for all Features stored. It then transmits these calculated trust levels to the requesting component. These three request types map the necessary functions given in the model to the Provider and allow to actually use the trust model.

4.3.5 Policy Encapsulation

Although all required parts of the trust model are now defined, there is one additional point that needs to be addressed: the integration of a policy. Section 4.2.2 defines a part of this policy already which needs to be extended in order to encapsulate all necessary information and make it available to the Provider. That is, there needs to be a policy which is used by the Provider and holds all relevant information in a domain-specific way. This section therefore defines the abstract policy for the Provider, which is then mapped into the environment.

The policy consists of three parts, whereas two of these parts are already defined by the SPM. Furthermore, the policy holds another section, abbreviated FS, which encapsulates the necessary functions as well as auxiliary definitions needed for these functions (e.g., domain of the values, value constraints, ...). That is, the policy is defined as the SPM part and the functional part. The SPM part is divided into the already explained static and dynamic part. The static part holds the security properties while the dynamic part holds the ratings for these properties. Although the SPM may also be referenced rather than directly included, a complete policy is defined as the following.

$$
\begin{aligned}
&Policy := \\
SPM &\begin{cases} \{SP_{PropertyIndex} : (PropertyName)_{operation} := \omega_{RatingIndex}\}, \\ \{\omega_{RatingIndex} = RatingValue\}, \end{cases} \\
FS &\begin{cases} (rF(\{\omega\}) = \omega_1 \circ \omega_2 \circ ... \circ \omega_n), \\ (tF(\{SL\}) = SL_{process_{Sender}} \circ SL_{transmit_{Sender}} \circ SL_{process_{Provider}} \circ SL_{transmit_{Provider}}), \\ \{\text{auxiliary definitions}\} \end{cases}
\end{aligned}
$$

The domain-specific policy may use a certain policy language which allows to express the elements defined above. This section does not determine this specific language as there may be different languages in different environment. If instantiating the overall model for a specific environment, the language needs to be determined appropriately.

4.4 Domain-specific Extension - TCADS

The remaining part of the overall concept is the domain-specific mapping of the previously introduced approaches, thus building up TCADS [18]. TCADS abbreviates Trustworthy Context related Anomaly and Signature Detection for smartphones. First of all, the cases which have been explained in section 4.1.3 are extended in terms of the ability to measure trustworthiness. This is done by mapping the conceptual parts to the two CADS-specific cases. That is, while the role model described in 4.1 abstracted the CADS-specific cases to one single general representation, this section takes this representation as well as the trust model based on this representation and maps it back on the specific cases, thus making it specific again.

4.4.1 Case I

As already described, the first case summarises the process where a Feature Collector sends a Feature to the Correlation Engine. The trust extended architecture is depicted in figure 4.13. It is based on the architecture that was used without trust.

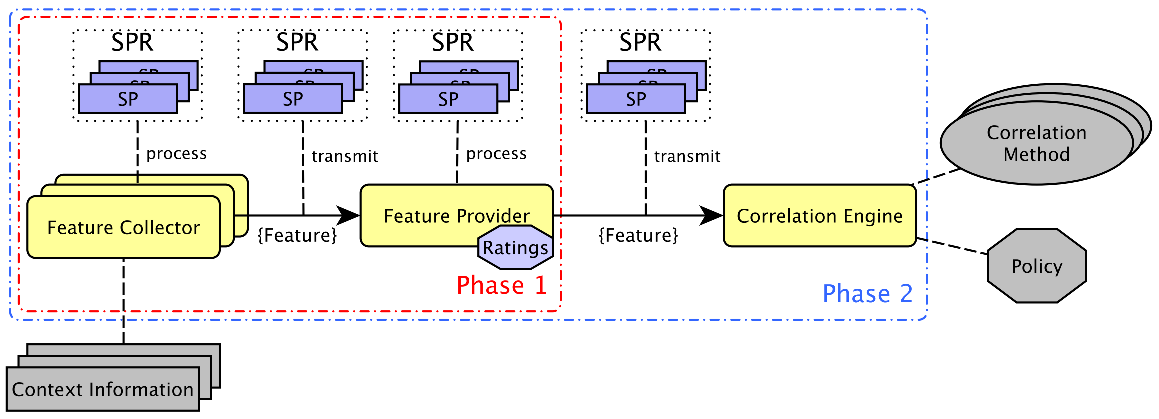


Figure 4.13: Case I in TCADS.

As shown in figure 4.13, there are defined operations that are carried out in the order given within the abstract model. That is, the Feature Collector first processes the Feature, then transmits it to the Feature Provider where it is processed again. Finally the Feature is transmitted to the Correlation Engine and used further. Due to this, the roles which where given to the components are the Sender for the Feature Collector, the Provider role for the Feature Provider and the Receiver role was assigned to the Correlation Engine. Each operation that is used to handle the Feature is extended by an appropriate security property record which is then being used to calculate the Feature's trust level. Furthermore, the architecture contains both defined phases. The first phase runs until the Feature Provider finishes the processing of the Feature while the second phase includes the final transmit operation from the Feature Provider to the Correlation Engine. The Feature Provider also holds the Policy used for the trust-specific parts (SPM, functions).

4.4.2 Case II

The second case expresses the situation where the Correlation Engine sends a Feature to the Feature Consumer (via the Feature Provider), for example an enforcement decision which should be processed by a firewall component. The extended version is depicted in figure 4.14.

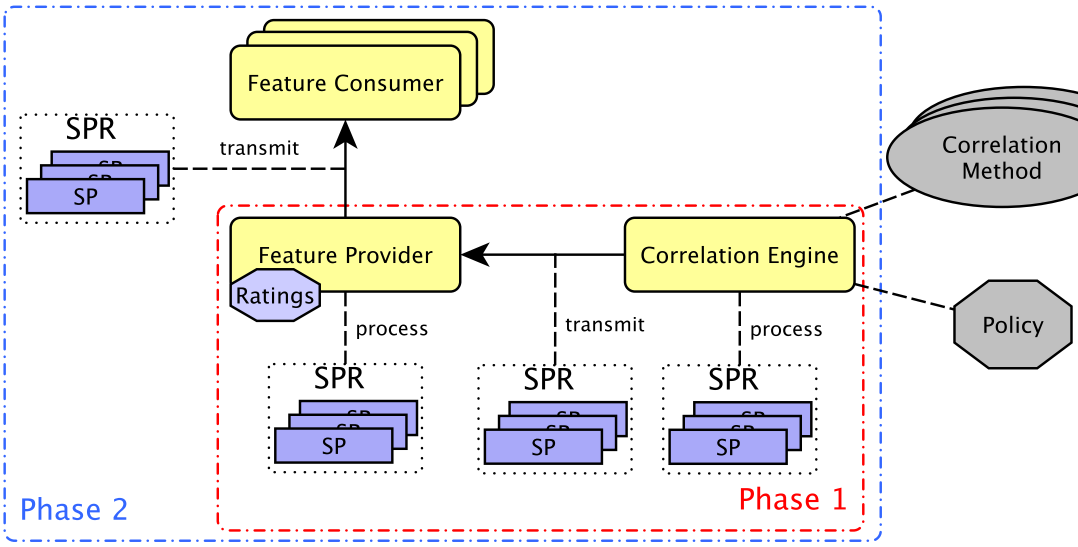


Figure 4.14: Case II in TCADS.

Equally to the extension of the first case, there are now components which possess the role of the Sender (Correlation Engine), the Provider role (Feature Provider) and the Receiver role (Feature consumer). The appropriate phases are also mapped, thus phase one summarises all operations but the last transmit while phase two includes this last transmit. Each of the operation is extended by a particular SPR being used to calculate the TL. As in the first case, the Policy is located on the Feature Provider again.

Besides the mapping of the cases itself, the TCADS architecture provides the following additions by using the trust model.

4.4.3 TCADS Features

Given the Correlation Engine with the CADS architecture, there is the already described problem of determining if a Feature is trustworthy. That is, the CE cannot distinguish between Features which are correct (in terms of their encapsulated data) and thus useful and Features that may not be useful due to providing false data. Using the trust extension (the complete TCADS system), the Correlation Engine is now able to make a decision in terms of a Feature's trustworthiness. To do this, the CE simply uses the calculated trust level of the Feature and the attached snapshots. In addition to the decision of an existing Feature's trustworthiness, the CE can determine if a deletion of a Feature has been trustworthy as well. If, for example, the CE receives two Features which measure the same issue but with different actual measurement values, the CE would use the one with a higher trust level. In particular, this for example may happen if one of the Feature Collectors has been compromised. The trust extension provides a way of recognising this.

Besides using a Feature's trust level directly, the CE can now be used to influence the trust calculation itself. This is done by defining rules which operate on certain trust-specific properties, for example on a SP's rating. In detail, if the CE recognises that a Feature Collector provides implausible values (i.e. Features), it can lower the rating of certain SPs which characterises this Collectors process operation. To do so, the Correlation Engine may use a special Feature type to directly communicate with the Feature Provider and calling the $SPRM$ to change these ratings. This change will then happen but only if the trust level of the CE's Feature is high enough. If the ratings could be successfully changed, the Feature Provider would recalculate the Feature's trustworthiness. This method allows to create a cascading system between the Correlation Engine and the Feature Provider, being able to make decisions in multiple communication rounds (CE to FP, FP to CE, CE to FP).

Another addition which may be used by the Correlation Engine is the direct correlation of trust values. That is, instead of using a Feature's value for correlation purposes, the CE can now make use of the Feature's trust level. E.g. the CE could monitor a certain Feature's TL and base a trend analysis on this level. If the trend becomes too negative, further investigations could be triggered by the CE. Besides using one trust level only, the CE could use a reasoning request and monitor all calculated TL_{ζ}^{P1} for the environment. This allows for a large scale monitoring.

4.4.4 Direct and Indirect Trust

Going back to the concept of direct and indirect trust which was presented in section 3.1.3, both types of trust derivation can be found in the TCADS system. The concept of using security properties for a basic trust foundation allows for having particular properties that can express either direct trust or indirect trust. Direct trust is usually inherited by specific properties of an operation, like the used algorithm or the used security layer for that particular operation. Indirect trust, usually a representation of a system's reputation, can also be expressed via the construct of a security property. In difference to the direct trust expressing SPs, which are set by the use of the $SPRM$ deriving the actual ratings from particular properties like a remote attestation, the indirect trust expressing SPs are rated by external[1] components. In detail, the TCADS system is able to express indirect trust by the use of a cascading relationship between the Provider and the Correlation Engine. If the Correlation Engine recognises certain Feature Collectors as untrustworthy, it can report this back to the Feature Provider. The Feature Provider uses this information and changes the appropriate rating of the corresponding SP. In fact, this is the process of changing an operation's trustworthiness by a particular reputation given for this operation from a third component. Due to this, providing special SP which are based on such cascades allow for an easy use of indirect trust. Besides using the Correlation Engine for this task, another external component may be leveraged to report reputation of a certain component. This is then mapped back to a particular operation carried our on or by this component.

[1]This can be also done by leveraging the $SPRM$'s capabilities but is triggered from outside.

4.5 Assessment

This chapter introduced three parts forming the trust model and by using this model, the domain-specific TCADS system. The first part, the generic model, introduced roles and operations. Roles are later used to map all CADS-specific cases into one architectural layout while operations express the abstract handling of the Features in the CADS system. Roles as well as the operations allow the abstract and generic definition of the trust model. This is done in the second conceptual part which introduced a reasonable definition of trust. So called security properties and their evaluational parts, the ratings, are then used to derive the actual trust measurement. This is done by calculating security levels for each operation that is part of the Feature handling process. These security levels are finally combined to one single value which is called trust level. To address the different stages of the Feature handling, the trust level is divided into a phase one and phase two part. Phase one represents all operations until the Feature is stored on the Provider while phase two adds the transmission to the Receiver. Based on this, a trust level for each phase can be calculated. The calculation itself is done on the Provider's side. The trust model defines a state machine therefore, which encapsulates all request types that the Provider can handle. The third conceptual part takes the trust model and maps the concept back to the particular TCADS cases, thus enhancing the architecture and giving a domain-specific system to actually deploy. Finally, the additional features that can be used on the Correlation Engine's side and benefits of the extension were shown.

To conclude this chapter, a detailed summary of the fulfilment of the requirements is given in the following. It is shown, which particular part of the concept can be used for certain requirements.

R1: Trust specification, calculation and evaluation Fulfilment of this requirement allows to distinguish between Features in terms of their trustworthiness. Using the approach presented here, this is done by defining security properties and calculating a trust level based on these properties for each Feature. The security property approach allows to specify and derive trust for a particular operation. The trust level provides trust calculation and evaluation. There are two sub types of this trust level: phase 1 and phase 2. The phase 1 trust level allows to generally evaluate the Feature based on the handling of the Feature until stored and kept ready on the Provider. Phase 2 allows the final Receiver to evaluate the Feature, but this time also including the last handling on the Provider. Furthermore, the evaluation approach is based on a holistic foundation, as it includes all handling operations for a particular Feature, thus allowing to express different parts for the overall trustworthiness. The domain-specific extension, TCADS, provides an evaluation method for a Feature's trustworthiness by introducing a trust-aware decision making system for the use in smartphone environments. Based on TCADS, an actual evaluation of a Feature's trustworthiness is possible for the named scenarios. Given these points, in particular the trust level concept of the generic trust model and its domain-specific usage, this requirement can be considered completely fulfilled.

R2: Trust history To provide a method for keeping track of Feature changes and in addition to that to changes of a Feature's trustworthiness, a history function must be provided in order to fulfil this requirement. The approach presented here does this with the so-called snapshots. Each time a change of the Feature is made on the Provider, a particular snapshot for this change is allocated. Furthermore, not only for value-based Feature updates but also for trust changes, which might happen if a rating of a particular security property is changed, a new snapshot is created. This allows to overcome the problems of losing track of the Feature's life cycle as well as losing the Feature if it is to be deleted. The snapshots are maintained on the Provider's side. Due to this, the last transmit operation between the Provider and the Receiver is not included within the snapshots stored on the Provider. If it is necessary to keep track of this part as well, the snapshot must be implemented additionally on the Receiver, which is also possible by using the snapshot definition provided above. Summarising this, it is possible to create a very sophisticated history mechanism by using the snapshot concept, thus this requirement is also fulfilled.

R3: Trust-based correlation In order to use the trust calculation and history features, it must be possible to not only recognise a Feature's trustworthiness but also to correlate on this trustworthiness. This is provided by the extension of the Feature-based data model. In detail, the Feature structure is embedded into the so-called TrustLog. The TrustLog encapsulates not only the Feature itself, including all relevant information like the Feature's value or the contextual information, it also encapsulates all trust related parts, like the trust level and the snapshots. Given that, the decision making system is able to access the trust-based values and include them into its correlation process. In particular, the domain-specific TCADS system uses a Correlation Engine which is able to access the TrustLog data structure. This is not only possible in an exclusive way, i.e. either correlating on a Feature's value or its trust, but it is possible in a mixed way. This allows to combine the trust level along with trust changes and the standard Feature properties like value and context. Considering this sophisticated mechanism, the requirement of providing a way to correlate trust-based can be seen as fulfilled by using the data model of the generic trust model and its usage within the TCADS system.

R4: Extending policies with trust To express trust related conditions it must be possible to include them into a particular policy. The concept presented allows this in two ways. The first way is given by the generic trust model, which defines the appropriate policy structures that are used for the trust-related parts. That is, the Provider's policy is used for trust-based reactions. This policy defines the building blocks, like the relevant security properties and their ratings, for particular operations. As these operations form the overall handling procedure and hence provide the parts for the overall trust calculation, the policy can be used to express reactions on trust. Although this gives a generic method for a policy inclusion, it does not provide a directly usable domain-specific method. This usable method is given by the second way which is the ability to express trust-based conditions within the standard policy used by the Correlation Engine. This is due to the trust enhancing data model, which allows to express trust as another contextual value for the Feature. Furthermore, as the Correlation Engine can act directly upon a Feature's trust, it is possible to base decisions upon this trust. Those decisions can be expressed within the Correlation Engine's policy. Given these two ways, it is possible to express arbitrary trust-based reactions for a particular domain, thus fulfilling this requirement.

R5: Extensibility in terms of used data and trust calculation methods As the generic trust model should be usable for particular different environments and domains, each with its own potential set of devices and trust sources, it is required to allow for a very flexible usage of data and trust sources. Furthermore, with different sources, the method of calculating the trustworthiness needs also to be very flexible. This is done by (1) using security properties for trust derivation and (2) by using freely definable functions in combination with intermediary steps for the task of trust calculation. The security properties (1) can be defined freely. By adding ratings to them, which are the actual atomic trust measurement, it is even possible to weight trustworthiness, not only by strong or weak but also by untrustworthy and trustworthy. That is, the property concept allows also to express negative trust. Using these properties and in particular their ratings, a trust level can be calculated. This is done (2) by providing a two step calculation approach. First, the security level of a certain operation is calculated in detail based on the properties. Second, all security levels are combined into the overall trust level valid for each operation being part of the Feature handling process and thus expressing the Feature's trustworthiness. Both functions used, the one for establishing the security level as well as the one used for combing the overall trust level, are freely definable. Given that and the property concept, it allows to tailor the trust model for the scenario it is used in. Such a tailoring is done within the domain-specific TCADS mapping which presents an usable way of deploying the trust model within smartphone environments. Furthermore, the two-step approach used for calculation allows to see every single part of the calculation, starting by the properties and their ratings via the security levels and ending at the phase 1 and 2 trust level. Due to this, it is not only possible to define relevant properties freely for one scenario, but also to use different definitions within one scenario, making the approach very flexible and thus fulfilling this requirement.

R6: Ability of seamless integration To actually use the generic trust model, it is required to provide a maximum of integrability. This is necessary in order to support a wide field of particular environments with their own infrastructure and components. The specified role model which introduced generic roles along with generic operations to picture an environment on a very abstract level can be used to achieve this requirement. In detail, not only roles and operations are defined but also a process to adopt this model to a particular scenario. It shows how to map the relevant functions into the environment. For the environment given here, the TCADS system represents this domain-specific mapping providing a good integrability on the conceptual level. However, the concept does not define the actual implementation used to provide the TCADS system. Due to that, the requirement can be considered fulfilled only on a conceptual level as the complete fulfilment of this requirement depends on the implementation used.

Given this summary, it is easy to see that most of the requirements are addressed in a satisfying manner. The integration requirement needs to be assessed on a particular implementation which is done later in this thesis.

In addition to the assessment of the requirements, a short review of the research questions can be given. The first question "Which data and characteristics can be used to derive trust?" can be answered with the security properties. That is, security properties along with their ratings are used for the task of trust derivation. The second question "How can trust be defined and calculated?" is answered by the trust level. In detail, trust is expressed by calculating a trust level for a Feature whereas the calculation is based on several steps beginning with the combination of the security properties. "How is it possible to use trust in the decision making process?" can be answered with the provided TrustLog and the policies used. I.e., the trustworthiness and all its parts including history information is provided within the TrustLog for a Feature. Furthermore, this TrustLog can be addressed within a policy to base decisions up on it. These three answers provide a satisfying conceptual solution for the overall problem. To go more into the technical part, a proof of concept implementation is provided within the next chapter.

5 Implementation

Contents

The previous chapter introduced the overall approach of a trust enhanced method for decision making in a particular smartphone-enabled environment. This overall approach was used for a domain-specific mapping which forms the TCADS system. This chapter introduces a proof of concept implementation for this approach, in particular a proof of concept implementation of the TCADS system. It is partly based on the already existing proof of concept implementation of the CADS system which is explained in [2]. Due to the CADS implementation relying on the IF-MAP protocol, IF-MAP is also used for the proof of concept presented here. The chapter itself consists of three main parts: a detailed view on the IF-MAP protocol, including a summary of trust relevant properties of IF-MAP, the introduction of the overall approach for implementing the TCADS system in IF-MAP by injecting trust and finally the actual technical implementation.

5.1 IF-MAP Revisited

As already explained in section 3.2.3, the TCG's IF-MAP protocol (cf. [152]) provides a content-based publish subscribe approach to share network-specific data among all IF-MAP enabled components. It is based on so-called MAP Clients (MAPC), which can either collect and publish data or subscribe to certain data thus receiving them. Besides the MAPCs the so-called MAP Server (MAPS) provides the central database for the shared data. That is, it receives the data from a MAPC by this MAPC publishing the data to the MAPS and provides data to another MAPC by sending them data. Sending is realised with a subscription model, thus the particular MAPC must subscribe to data in order to receive information on changes to those data. Using IF-MAP allows for an interoperable exchange of arbitrary data within a network.

From a technical point of view, IF-MAP is entirely based on existing technologies and protocols. The secure hypertext transfer protocol (https, [160]) is used by IF-MAP for realising the basic communication. This communication is provided in a secure way as it relies on the https protocol. Furthermore, TLS [159] is used within this context, which allows to utilise all capabilities TLS offers. Layered on this, the SOAP protocol (cf. [161, 162]) is used to allow an interoperable communication between the MAPCs and the

MAPS. SOAP itself relies on exchanging XML-based (cf. [163]) messages and thus the IF-MAP specific data is defined within an appropriate XML schema (cf. [164, 165]).

5.1.1 Data Model

The data model used by IF-MAP consists of the following three elements.

- Identifiers which are used for identification purposes,
- Links which connect identifiers and metadata to each other as well as
- Metadata which consist of descriptive data.

By the use of these elements a graph is formed. Figure 5.1 depicts a typical graph, consisting of all elements and two distinct sub graphs. The example is based on sharing network security related information. Within this figure, identifiers are depicted as ovals,

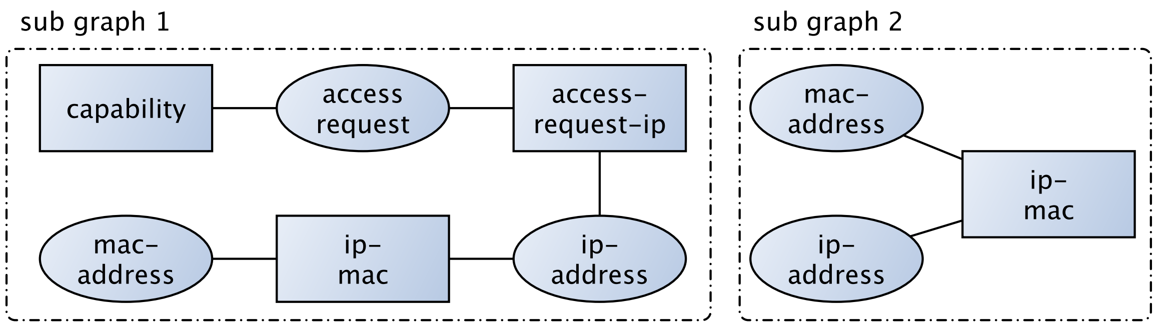


Figure 5.1: Example MAP graph (cf. [166]). Identifiers are shown as rectangles, ovals depict metadata and lines express the links between the particular elements.

metadata is depicted by rectangles and links are depicted as connective lines between the other elements. Details about each type of element is given in the following.

Identifiers

This type of element expresses information which are able to uniquely identify entities of a network. Such entities may be either devices and their interactions or users. In case of devices, they may for example be identified by their MAC address and in case of user related information, the user may be identified by her username. Identifiers cannot be created or deleted, they implicitly exist at each time and are only visible within the MAP graph as long as they have connections via links or metadata to other elements of the graph. IF-MAP specifies five different types of such identifiers [152] which are given in the following.

access-request This type of identifier represents a physical request to access a network made by a device. It is characterised by a unique id which allows it to be identified. Access requests are usually handled by Network Access-based components, for example a Policy Decision Point from the TNC architecture. If an Access Requestor would try to access a network, the PDP would publish the appropriate data and attach it on to the access-request identifier. The XML-based specification is as follows [152].

```
<xsd:complexType name="AccessRequestType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
<xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
```

Listing 5.1: IF-MAP XML schema for access-request identifier.

device A device identifier is used in order to depict a physical or virtual device within the MAP graph. IF-MAP knows two types of devices: devices which gain access and devices which may provide access or services. Given the example case of the access-request from above, there may be either a device for the PDP attached to the access-request as well as a device for the AR. The XML-based specification is as follows [152].

```
<xsd:complexType name="DeviceType">
  <xsd:choice>
    <xsd:element name="aik-name" type="xsd:string"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:choice>
</xsd:complexType>
```

Listing 5.2: IF-MAP XML schema for device identifier.

identity An user is identified using this type. In detail there are several sub types of this identifier which allow to express several information used for this identification purpose. Examples for such sub types are the username, the email address or even a DNS name. Furthermore, a so-called other type definition can be made. This allows to define own sub types for this kind of identifier. Given the CADS implementation in its current version, this other type definition is used to express the category-based graph structure. The XML-based specification is as follows [152].

```
<xsd:complexType name="IdentityType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="type" use="required">
  <xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="aik-name"/>
    <xsd:enumeration value="distinguished-name"/>
    <xsd:enumeration value="dns-name"/>
    <xsd:enumeration value="email-address"/>
    <xsd:enumeration value="kerberos-principal"/>
    <xsd:enumeration value="username"/>
    <xsd:enumeration value="sip-uri"/>
    <xsd:enumeration value="tel-uri"/>
    <xsd:enumeration value="hip-hit"/>
    <xsd:enumeration value="other"/>
  </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="other-type-definition" type="xsd:string"/>
</xsd:complexType>
```

Listing 5.3: IF-MAP XML schema for identity identifier.

ip-address This identifier expresses the ip address for a certain device. This ip address may be either of the type IPv4 or IPv6. The XML schema [152] is specified as shown.

```
<xsd:complexType name="IPAddressType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
  <xsd:attribute name="type">
```

```
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="IPv4"/>
      <xsd:enumeration value="IPv6"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
```

Listing 5.4: IF-MAP XML schema for ip-address identifier.

mac-address As with the ip address, the same applies for this identifier which simply expresses a device's mac address. The schema defined in [152] specifies it the following way.

```
<xsd:complexType name="MACAddressType">
  <xsd:attribute name="administrative-domain" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>
```

Listing 5.5: IF-MAP XML schema for mac-address identifier.

Given these identifiers, IF-MAP allows to widely define roots for identifying certain entities. Furthermore, the latest version of the IF-MAP specification allows even to define own identifiers, which are then called extended identifiers. This allows for a more versatile use of IF-MAP as one is able to define an environment tailored identifier for arbitrary purposes. The base definition [152] for such identifiers are shown in the following.

```
<xsd:complexType name="IdentifierType">
    <xsd:attribute name="administrative-domain" type="xsd:string" use="
        required"/>
</xsd:complexType>
```

Listing 5.6: IF-MAP XML schema for extended identifiers.

Links and Metadata

To establish a relationship between two different identifiers, a link is used. Links are always given by the particular metadata attached to them. That is, if no metadata is attached, the link is no longer valid and is thus being deleted. Furthermore, metadata may not only be attached to a link between two identifiers but also directly to only one identifier. In the first case, the metadata expresses information about the relationship between the two identifiers connected while in the second case information about a single identifier are expressed. The basic schema of attributes that metadata must include is specified by the TCG as following [152].

```
<xsd:attributeGroup name="metadataAttributes">
  <xsd:attribute name="ifmap-publisher-id"/>
  <xsd:attribute name="ifmap-timestamp" type="xsd:dateTime"/>
  <xsd:anyAttribute/>
</xsd:attributeGroup>
```

Listing 5.7: IF-MAP XML schema for metadata.

The TCG allows to either use pre-defined metadata, for example in the area of network security or, due to the basic specification, provides a way to define an own set of metadata. Both types must be based on the attributes shown above and thus consist of so-called operational attributes. Three of these attributes are defined within the basic schema shown in 5.7. The remaining two must be defined within the appropriate metadata schema. Namely, the operational attributes are the ifmap-publisher-id, ifmap-timestamp, ifmap-cardinality, lifetime and anyAtttribute.

ifmap-publisher-id This attribute is used in order to uniquely identify a publishing MAP Client. When an MAPC initially connects to the MAPS by publishing metadata, this metadata is assigned with this id. Future metadata published is then also assigned with this id, which is never changing for this MAPC. The id itself is responsible for the assignment process and in detail also for the uniqueness of the id itself.

ifmap-timestamp The ifmap-timestamp is assigned when a MAPC publishes metadata on a MAPS. That is, when the server places this metadata within its graph, the time of this operations is assigned to this metadata. Only the server is able to set this type of attribute.

ifmap-cardinality This attribute defines if an instance of a particular metadata can be assigned only one time or multiple times to a link or to a identifier. This means, if this attribute is set to singleValue, then only one metadata of this type can be assigned at the same time. If it is set to multiValue, an arbitrary amount may be assigned at the same time. This behaviour is specified in [152] as follows, thus extending the basic metadata attribute group specification.

```
<xsd:attributeGroup name="singleValueMetadataAttributes">
  <xsd:attributeGroup ref="metadataAttributes"/>
<xsd:attribute name="ifmap-cardinality" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="singleValue"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
<xsd:attributeGroup name="multiValueMetadataAttributes">
  <xsd:attributeGroup ref="metadataAttributes"/>
<xsd:attribute name="ifmap-cardinality" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="multiValue"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```

Listing 5.8: IF-MAP XML schema defining single and multi value metadata.

lifetime This attribute determines how long particular metadata exists on the MAP Server. It may be either set to session or forever. Session lifetime indicates, that this metadata is to be deleted at the time the session of the publishing MAP Client is being terminated. That is, if that particular client disconnects from the server, the server deletes all metadata from this client which is set to session lifetime. In difference to that, the metadata won't be deleted if the lifetime is set to forever. In this case the metadata will never be deleted by the server. The value itself is set by the MAP Client and if no value is given, the server assumes session as default behaviour.

anyAttribute To allow for an extension of the metadata attribute group, the specification allows to assign more attributes via the anyAttribute construct. This allows to extend metadata by own attributes while still being conform to the specification. Although it is possible to freely define own attributes by the use of this it is not intended by the TCG. Due to this, it needs to be evaluated if there are other ways to do this while staying conform to the specification. Nevertheless it is possible to use this attribute for other purposes.

These three basic constructs, identifiers, links and metadata stored in an appropriate graph on the server, form the basic data model of the IF-MAP protocol. The actual exchange of messages building up instances of this data types are explained in the following.

5.1.2 Communication in Detail

The underlying communication model used by the IF-MAP protocols defines three parts:

- two different channels used to exchange data,
- operations which are carried out through these channels and
- the handling of IF-MAP sessions.

IF-MAP Channels

The start of each communication is always given by the MAP Client as the server responds only to requests and commands from the client. In detail, the client may use both types of channels to communicate with the server. It first establishes a so-called synchronous send receive channel (SSRC). Using this SSRC, all commands which can be immediately processed by the server are exchanged, e.g. the publication of metadata from the client. There is one operation, poll, which returns metadata to the MAPC. This poll operation might block the communication channel as it waits for changed metadata. Due to this, a second channel is required. This channel, called asynchronous receive channel (ARC), is also established by the client and exclusively used for polling.

Session Handling

As IF-MAP is a stateful protocol relying on the two communication channels, a session handling is provided. This session handling is controlled by the following commands and their appropriate operations.

newSession This operation is initially carried out and provides a session id for a MAPC. Due to this, the operation itself does not require a session id to be sent. The generated session id is given back to the client which uses it for all future communication with the server.

renewSession A MAPC's session id is treated as valid as long as the MAPC has an active channel to the MAPS. In order to allow the client to change the https-based connection without losing its session, the renewSession command can be used. If the session is not refreshed within a specific amount of time and there is no active channel to the client, the server terminates this session and thus the session id becomes invalid.

endSession To actively terminate a session from the client, the endSession command can be used. If the server receives this command, all metadata published by this particular client which has its lifetime set to session is deleted. Furthermore, the session id is considered invalid from this time.

By using these three commands, the client and server are able to handle their session in a satisfying way.

Operations

Within an established session, there are operations to actually exchange data between MAPCs and MAPS. There are five types of operations defined by IF-MAP which are explained in the following.

Publish The publish operation allows a MAPC to change metadata on the server. There are three sub types of this operation which define the actual meaning of the term changing metadata. Due to this, the particular operation may be either a publish-update, a publish-notify or a publish-delete.

update Adds or updates particular metadata on the server. In detail, if the metadata included in this operation does not exist on the server, it is initially added. If it is already existing and is of the cardinality type single value it is updated with a new one. If the cardinality is set to multiValue, another metadata is simply added.

notify Metadata provided by this operation is not stored within the server's graph. Instead, it is directly forwarded to all clients having a subscription for this kind of metadata.

delete Simply allows to delete particular metadata on the server.

One single publish operation may consist of an unlimited amount of combinations of all three sub types.

Search The search operation provides a way to retrieve metadata and identifiers connected by this metadata from the MAPS. The client needs to provide one identifier as root for the search operation. Based on this root all connected links, metadata and identifiers are put together into a searchResult. This searchResult effectively provides a sub graph of the overall graph stored on the server. To limit the effect of the search process, there are some parameters which define filter like conditions.

identifier As already written, this parameter defines the initial root which is used to start the search at.

match-links This option provides a method to filter the result of a search by the links that are traversed. That is, if there is a value given for this option, only links and their appropriate data are traversed if they match the given value.

max-depth Specifies the deepness of the search operation. As the map graph may become rather big over time, going into an unlimited deepness may result in a rather bloated answer. To overcome this, the max-depth specifies the amount of identifiers that should be traversed, beginning by the root.

max-size Another way of limiting the search result is the setting of this parameter. It specifies a maximum in terms of size the result can reach and instructs the search operation to stop at this point.

result-filter This parameter provides another option for filtering the search result. The value set for this parameter defines which entries of the result should be dropped and thus excluded from the search result.

terminal-identifier-type If reaching the identifier set in this parameter, the search operation is terminated and the result is returned.

These options allow for a highly customisable search operation.

Subscribe This operation is used by the MAP Client in order to select metadata on the MAP Server the client wants to be informed about. That is, each time the metadata changes on the server, usually by a previously done publish operation, the MAPC which has an active subscription for this data will be informed about the changes and the content of the change. Furthermore, there are two sub types of a subscription: a subscribe update operation as well as a subscribe delete operation. The subscribe update is used to add or change a subscription while delete is used by a client in order to delete an active subscription on the server.

Poll After a subscription has been established by the client on the server, the client can carry out the poll operation. This is done via the ARC as poll only returns if there is currently an actual poll result and otherwise blocks. If there is a poll result which is provided through the ARC, it consists of the name of the subscription that triggered this particular result. Thus, one client may differentiate between subscriptions on a single server. The poll result itself contains only changes of the data, thus omitting parts of the returned graph which were not changed. A poll result consist of one of the same sub types like a search result or a general search result. That is, a general result, simply called searchResult, is returned the first time the appropriate subscription matches. In this case, no partial elements of the returned graph are omitted. Furthermore, this type is only given once at the first time of a matching subscription to the client. The next time parts of the subscription are changed by a conventional publish, the client receives an updateResult. If something is deleted, a deleteResult is used. If the publish was done through a publish-notify operation, the poll returns a notifyResult, indicating the particular different publish causing the update.

PurgePublisher This last operation allows a MAP Client to order the server to purge all previously published metadata from this client. To do so, the client provides the correct publisher-id which is used by the server to identify the metadata that should be deleted. While this operation is not used for ending a session, as the endSession command takes care about deleting metadata with session lifetime, it is used if the client unexpectedly terminates. It allows the client to clean up all its metadata on the server to provide a new starting point.

Given this communication model, a deeper look into the trust model behind the IF-MAP protocol can be done.

5.1.3 Trust Model

Based on the analysis of the propagated trust model made in [166], there are two parts of this model: the IF-MAP protocol itself as well as the security of IF-MAP enabled components, in particular the server. IF-MAP proposes the idea of allowing to share network related data between arbitrary systems to not only gain an overall picture about that network but also to support a decision making. Given that, IF-MAP systems may be rather threatened. This is due to the fact, that an attacker is able to influence certain aspects, like the decision making, in order to gain an advantage. To limit this, the TCG proposes the named trust model with its two parts.

Protocol Trust Model

As stated in [166] and [152], the IF-MAP protocol itself must fulfil three requirements in order to be considered secure. These three requirements are:

- a mutual authentication of the communicating parties, thus an authentication of the clients and the server;
- the use of an integrity preserving protocol for data exchange;
- the support of encrypted data exchange along with the security against replay-based attacks.

A strong availability is not explicitly required, although the specification points to it as a possible improvement.

To fulfil these requirements, TLS is used. TLS provides strong and established means to authenticate the communicating parties, to encrypt data and to check the integrity of the exchanged data by the use of particular message authentication code mechanisms (cf. [167]). To ensure the MAP Servers authenticity, IF-MAP relies on an appropriate certificate of the server. That is, the client must only establish a connection with a server if the certificate presented by the server is considered to be valid. Additionally, IF-MAP provides two possibilities for authenticating the client: as so-called basic authentication and a certificate-based authentication. The basic authentication is a simple exchange of a username and password combination. This credential combination is then matched against a set of users the server holds. If the user is known and the credentials are correct, access is granted. As with the certificate-based authentication, the same procedure the client uses to authenticate the server is used. Thus, the client must possess a valid certificate which is being presented to the server in order to gain access. All these authentication mechanisms are based on the TLS protocol which provides well established means and can be considered as a secure way of doing this.

Trust Model for IF-MAP enabled Systems

The second part of the trust model is given by the requirements for the actual IF-MAP enabled systems. That is, this part aims to reduce the threats imposed on IF-MAP enabled systems and platforms. Given the analysis in [166] there are the following general points which are applicable for both, MAP Servers and MAP Clients.

- A hardened operating system should be used. Furthermore, the OS should be maintained actively.
- Physical access to the appropriate systems should be limited, thus only individuals which are allowed to access certain systems should gain physical access.
- A trusted platform should be used, thus allowing to perform a remote attestation of the system prior to accessing the network.
- In terms of providing forensic information for the case of an incident, a sophisticated logging approach should be used. This logging approach should keep track of all relevant events taking place on the particular system.

Besides these general points, there are some specific advises for either MAP Clients and MAP Servers.

MAP Clients

The specification advises the following points to increase a MAP Client's security.

- The MAP Server should provide a more sophisticated access control scheme than allowing full access if a MAP Client was successfully authenticated. That is, the server should allow to distinguish between read only access and read write access. This gives the ability to grant MAP Clients only the lowest rights necessary and if the client tries to perform operations which are not allowed within its current rights, access should be denied.
- Improving this basic access control scheme, the TCG advises to optionally use a scheme which allows to set each allowed operation individually for each particular MAP Client. Extending this, an administrative interface for this scheme should be established, allowing the administrator to be informed about a MAP Client trying to use not permitted operations.
- Honeypot like metadata can be provided by a MAP Server to recognise MAP Clients that may be compromised.

MAP Servers

The TCG advises certain points addressing the servers.

- The network access to the server should be (a) limited by a firewall and (b) monitored by an intrusion detection approach.
- A host-based system monitoring should be used.
- The server should not rely on basic authentication but only on the certificate-based method.
- An established ARC and an SSRC which are supposed to belong to the same client should be verified for this. That is, the server should be able to recognise the hijacking of certain communication channels by a rogue client.

- Maximum size of the search result should be limited to prevent denial of service attacks. Denial of service may happen if the server is unable to process the search in a reasonable time (i.e. due to the result growing bigger than the server's RAM).
- The last advise is carried out by MAPCs but it aims at recognising a rogue MAPS, thus this point also addresses server related security. MAP Clients should be (a) able to recognise behaviour of the MAP Server that is considered abnormal and (b) should check the validity of the received data. How this can be achieved in detail is not given by the specification.

Given that, the trust model defined by the TCG only addresses the communication itself and the participating system's security. There is no explicit trust model for the data being exchanged. A detailed view on this point is given in the following.

5.2 Trustworthiness of IF-MAP

As shown in the previous section, IF-MAP does provide communication security, i.e. published and subscribed data is transferred securely between the appropriate parties. IF-MAP also makes certain advises about the hardening of the systems participating as MAPCs and MAPSs, for example by the use of trusted computing-based technologies. However, IF-MAP does not provide mechanisms to reason about the trustworthiness of the (meta)data itself but states (cf, [152]) that this trustworthiness is explicitly out of scope. Furthermore, the network overview created by IF-MAP is also defined as one possible state the network could be within, not necessarily the actual state. Due to these reasons, there are several enhancements necessary. Given the trust definitions made in this thesis (section 4.2.1), which states *A Feature is considered trustworthy if and only if all operations handling this Feature perform as expected.*, a technical version of this definition can be made in order to use IF-MAP as the basic communication layer. This definition [166] simply states from a technical perspective, that trustworthiness is based upon the integrity of metadata. This integrity might be compromised either on the MAPC or on the MAPS but also on the communication channel used to transmit this metadata. That is, the metadata can be considered trustworthy if the IF-MAP based operations that handle this metadata behave as expected. Using this definition and the findings about IF-MAP's trust model, the particular concept for the implementation can be introduced.

5.3 IF-MAP Mapping of TCADS

This section describes the approach which is used to map the TCADS system to IF-MAP. In particular, the implementation is based on the CADS' implementation described in [2]. Due to this, this section only provides the trust-specific part of the implementation's concept. First of all, a recap of the TCADS system in terms of the trust extension is given.

5.3.1 TCADS Revisited

Section 4 introduces the overall trust model and the domain-specific TCADS system. These approaches consist of several parts, in particular of an abstract model defining the roles and their associated operations. Based on these roles and operations, trust enhancing components are added. Figure 5.2 depicts the combined model elements. As the

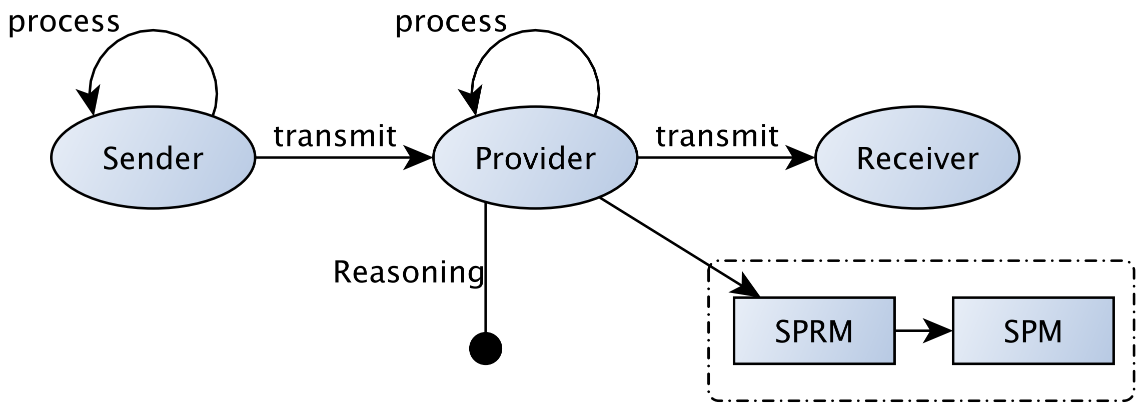


Figure 5.2: Summarised abstract trust model (cf. [166]).

figure shows, there are three roles: a Sender which creates Features, a Provider that is responsible for storing the Features and making them available and the Receiver which uses the Features. There are two operations, processing a Feature as well as transmitting a Feature. Furthermore, the Provider offers a special Reasoning interface and uses a component called security property record manager (SPRM). This SPRM relies on the security property map, which provides the conjunction of the basic trust derivation factors and their attachment to the appropriate operations. These factors used for trust derivation are represented by the security properties (SP), which are valid for a particular type of operation. Combining these SPs and their appropriate ratings, an intermediate security level (SL) can be calculated. There is one SL for each operation which has been part of the overall Feature handling process. Given that, the final trust level of a Feature can be calculated by combining these SLs into one value. Based on these key abilities, a mapping to IF-MAP is described in the following sections.

First of all, the instantiation process (4.1.4) describes roles that need to be mapped onto the appropriate components of the particular architecture. This is done in the following.

5.3.2 Abstract Role Mapping

IF-MAP is based on two different components, the MAPC as well as the MAPS. As there are three roles defined by the trust extension of TCADS, the following detailed mapping is used.

- **Sender Role:** The role which provides Features to the Provider is mapped to the component MAP Client. That is, operations carried out on and by the MAP Clients are either of the type process or transmit. Transmit is mapped to the actual publish operation of the metadata from the MAPC to the MAPS. All other Feature and metadata relevant operations, like measuring properties of the client's platform are mapped to the process operation.

- **Provider Role:** As the Map Server receives metadata from a publishing MAP Client and provides this metadata via a pollResult to another client requesting this data, it is mapped to the Provider role. Furthermore, with the incoming publishing operation already mapped, the outgoing pollResult based on an active subscription is mapped to the transmit operation started by the Provider. All other handling of metadata on the MAPS is mapped to the process operation.

- **Receiver Role:** Due to the previous mapping, the Receiver Role is also mapped to MAP Clients. The MAP Server provides metadata to those MAP Clients by handling their subscription and transmitting changed metadata to the appropriate clients.

Besides the particular role-based mapping, there are two more conceptual components which need to be addressed in order to completely specify the mapping. The first component is the SPRM, which is responsible for retrieving the actual security properties and their ratings out of the second component, the security property map. These two components may be either placed on the MAP Server itself or on a particular MAP Client. There are several advantages and disadvantages for both ways, as pointed out in [166]. Placing the components on a MAP Client would allow for a rather resource keeping strategy. Due to the fact that the SPRM may have to handle sophisticated tasks in order to assign the correct security properties, it requires some system resources. Placing it on a separate MAPC allows to assign all of the client's resources for this specific task. However, according to [166], this is the only actual benefit of this approach. In detail there are three particular problems if using a MAP Client as SPRM:

- A special MAP Client to MAP Server interface is required.
- Unavailability of the calculating MAPC.
- Limited MAPC security.

First of all, the MAP Client requires a special interface to the MAP Server. Although IF-MAP provides some means to exchange required data, those means do not fully allow to exchange all data from the MAPS to this MAPC. In detail the MAPC would need a starting identifier and, which makes it rather complex, it must be able to access all sub graphs stored on the MAPS. This is practically very hard to achieve, as the starting identifier needs to be pre defined.

The second problem concerns the availability of the MAPC as without it there is no trust level calculation possible. Although a MAPS may also be unavailable, the situation would result in metadata as well as the trust levels are not available. Relying only on a MAPC would result in the weird situation in case of the MAPC being unavailable that metadata is available but without a trust level.

The third disadvantage is about the MAPC's security itself. With providing an extra interface, this interface may be compromised. Furthermore, there are two critical components by using this approach: the MAPS and this MAPC. Even more critical, the connection between the MAPS and this MAPC may be interrupted resulting again in the situation with no trust levels available.

Given these points, the MAPS should be the place of choice. Emphasising this, besides the increasing resource requirements, the following benefits can be found. First of all, the MAPS receives all metadata that is exchanged due to its centralised position and function. Given that, all metadata exchanged is extended with trust if the SPRM is placed on the MAPS. Furthermore, the MAPS is the only component which is able to see most of the operations carried out to handle metadata. Although it cannot directly gain information about the processing of a sending MAPC, all other operations are within the MAPS's visibility. It is therefore rather easy to derive the security property attachment to the appropriate operations. Finally, the MAPS must be trusted under all circumstances. Due to this, placing the SPRM on it does not create a new instance which must be trusted in addition. Given all these benefits of placing the SPRM on the MAPS, the MAPS is responsible for carrying out the trust calculation including all necessary parts. Figure 5.3 depicts the situation after the appropriate mapping of the abstract roles to IF-

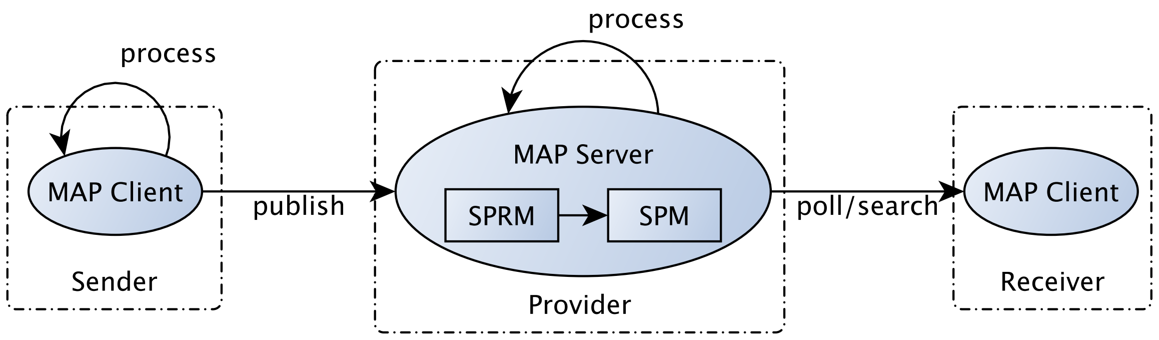


Figure 5.3: Mapping of Roles to IF-MAP entities [166].

MAP components. Furthermore, the IF-MAP defined operations are also included, thus transmit is carried out by either publish, the returning poll or a search operation. Using this mapping, the actual trust extension for the IF-MAP protocol can be introduced.

5.4 Trust Extension in IF-MAP

The overall trust extension used to apply the relevant conceptual parts of TCADS to IF-MAP is depicted in figure 5.4. It consists of two relevant layers: the extension layer itself and the interface layer. The extension layer consists of

- the Security Property Map (SPM) realising component,
- the Security Property Record Manager (SPRM) component and

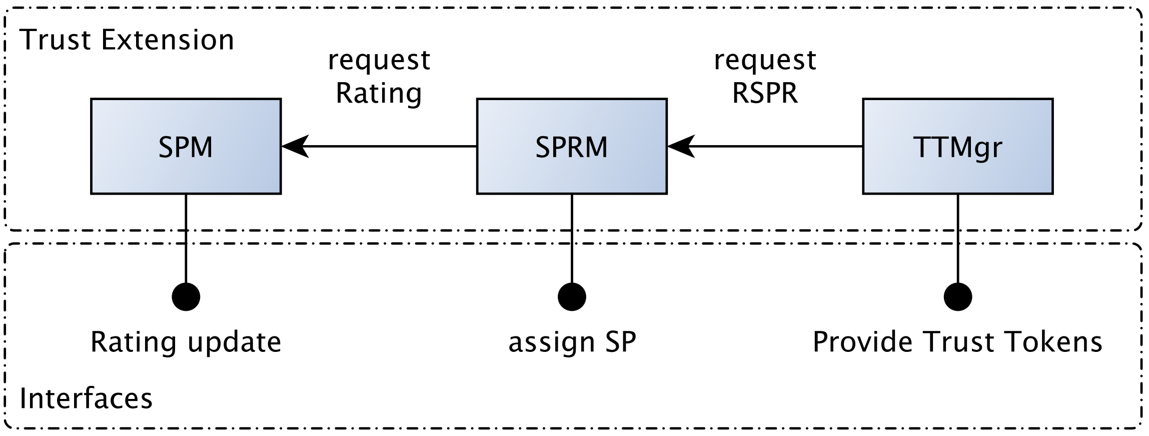


Figure 5.4: Trust extension of IF-MAP[166]: Trust layer consisting of relevant components and their exposed interfaces. In particular, the security property map (SPM), the security property record manager (SPRM) which request an appropriate rating from the SPM and the Trust Token Manager (TTMgr) which request a Rated security property record (RSPR).

- the Trust Token Manager (TTMgr, [166]).

The interface layer provides the accessible interfaces to each of these components. Each part is described later in this section. Prior to this, the mapping of the trust level into the domain of IF-MAP is introduced. This mapped trust level is defined as a so-called Trust Token (TT).

5.4.1 Trust Tokens

Section 4.2.3 and 4.2.4 introduce the trust level concept. In short, a trust level expresses the overall trustworthiness of a Feature based on the security properties of the operations used to handle this Feature throughout the system. Furthermore, it is distinguished between a phase 1-specific trust level (TL_{ς}^{P1}) and a phase 2-specific trust level (TL_{ς}^{P2}). A TL_{ς}^{P1} expresses the trustworthiness of the Feature for the first three operations, thus for the Feature residing on the Provider. The TL_{ς}^{P2} enhances this by adding the last transmission operation to the trust calculation. As the current IF-MAP version provides no capability of expressing such a trust level, a method for doing so must be introduced. It is therefore necessary to extend IF-MAP, allowing to incorporate trust into the metadata environment. IF-MAP creates a graph out of published metadata reflecting a network's state. To do this, IF-MAP clients publish arbitrary information in a specification compliant way. This metadata is collected by the MAPS and is stored within a graph. Besides adding metadata about certain network relevant properties and other things, this concept allows also to add metadata representing the trust level of other published metadata. In order to realise this, IF-MAP needs to be extended with a special metadata type. This type, called Trust Token (TT) is defined in a XML Schema as trustTokenType metadata in the following way.

```
<xsd:complexType name="trustTokenType">
  <xsd:sequence>
  <xsd:element name="trust-level" type="xsd:integer" />
  <xsd:element name="mapc-id" type="xsd:string" />
  <xsd:element name="spr-process-sender" type="spRecordType" />
  <xsd:element name="spr-transmit-sender-provider" type="spRecordType" />
  <xsd:element name="spr-process-provider" type="spRecordType" />
  <xsd:element name="spr-transmit-provider-receiver" type="spRecordType" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="spRecordType">
  <xsd:sequence>
    <xsd:element name="security-level" type="xsd:integer" minOccurs="1"
        maxOccurs="1" />
  <xsd:element name="security-property" minOccurs="0" maxOccurs="unbounded"
       type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
```

Listing 5.9: TrustToken XML schema.

It consist of all necessary elements to map the results of the trust calculation $\{SPR, \{\omega\} \xrightarrow{rF} SL\}_{operation} \xrightarrow{tF} TL_\zeta$ introduced in this thesis, also including the intermediary results and steps. It consist of the following parts.

- **trust-level:** This element encapsulates the overall calculated trust level. It can express either a TL_ζ^{P1} or a TL_ζ^{P2}, according to the generation of the Trust Token itself.

- **mapc-id:** A relationship between the MAP Client and the published metadata the Trust Token is valid for is established through the use of this element. In detail, the appropriate publisher id of the MAPC is stored here. In case of a delete type operation, the id of the deleting client is used for storage.
- **spr-operation:** Each of these four elements represent the security property record generated for a particular operation. Based on the entries made in these fields, the RSPR can be obtained. As the syntax of the four elements is the same, all of them rely on the spRecordType which encapsulates each of the security properties assigned and the calculated overall security level for this particular operation. Given that this record is entirely filled with values, it can be considered as RSPR.

SPRM and SPM

The SPRM is responsible for providing the appropriate security properties along with their ratings. The SPM provides a way of holding these properties and the ratings in a dynamic fashion. Although the conceptual part defines the SPRM as solely responsible for managing the properties and everything connected to them, the implementation provides also a logical SPM component which handles the actual SPM policy. Given that, there are two interfaces: the SPM interface for updating ratings and the SPRM interface to assign the actual property to an operation. The SPRM uses the SPM internally to retrieve the appropriate rating for a property assigned. The actual task of measuring an operation to determine which property is valid for this operation, is outsourced to the MAP Server. Due to this, there are some constraints for the properties that are usable within this implementation. The particular definition of the properties and the used function to calculate the security level for an operation is shown later in this section.

Trust Token Manager

The Trust Token Manager is responsible for building up the final phase 1 Trust Token (P1TT) as well as the final phase 2 Trust Token (P2TT). To calculate them, it uses the preparation done on the SPRM's side. In detail, the security levels are used to form the Trust Tokens which are then provided through the TTMgr's interface.

The data model provided within the conceptual part of this thesis for storing the Trust-specific data (i.e. the TrustLog data structure) is applied to allow to store the P1TT as well as the P2TT. Furthermore, the P1TT is directly stored within the MAPS graph structure building up the provided data model. The P2TT is provided using enhanced communicational means. Furthermore, the function to calculate the TTs is also maintained and used by the TTMgr. This is necessary as the TTMgr is responsible for the calculation task, thus it is the only component which can use this function. As the P2TT is eventually communicated to the appropriate MAPC, the TTMgr is also responsible for providing it in a way it can be used for this communication task.

Based up on these components and their provided interfaces, the data model of IF-MAP can be extended with trust.

5.4.2 Extended Data Model

The P1TT needs to be calculated and stored within the MAPS graph. It is used to calculate the P2TT if a client requests the appropriate metadata. To store a particular Trust Token within the map graph, the trustTokenType introduced above is used. Based on this, the actual structure used to represent the TT within the MAP graph can be developed. This structure is called Trust-Token-Metadata (TTM) and specified in the following form [166].

```
<xsd:element name="trust-token-metadata" type="trustTokenMetadataType" />

<xsd:complexType name="trustTokenMetadataType">
  <xsd:complexContent>
    <xsd:extension base="trustTokenType">
      <xsd:attributeGroup ref="ifmap:multiValueMetadataAttributes" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 5.10: Trust-Token-Metadata XML schema.

To establish the relationship between the particular metadata stored in the graph and the TTM, a self defined operational attribute is used. This attribute enhances metadata by using the anyAttribute extension mechanism. It is called Trust-Token-ID (TTID) and holds a unique string referencing the particular TTM, which is using the same id. Figure 5.5 depicts the structure which results by using this approach. The MAP Server is responsible for generating the appropriate TTM. Due to this, there are some constraints for

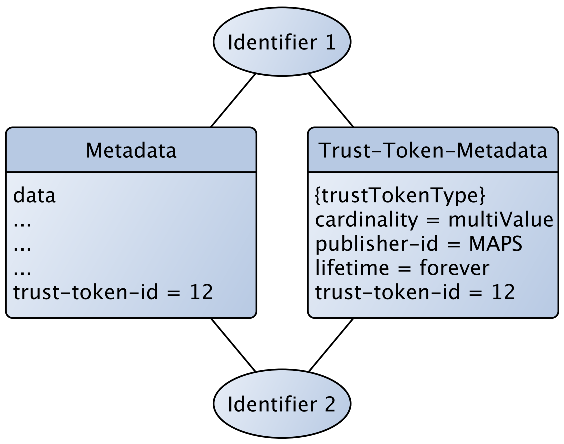


Figure 5.5: Ttrust Token Metadata as defined in [166].

setting the operational attributes of the TTM. First of all, the publisher-id must be set to the MAPS itself. This is necessary to allow all parties, in particular MAPCs, to recognise that the TTM is managed by the MAPS. As written in [166], this id should be set when starting the MAPS and must not be changed at any time the MAPS is running. The lifetime attribute is set to forever which is only done to mark the TTM for readability. As the TTM is bound to the MAPS itself, it implicitly has a lifetime of forever as there is no real session attached to it. To allow for more than one TTM in case of multiple metadata between identifiers, the cardinality must be set to multiValue.

Feature Base Snapshots

The introduced concept demands the use of a history providing function within the trust extension. This history is used to keep track of value changes as well as of trust changes. The implementation does not place this history on the MAPS, but uses the abilities the CADS system already provides. Detailed in [2], the Correlation Engine responsible for decision making uses a so-called Feature Base to internally manage the Features received from the MAP Server. This Feature Base already possesses the means to build up a history for Feature values. Due to their sequential storage of all Feature updates it can be used to provide the history function. In detail, the sequential storage is not based on overwriting an existing Feature but on using a linked list where the newest Feature is simply placed in front. The history function is provided by utilising the enhanced data model, which encapsulates the Feature into the TrustLog data model. The Feature Base is extended to use the TrustLog instead of the Feature, thus also including the trust updates in its storage model. The detailed data model used for this implementation is shown later in this chapter.

5.4.3 Extended Communication Model

In order to provide the P2TT for a particular MAPC, the communication model of IF-MAP has to be enhanced. The first operation which is performed consists mainly of the procedure to establish a valid session between the MAPC and the MAPS. As there is no particular metadata exchanged, no enhancements are necessary. Besides this, the operation itself might be used in order to assign security properties, similar to the type of the authentication used. Operations which terminate the communication between the MAPS and the MAPC, like an intended endSession or an unintended communication interruption, don't need to be enhanced either. The MAPS has only to take care about the correct deletion of the appropriate metadata, which belongs to the client the session has ended for. Furthermore, TT-specific data may be deleted too if there is no other client linked to it. As written above, keeping track of such changes, in particular of the deletion, is done on the Correlation Engine's side by utilising the Feature Base. Due to this, the MAPS enhancement can be kept rather simple.

In particular, operations which need to be enhanced are

- purgePublisher,
- publish-update,
- publish-delete,
- search and poll.

Details to each operation are given in the following.

The first operation which influences the TT calculation is purgePublisher. It is used by a client to indicate a cleanup of its metadata which can be used as a fresh staring point for republishing. Due to this cleanup, all associated TTs have to be recalculated if a purgePublisher is triggered. If old data is not reconstructed by a following publish of the same MAP Client, the associated TTs have to be deleted after some time, in particular after they have been provided to all subscribers.

If a MAP Client performs a publish-update operation, all Trust Tokens (phase 1) have to be updated as well. In particular, the stored TTM within the MAP graph is updated with the newly calculated result. After this process is done, the Correlation Engine is notified about this change to allow a complete picture of the appropriate trust and value changes. Furthermore, active subscriptions will be handled after the complete trust calculation is finished. This is necessary to avoid states where a TTM is outdated. Finally, the MAPC cannot publish TT itself, the server must prevent this and reply with an appropriate error.

In case of a publish-delete from a MAP Client, the metadata and their associated TTs need to be deleted from the MAP graph. Given other subscriptions to this metadata, the subscribers need to be informed about the deletion. In particular, the TT has to be recalculated in order to represent the trustworthiness of the delete operation itself. Furthermore, the Feature Base of the Correlation Engine receives this delete operation and can keep track of the deleted metadata. Therefore, it stores the last valid version of the metadata, along with the delete operation's trustworthiness.

Doing a search on the MAP Server, a MAP Client receives not only the searchResult itself but also the appropriate P2TT attached to this search. That is, the MAP Server calculates the P2TT based on the P1TT as soon as it receives the request for the search operation as it is able to determine the appropriate security properties for this client (i.e. by using the SPRM which can evaluate the communication between the MAPS and the requesting MAPC). Equally to the search operation, a pollResult is also attached with the appropriate P2TT valid for the polling client.

5.4.4 MAPS Security

Based on the enhancements for the IF-MAP protocol made above, the security situation of the MAP Server has to be considered again. The discussion of the security of the MAPS consists of two parts: First, the communication part including the publish and subscribe operations. And second, the security of the MAPS itself. As the communication is inherently secured by using appropriate methods (e.g. TLS) defined within the MAP specification, there is no need to investigate this part again. Contrary to this, the security of the MAPS itself need to be treated again as the specification does only give minor hints on how to treat this issue. This is due to the fact, that the MAPS is not intended to hold the actual state of the network with real data. The specification defines, that it holds a state which may reflect the networks state. Furthermore, the MAPS is not intended to be used as a source for decision making but solely as an information database. As the concept presented above moves the MAPS into an active role, these drawbacks need to be addressed explicitly.

To use the MAPS for the described kind of decision making source, it has to meet the following requirements:

- The database the MAPS holds must be changeable in a defined way only, i.e. by using data received via IF-MAP operations. Unauthorised access to this database must be prevented.

- As the MAPS itself creates the Trust information and publishes those into the MAP graph, it is important that this process runs in a defined way. A possible compromising of this process results in invalid TTs, thus breaking the whole concept. As said above, a MAPC must not be allowed to publish trust related information itself.

One approach which provides a reasonable security is the use of a trusted boot as described by Trusted Computing. To perform a trusted boot, the first step is to define an integrity state of the platform the MAPS runs on. This state should be clear and trustworthy, thus the system should not be compromised in any form. Additional features may be defined, such as the systems patch level or software packets. If this state is defined, a Trusted Platform the MAPS system has to be to use trusted boot, can be configured in a way, that it only boots up completely if the current system state matches the predefined one. This means, the MAPS can only start if its integrity state is as expected. This does in particular not help against zero day issues, such as software problems, but it minimises the risk of a compromise. Another variant of this approach can be used to derive trust for MAPCs. A detailed description of this approach can be found in 7.4.

Based on this concept, the actual implementation is described in the next section, including the used software.

5.5 Implementation based on IF-MAP

The implementation of the enhanced MAP Server is based on the irond server developed and maintained by [168]. It is a TCG certified IF-MAP 2.0 capable MAPS written in Java. Furthermore, the snapshot-based enhancements provided by extending the Correlation Engine's data model are implemented by using the CADS reference implementation. This reference implementation consist mainly of the Correlation Engine, which was also developed by the Trust@FHH research group. The particular implementation of it is called irondetect [168] and is also written in Java. Enhancing both subsystems allows to provide a reference implementation for the TCADS system. Due to both systems being written in Java, all implementation parts of this work are also written in Java.

Going back to the MAP Server, the trust extending components are implemented in a stand alone manner first. In order to actually use them within the irond, a interface connecting both parts is developed as next. The trust extending components are given by the already introduced TrustService.

5.5.1 TrustService

Internally, the TrustService is realised by three different classes forming the domain model. This domain model is depicted in figure 5.6. As the figure shows, there is a class repre-

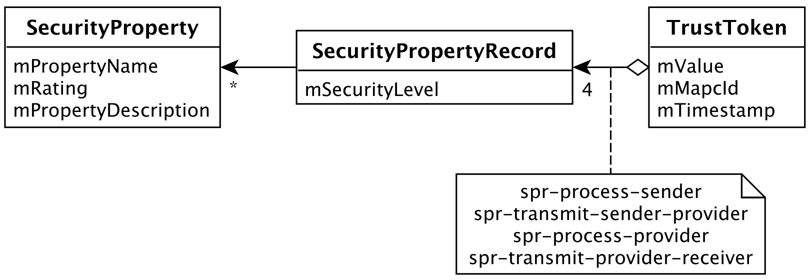


Figure 5.6: TrustService domain model [166].

senting the security property, the security property record and the final Trust Token. The SecurityProperty class simply encapsulates the name of the property represented by this element, the current rating retrieved by the SPRM and a description giving element. The SecurityPropertyRecord class references n actual SPs, which are used to calculate the security level of the record. The security level is stored within this class. Four of these SecurityPropertyRecord objects are referenced by the TrustToken class, due to the four defined operations. Furthermore, the TrustToken class holds the particular trust level value, the id of the MAPC and a timestamp indicating the time of calculation of the particular trust level. Given theses domain classes, the actual service can be described. This is done in the following.

As described within [166], aim of the TrustService-based implementation is to decouple the trust extension from that particular MAP Server. The actual measurement of the security properties is not part of the service itself and must be provided by the SPRM-based component. The overall class structure is depicted in figure 5.7.

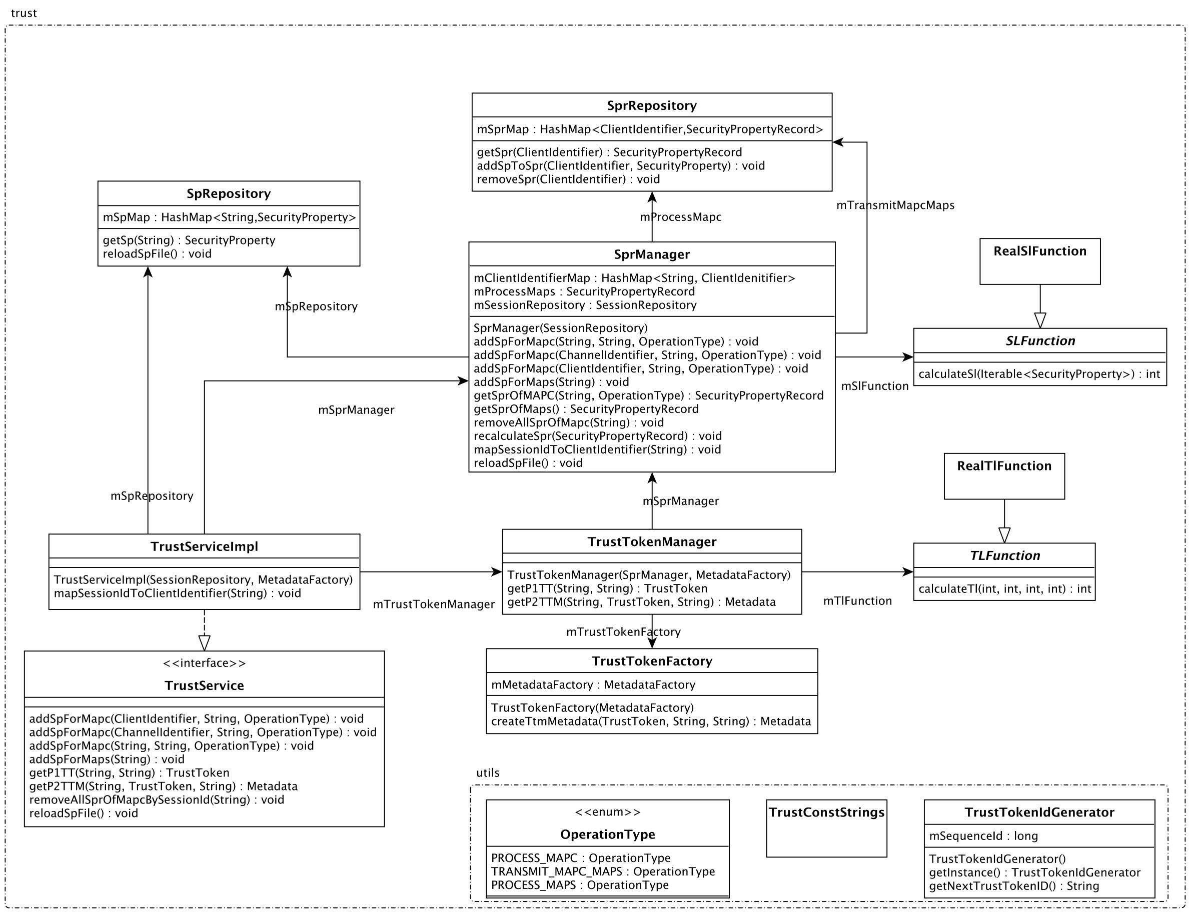

trust
SprRepository
mSprMap : HashMap<ClientIdentifier,SecurityPropertyRecord>
getSpr(ClientIdentifier) : SecurityPropertyRecord
addSpToSpr(ClientIdentifier, SecurityProperty) : void
removeSpr(ClientIdentifier) : void
SpRepository
mSpMap : HashMap<String,SecurityProperty>
getSp(String) : SecurityProperty
reloadSpFile() : void
mTransmitMapcMaps
mProcessMapc
SprManager
mClientIdentifierMap : HashMap<String, ClientIdenitifier>
mProcessMaps : SecurityPropertyRecord
mSessionRepository : SessionRepository
SprManager(SessionRepository)
addSpForMapc(String, String, OperationType) : void
addSpForMapc(ChannelIdentifier, String, OperationType) : void
addSpForMapc(ClientIdentifier, String, OperationType) : void
addSpForMaps(String) : void
getSprOfMAPC(String, OperationType) : SecurityPropertyRecord
getSprOfMaps() : SecurityPropertyRecord
removeAllSprOfMapc(String) : void
recalculateSpr(SecurityPropertyRecord) : void
mapSessionIdToClientIdentifier(String) : void
reloadSpFile() : void
mSpRepository
mSprManager
RealSlFunction
SLFunction
calculateSl(Iterable<SecurityProperty>) : int
mSlFunction
RealTlFunction
TLFunction
calculateTl(int, int, int, int) : int
mTlFunction
mSpRepository
mSprManager
TrustServiceImpl
TrustServiceImpl(SessionRepository, MetadataFactory)
mapSessionIdToClientIdentifier(String) : void
mTrustTokenManager
TrustTokenManager
TrustTokenManager(SprManager, MetadataFactory)
getP1TT(String, String) : TrustToken
getP2TTM(String, TrustToken, String) : Metadata
mTrustTokenFactory
<<interface>>
TrustService
addSpForMapc(ClientIdentifier, String, OperationType) : void
addSpForMapc(ChannelIdentifier, String, OperationType) : void
addSpForMapc(String, String, OperationType) : void
addSpForMaps(String) : void
getP1TT(String, String) : TrustToken
getP2TTM(String, TrustToken, String) : Metadata
removeAllSprOfMapcBySessionId(String) : void
reloadSpFile() : void
TrustTokenFactory
mMetadataFactory : MetadataFactory
TrustTokenFactory(MetadataFactory)
createTtmMetadata(TrustToken, String, String) : Metadata
utils
<<enum>>
OperationType
PROCESS_MAPC : OperationType
TRANSMIT_MAPC_MAPS : OperationType
PROCESS_MAPS : OperationType
TrustConstStrings
TrustTokenIdGenerator
mSequenceId : long
TrustTokenIdGenerator()
getInstance() : TrustTokenIdGenerator
getNextTrustTokenID() : String

Figure 5.7: TrustService class model [166].

It consist of several classes providing the necessary methods and attributes for the TrustService implementation. The main interface is provided by the class TrustService itself.

TrustService Interface

This interface allows to store measured security properties and to generate (i.e. to calculate) the P1TT as well as the P2TT. It provides the following methods in order to realise this.

addSpForMapc() This method allows to store a measured *SP*. It is usually triggered by the irond MAP Server and allows to identify the SP by a particular key. This key may be given as one of three types: the `ClientIdentifier`, as a `ChannelIdentifier` or as `SessionID`. The identifier-based keys allow to use either a MAP Client's username or the ip/port combination as method for identifying a certain property.

addSpForMaps() To store a property measured for the MAP Server itself, i.e. the irond server, this method is provided. As this property can only be set for exactly one server, where the TrustService is running on, no key must be given.

getP1TT() Provides a P1 TrustToken by the use of a Session- or Publisher-ID to the MAP Server.

getP2TTM() Allows the MAP Server to retrieve the overall P2 Trust Token. A Session-ID, the already calculated P1TT as well as an id for the metadata is required to do this. The method returns the particular P2TT-specific metadata.

removeAllSprOfMapc() Requires a Session-ID in order to delete all measured security properties of a particular MAPC.

reloadSpFile() To allow for dynamic ratings, the policy containing these ratings (i.e. the SPM's implementation) can be reloaded at every time. This is done by triggering this method.

TrustServiceImpl Class

The class provides an actual implementation of the TrustService interface. In order to fulfil its task, it makes use of the `SprManager`, `SpRepository` and `TrustTokenManager`.

The TrustServiceImpl is in particular responsible for delegating the appropriate method-based requests to one of these classes. The maintenance of the particular security property records is done by the `SprManager`. In detail, this class is responsible for receiving the particular SP from the SPRM and the calculation of the corresponding security level. The `SpRepository`, which can be seen as logical part of the SPRM provides the particular access to the assigned security properties. Given both classes, the `TrustTokenMananger` can create the P1TT as well as the P2TT. In detail, the assigned trust level for both Trust Tokens is being calculated by using the given SPRs and their connected SPs. The irond implementation allows to map the accepted keys (that is the `ClientIdentifier`, `ChannelIdentifier` and `Session-ID`) by the use of its `SessionRepository` class. Internally, irond uses the `ClientIdentifier` to map all known MAP Clients to sessions and channels. Due to this, the two other key types are mapped back to a `ClientIdentifier` key by using the database of the irond. Given this, the measured security properties are stored as references of a certain `ClientIdentifier`. While the mapping outsourced to the irond itself provides a straightforward access, there is one problem: if an endSession operation is triggered, the irond server deletes the appropriate mapping instantly. This results in a situation, where the TrustService would be unable to resolve the key using the irond's internal mapping database. In order to provide a solution to this problem, the TrustService holds another `ClientIdentifier`-based mapping on its own. The particular TrustServiceImpl allows therefore to use the method `mapSessionIdToClientIdentifier()`. The SprManager is able to trigger the key resolving when performing `removeAllSprOfMapc()`.

SpRepository

This class provides the interface to the security properties, and in particular to the assigned ratings. The locally stored policy, representing the SPM to some extent, is loaded instantly when starting up the implementation. A HashMap-based runtime storage is used to cache the entries of the policy: the SP as key and the rating as its value. To allow the ratings to be changed in a dynamic fashion, the `reloadSpFile()` method is provided. This method triggers a reloading and re-caching of the stored policy, thus a reloading of the appropriate properties and their ratings. In order to update a rating, the method must be triggered actively. That is, the current implementation does not monitor the SPM on its own and demands an interaction in order to begin the reloading process.

SprManager

The assignment of the particular security properties to the operations, thus to the SPRs are managed by this class. Due to each MAP Client possessing two potential SP sets, one for the process operation on the client itself and one for the transmission between the client and the server, the SprManager holds two references to the SprRepository class. This references encapsulate the particular security property record per operation for a client. Contrary to that, only one direct reference to the SecurityPropertyRecord instance is required for the MAP Server itself. This is due to the already mentioned fact of the MAP Server only being one instance and only holding one set of operational properties. To resolve a particular `propertyName`, the SpRepository is used in order to provide the actual security property. The `propertyName` itself is provided via the already described `addSpForMapc()` and `addSpForMaps()` methods. Using the SlFunction class, the actual security level can be calculated for a security property record. In case of carrying out one of the get methods (`getSprOfMapc()`, `getSprOfMaps()`), the SL is being calculated implicitly. The particular function used of calculating the SL is not completely defined but being provided by the means of a strategy patterns-based class. This class, RealSl-Function, inherits from the SlFunction class and implements the specific method used for the calculation.

TrustTokenManager

Given the security property records by using the Session-ID of the particular MAP Client encapsulated within the SprManager, the TrustTokenManager is able to provide the P1TT as well as the P2TT. To calculate the particular trust level included within the TT, the TlFunction class is used. Equally to the SlFunction class and the used strategy pattern, the actual method used to calculate the trust level is given within the RealTlFunction class. Providing the P2TT, the TrustTokenFactory is used. This class uses another factory-based class: the MetadataFactory which is able to create an instance of the Metadata class. This instances is used by the irond to represent metadata within its graph structure, thus providing the methods to create search and pollResults answering P2TT requests. Besides that, the main responsibility of this class is to answer requests for either a P1TT or a P2TT.

getP1TT() The overall procedure which is carried out by triggering this method is depicted in figure 5.8. The first step in order to receive a P1TT is to access the assigned SPRs. As the P1TT represents a phase 1 trust level, three SPRs are needed for calculating the TL: the SPR of the client's process operation, the SPR for the transmission between the client and the server as well as the process SPR of the MAPS. Having retrieved these three SPRs, the TrustTokenManager is able to instantiate the particular TrustToken. This TrustToken represents a p1tt which is being communicated to the requesting client. Given the view from the TrustService, a client is rendered by a party requesting this p1tt. In case of the implementation described here, the irond represents such a client.

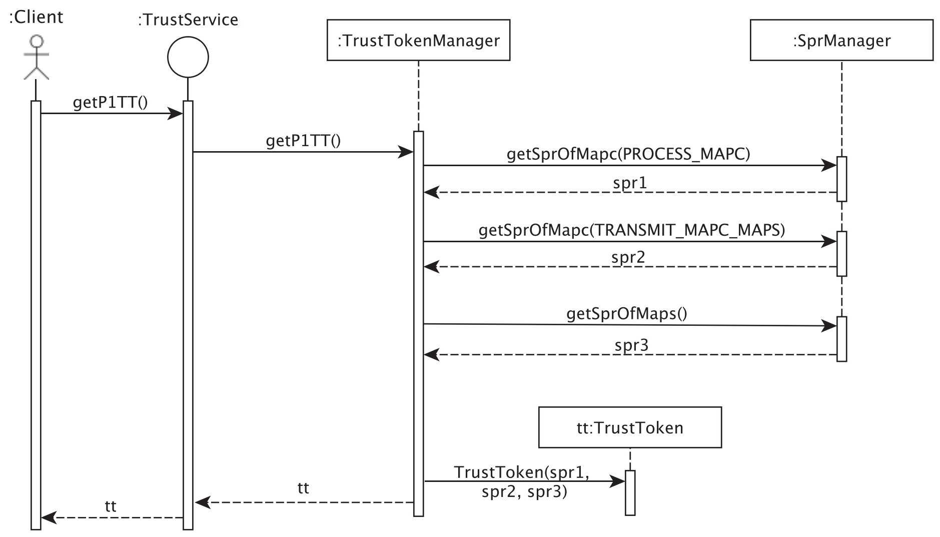


Figure 5.8: P1TT request [166].

getP2TTM() In order to provide the P2TT, this method is used by the irond (i.e a particular component of the irond). The P1TT which was previously instantiated is used for this task. This is due to the P1TT holding the particular parts which form the final trust level (phase 2 trust level), in detail the first three security levels. As there might be changes to these security levels, a recalculation is triggered as first. The task of recalculation is done by the SprManager class. Using this recalculated values, the p2tt can be created. As there is one SPR missing, the next step is formed by retrieving the SPR for the last operation: the final transmit between the server and the receiving client. Given this SPR, the appropriate security level for this set of security properties can be calculated. By finishing this, there are four security levels available which are finally used to calculate the overall (i.e. phase 2) trust level. This trust level is encapsulated as metadata by the use of the `createTtmMetadata()`and provided to the requesting client (as already stated, the irond server). Theses steps are depicted in figure 5.9.

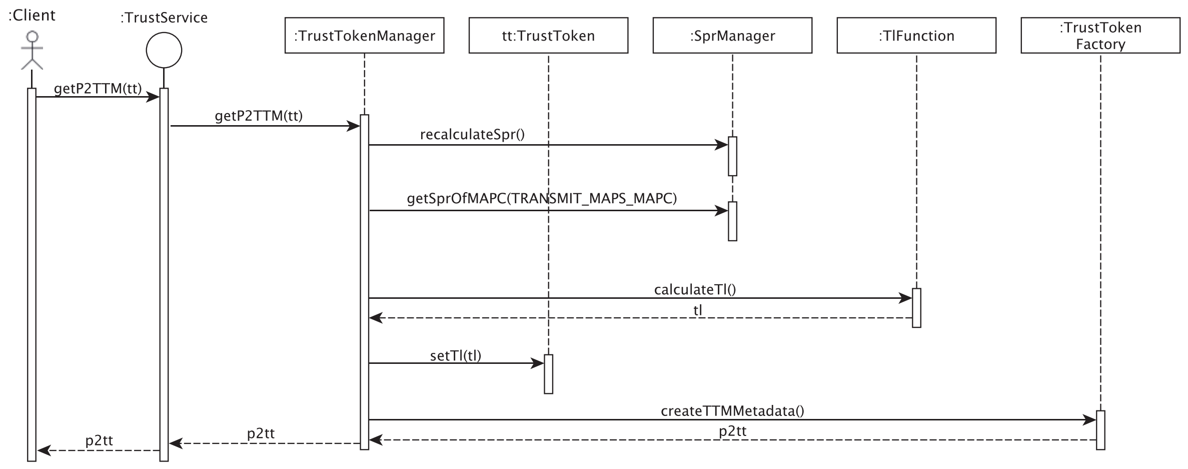


Figure 5.9: P2TT request [166].

Besides these parts, which bundle the main functionality of the TrustService there are some more classes which are used to realise the required auxiliary functions.

TrustTokenIdGenerator

This class takes responsibility for providing unique and non repeating ids for establishing the link between the TrustTokens and the metadata they belong to. To fulfil the requirements of uniqueness, the instantiation process is realised by utilising a Singleton pattern. That is, there must be only one instance existing at the same time. This instance takes care of managing the ids, so that no id is used twice. Internally, a long type-based attribute is used for this task. This attribute is simply increased each time an id is requested.

OperationType

This enumeration maps the appropriate operations to constant-based string values. The mapping is used to assign the security properties to the operations. Because of this there is a mapping for the process operation on the MAP Client (`PROCESS_MAPC`), a mapping for the process operation on the MAP Server (`PROCESS_MAPS`) and a mapping for the transmission between the MAP Client and the MAP Server (`TRANSMIT_MAPC_MAPS`). Due to the implementation creating the P2TT only on request, there is no particular mapping for the final operation.

TrustConstStrings

This class holds all other constant strings used within the implementation. Furthermore, external components accessing parts of the implementation can use it in order to operate on the same attributes.

To use this TrustService in combination with the irond implementation, a strategy for implementing it, based on the particular irond components, is required. This combination is explained in the following section.

5.5.2 Combining MAPS and TrustService

Details of the irond implementation are given in [168] including the architectural layers of the server. Summarising this, there are two of these layers: the communication layer as well as the data model. The communication layer is responsible for managing the authentication of MAP Clients, the handling of the sessions and the authorisation of the MAP Clients. These tasks are not limited to one client at a time but are implemented in a highly parallel manner. The particular storage of the graph and the process required to realise this handling is encapsulated within the data layer. The interface between those two layers is given by accessing data model specific methods through a call from the communication layer. Based on this, the interface between the TrustService and the remaining irond components must be realised on both layers (cf. [166]). The overall model the implementation is based on, is depicted in figure 5.10. As it easy to see, the TrustService is being called from both layers of the irond and therefore acts passively.

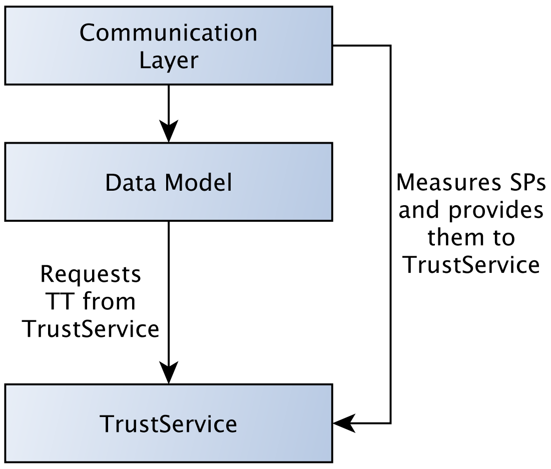

Communication Layer
Data Model
Requests TT from TrustService
Measures SPs and provides them to TrustService
TrustService

Figure 5.10: TrustService extended layer model of irond [166].

The TrustService is being initialised within the irond's initialisation process. That is, at the same time the irond server starts its internal components, the TrustService is instantiated. Besides that, a customised signal handler was added to the irond. This signal handler waits for the *SIGHUP* signal which can be given to the irond from outside. Receiving this signal, the irond calls the already mentioned method to reload the policy file holding the security properties as well as their ratings. This mechanism is used to dynamically update the ratings at runtime of the server. As already stated, the policy file itself is not being monitored actively, thus the only method to instruct an update is to use the appropriate signal.

Communication Layer Enhancements

To provide the appropriate security properties, the communication layer of the irond needs to be enhanced. The process of measuring the SPs is divided into two parts: the measurement of the MAP Client's SP and the measurement of the server's SP. Figure 5.11 depicts the enhanced communication layer, including the TrustService interface. To measure the appropriate security properties which can be derived from the connection between the server and the client, the `ChannelAuth` class as well as the `ChannelAcceptor` class are able to call methods from the TrustService. Both classes determine the type of the connection which is either a certificate-based or a simple authentication-based connection. Using this

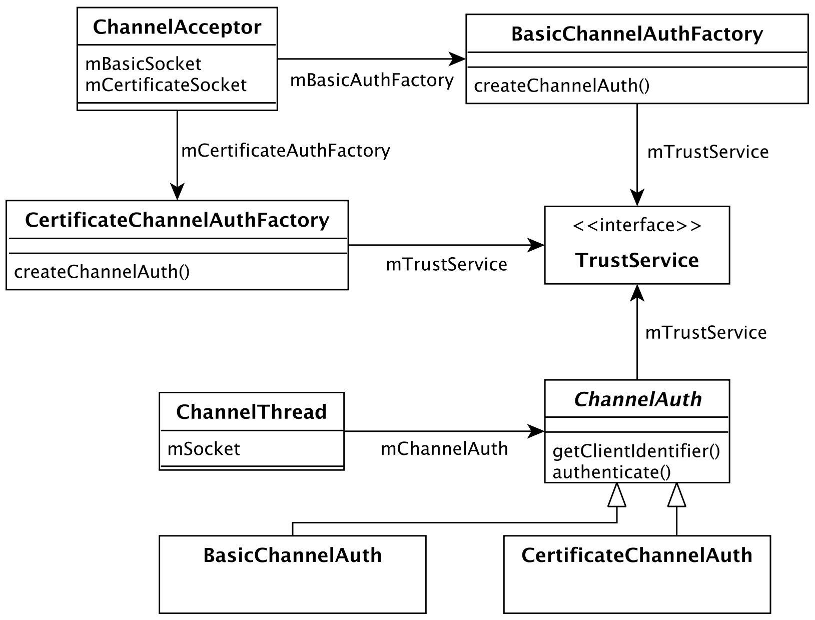


Figure 5.11: TrustService extended communication layer [166].

information, they are able to assign appropriate security properties which represent these different connection types and bind them to the correct `ClientIdentifier`. The partic-

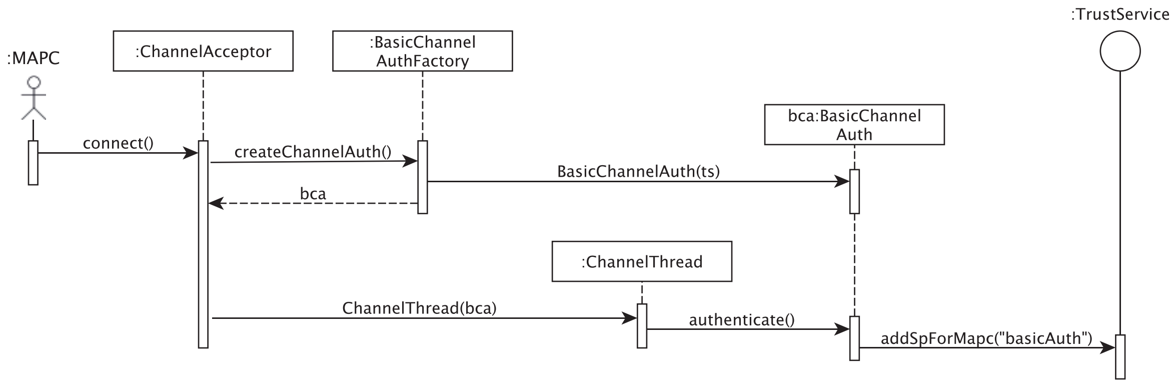


Figure 5.12: Basic auth SP measurement [166].

ular steps when measuring the communication channel are depicted in figure 5.12. After the `ChannelAcceptor` received a connection request for a simple authentication-based connection, it instantiates a `BasicChannelAuth` object (`bca`) including the TrustService reference. Following this, the `ChannelThread` authenticates the clients using the provided `bca` instance. This is done by calling the `authenticate()` method. After this step, the derived security properties are assigned by the use of the TrustService interface.

The derivation of appropriate security properties for the MAP Server itself, thus the SP for the MAPS's process operation is rather straightforward. In detail, the only effective way of providing a measurement for the server itself is the use of a TPM sealed certificate. As stated above, this certificate allows to verify that the server was booted up into a pre defined state. Using this certificate and some signing methods, a MAP Client is able to verify this state of the MAPS. Given that, the SP for the process operation on the MAPS are derived using this method.

Data Model Enhancements

In order to store the P1TT metadata structure within the irond's graph, certain enhancements to the data model are necessary. As already written, the reference between the metadata and their Trust Token is established through the TrustTokenID (TTID). Both, the metadata as well as the P1TT use the same TTID to indicate that the P1TT belongs to exactly this metadata. As depicted in figure 5.13, the connection between the P1TT

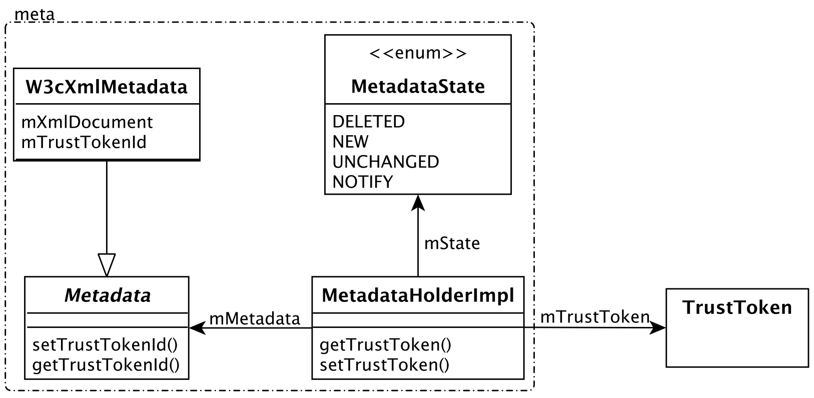


Figure 5.13: Enhanced data model [166].

and the metadata itself is being managed by allowing the class responsible for handling the metadata to get or set an appropriate TTID. In detail, metadata managed by the irond is being encapsulated within a `W3cXmlMetadata` object. This object holds the XML-based representation of IF-MAP complying metadata. To link this metadata to a P1TT the class is enhanced with a TTID member holding the appropriate TTID. To actually store metadata within the graph structure, the class `MetadataHolderImpl` is being used. This class is enhanced by Trust Token-specific methods and a member referencing the appropriate Trust Token. That is, the class not only provides the metadata storing means but also binds the appropriate Trust Token to this metadata in the data model. Furthermore, there are four classes, `ClientService`, `SubscriptionService`, `SearchService` and `PublishService` which are responsible for the operation-based handling. Using the enhancements explained above, the overall procedures when performing an IF-MAP based operation can be explained following the responsibilities of these classes.

ClientService

This class is responsible for the overall session handling between the MAP Client and the server. It therefore provides three methods, which all have been enhanced. Details are given in the following.

newSession() This method is responsible for establishing a new session for a requesting client. It is directly called when a MAP Client carries out the newSession IF-MAP operation. Using the trust extension, it is also responsible for initialising the mapping between the Session-ID and the ClientIdentifier used.

endSession() Calling the endSession IF-MAP operation on the MAPS by a client triggers this method. It is responsible for cleaning up the session itself as well as attached metadata. In addition to that, all assigned security properties of the ending MAP Client are removed by calling the appropriate method of the trust extension.

purgePublisher() Responsible for performing the purgePublisher IF-MAP operation, this method takes care of a recalculation of the Trust Tokens when called.

PublishService

All types of the IF-MAP publish operation are handled by this class. In detail, the class consist of three particular private methods, which handle the update, delete and notify publish operations. The only one of these three types which is rather different in terms of trust-specific handling is the delete method. It simply calls the TrustService interface to obtain a particular P1TT for the client deleting the metadata. This P1TT is attached to the metadata which is to be deleted. As for the publish update and notify operations, the first step is also to obtain a particular P1TT. After this, it is checked if any of the metadata which is to be published contains a Trust Token already. In detail, the namespaces of the metadata is checked against the trust-specific names and if a match is found, which indicates that a client tries to publish a Trust Token, the process is stopped with an AccessDeniedError. If there is no Trust Token included, the metadata is extended with the particular connection to the previously created Trust Token. In order to store the resulting metadata elements within the MAP graph, a TTID is generated and inserted into the metadata itself.

SearchService

IF-MAP based search operations are handled by this class. As a searchResult is generated by this class which is given back to the requesting client, some changes are necessary. When the searchResult is actually generated, all P1TTs for the entries stored within this result are retrieved. These P1TTs are then converted into P2TTs. This is done by using the MAP Clients identifier which requested the search- . Finally these P2TTs are added to the result itself and communicated to the requesting client.

SubscriptionService

In order to provide a pollResult to a client which has a particular subscription, this class is used. Furthermore, the initial searchResult is also provided by this class. Equally to the SearchService, this class takes P1TTs for each metadata within its result set and generates the appropriate P2TT for it as it is aware about the requesting client. This combined result is then provided to the client.

As explained above, all four services have been enhanced in order to provide trust-specific means. A detailed explanation which includes the particular single steps is given in [166].

5.5.3 Correlation Engine Enhancements

As already stated, the snapshot-specific enhancements allowing to build up a history for the Feature values as well as their trust level are done on the Correlation Engine's side. Due to the reason, that the CE provides an internal mechanism already, referred to as Feature Base, which stores all Feature-specific changes, only minor changes to the underlying Feature data model are required. These changes do not only allow to store trust related information but also to include the introduction of the trust level as a special context type.

Figure 5.14 depicts the enhanced Feature data model. As it is easy to see, the TrustLog as well as an enhanced Context Parameter is included. The TrustLog includes fields to

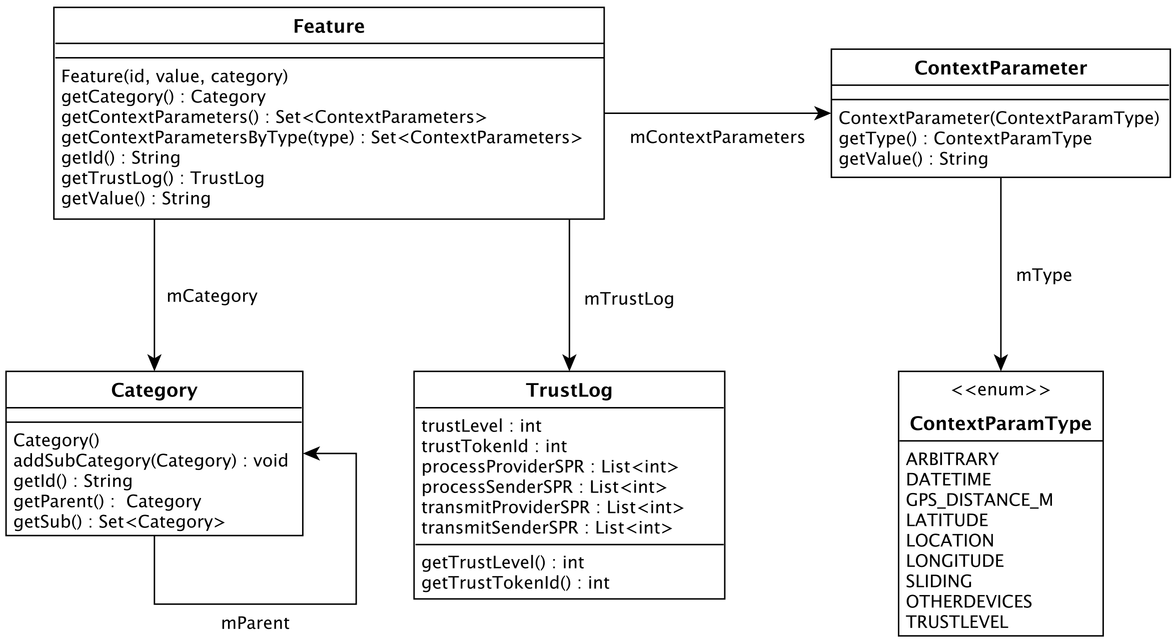


Figure 5.14: TrustLog enhanced Feature model.

store the appropriate ratings of the particular SPs within the security property records (i.e. the RPSRs) for each operation. Within this implementation, basic integer values are used for this purpose. This fine grained storage allows the Feature Base to keep track of every single change which appears on the side of the security properties and their ratings. Furthermore, the Trust Token Id is stored. When parsing the Feature out of the received answer from the MAP Server, the connected Trust Token is parsed as well. To allow for a reasoning between different Trust Tokens and their appropriate links to actual metadata, this TTID is used and thus stored. In addition to these rather fine grained trust-specific parts, the overall trust level for the Feature is stored as well. Although the expected trust level would be a phase 2-based version for the Correlation Engine, phase 1 is used here. This is due to the fact, that the more important changes are included in the phase 1 version and the communication channel between the MAPS and the CE should be considered secure within this implementation. However, the Correlation Engine is still able to operate on the phase two version by combining the appropriate stored SPR field on its own. The stored trust level provides a more convenient way of accessing the value.

Furthermore, the `ContextParamType` enumeration is extended by the TRUSTLEVEL type. This allows the Correlation Engine to use the trust level of a Feature as a context parameter within its policy. In detail, when constructing a Feature for its use, the TrustLog not only stores the trust-specific information but also enables to access the phase 1 trust level field to be accessed as special context parameter. This is done by simply adding it to the already existing context set of the Feature.

Given all that, the Feature Base is able to provide a history of all the changes made to these elements. Thus it stores the changes of the Feature's value along with the change of the trust level including particular SPR changes and the change of the trust level context type. Finally, as the enhanced MAPS provides deleting information for Features, it is also able to keep track of deleted Features and their corresponding values.

5.6 Evaluation

Given the implementation of the TCADS approach from the previous sections, this section provides an evaluation of not only the implementation but the overall concept. This is done in two steps: first the trust enabled MAP Server is evaluated in terms of performance and isolated test cases and second an overall test case including all components of the TCADS system is carried out. The case used for the second part is given by the ESUKOM project and provides a real world like application to the system. To operate on a defined basis, the used security properties are defined in the next section. These properties will be used throughout the whole evaluation.

5.6.1 Security Property Definition

In order to provide a basis for the following evaluation, the security properties that are used for determining the trust levels of the communicated Features are introduced including their initial ratings and functions for calculating security levels and trust levels.

There are six distinct properties defined for the evaluation scenario. They are explained in detail in the following with the appropriate ratings given to each property are explained afterwards.

Android-based smartphone This property expresses the processing of a Feature on an Android-based smartphone. That is, this property is assigned to each process operation which is carried out on such a smartphone. Due to the fact, that the scenario used for the evaluation only uses Android-based smartphones, this property allows to globally distinguish between smartphone-based Feature processing and other systems carrying out process operation for Features. More in detail, the assignment of this property is done by evaluating the basic authentication properties of the MAP-based communication. As the TCADS system relies on a special agent in order to collect smartphone-specific data, the authentication credentials of this client are known to the MAP Server. Using the SPRM component, these credentials are transformed into the appropriate property.

User Account Each time a MAP Client of the TCADS system is about to publish Features, it needs to perform an authentication to provide its identity against the MAP Server. This authentication and the derived identity of the MAP Client can be used to assign a security property to the process operation of this client. The identity of the particular MAP Client is derived by the identity of the user that is assigned to this MAP Client. As with the Android-based smartphone property, the identity is used to distinguish between different levels of trustworthiness of the client. This is used to distinguish between internal clients and external clients. Internal clients are Feature processing systems which are considered as more trustworthy as they are controlled by the domains administrator. External clients, like laptops which can be moved around, are not that trustworthy due to their usage profile and the lower level of control the administrator has. The internal clients are likely placed on infrastructure components like routers. Reflecting these different client types, the rating of the property differs according to each type.

Communication Channel Type This property allows to distinguish between cleartext and encrypted communication. The SPRM component residing on the MAP Server evaluates each communication channel which is established between a MAP Client and the MAP Server. If the channel type that is used does not provide communication security by the use of TLS, a property expressing this is assigned. As default, the evaluation provided here should not communicate without the use of TLS encrypted channels. Due to this, no property is assigned if the channel is encrypted but only if there is no encryption or the certificate used to establish the connection is expired or invalid. Thus this property indicates negative trust and the rating needs to express this. As the name indicates, this property is assigned to transmit operations, indifferently which particular type of transmit operation is used.

OpenVAS-based Vulnerability Level In order to provide an external measurement of a system's vulnerability, an OpenVAS [169] enabled platform is used. This system performs regularly scans of all other systems being part of the network including the connected smartphones. The results of these scans, which are also based on CVE vulnerabilities (cf. [170] for a CVE overview), are given back to the SPRM on the MAPS as input. A default property indicating the scan result is attached to each process operation of the client. Using the detailed result of the scan, the rating of this property is changed. That is, the more vulnerabilities found, the lower the rating becomes. Furthermore, the amount of the rating's de- and increase is connected to the type of vulnerability found. OpenVAS distinguishes between several severity levels. These levels are used for the rating changes as well. More details on this are given later in this section when the actual rating values are defined.

Attestation Level As explained in section 7.4, a special type of certificate can be used to provide a measurement of a system's state. This certificate is only accessible to the client if the state of the platform the client is placed on is as expected. The certificate is sealed to this particular state, thus it is encrypted using the measurement of the clients platform which are performed by a TPM. Storing these measurements within the TPM's PCRs an encryption key can be derived by the TPM and the PCR values. If the PCR values change due to a change of the system's state, the key can no longer be derived, resulting in the system being unable to access the certificate. Given that, as long as the MAP Client is able to present this certificate to the MAP Server, the client's platform is to be considered within the expected state. Using this, a security property indicating this can be assigned to the client's process operation. The rating of this property is usually indicating a high level of trustworthiness as the system's desired state is usually considered trustworthy. However, this property may also be used to determine untrustworthy states under certain circumstances.

Trust Level Trend This property is based on previously calculated trust level values for Features of a certain Feature Collector (i.e. MAP Client). It is used to characterise the process operation of that Feature Collector. Unlike the other properties described, it aggregates trust measurements to a higher level as it uses the already existing trust level as input value. In detail, not a single trust level value is used but the trend of this trust level. If this trend decreases lower than a particular value, the rating of this property is changed appropriately. Given that, this property amplifies the trend of the trust level itself. The calculation used for this property is done on the Correlation Engine's side as it is necessary to use the history of the trust level values for deriving a trend.

Correlation Engine Alerts The previous explained property already uses the Correlation Engine to assign a particular property to the process operation of a MAP Client. This can be used in a more versatile way by assigning a general security property which is solely controlled by the Correlation Engine. This method is provided by the use of this property and a rating set by the Correlation Engine. It allows to create a cascade between the MAP Server's trust level calculation and the Correlation Engine. In detail, if the Correlation Engine is able to determine unexpected behaviour of a MAP Client, it can lower the rating of this client's property on the MAP Server thus influencing the trust level for all Features published by this cline. This method allows for a fine grained gearing between the Correlation Engine and the MAP Server in terms of trust calculation and unleashes a high potential of functionality for the TCADS system.

In order to assign appropriate ratings to these properties, the particular values used need to be defined for the evaluation environment. Based on these definitions, the initial rating values and their meaning can be introduced.

Ratings

The ratings used within the evaluation are numerical-based values indicating the trustworthiness of a certain property. Furthermore, they are represented by signed integer values whereas negative values indicate rather untrustworthy values while positive numbers represent trustworthy values. Given that, the initial rating values can be assigned to the above defined properties that are used throughout the evaluation.

- The Android-based smartphone property is statically assigned a -5 as rating. This indicates the lower trustworthiness of Android-based devices. As written above, this property is used to tag Features received from a smartphone.
- As the User Account property also indicates which MAP Client's process operation was used to create the Feature, the initial rating depends on the type of client. If it is an infrastructure-based component which is under the administrator's control, a value of 5 is assigned. This indicates a higher level of trustworthiness which is derived from the direct control of that component. If it is another component, a neutral 0 is assigned. However, due to their usage profile, smartphones receive an additional negative rating.
- If the communication channel used is not as expected, the Communication Channel property is assigned indicating a problem with that channel. Due to this, a -10 is assigned to this property in a static manner. Due to the high negative value, if this property is assigned it has a rather high impact on the overall calculation.
- The OpenVAS property is rated according to the findings of the OpenVAS system. If there are no findings but only informational hints, a 5 is assigned. This value indicates a rather good shape of the system. If there are warning level findings, a -1 is assigned in order to indicate minor problems. OpenVAS can also scan for critical problems. If there is one of those problems found on the system, a -10 is assigned which indicates a rather severe problem on the platform.
- The attestation level property is assigned if a client is able to present the sealed certificate or if a client is unable to do so although it should. In the first case, where the client is able to represent the certificate which indicates that the client's platform is within the expected state, a 10 is assigned. This is due to the fact, that the state

of the system is well defined and considered to be very trustworthy. If it is unable to present the certificate, a -5 is assigned as the system may be untrustworthy.

- The rating used for the Trend property is not statically defined. If the trend increases or decreases, this rating may be changed accordingly in small steps. However, initially it is set to 0.
- The same situation like for the trend's rating applies to the Correlation Engine's alert properties. A static rating is not defined as the Correlation Engine decides on its own how big the impact factor to the trust calculation should be.

With these basic definitions of the used ratings, the functions which are utilised for calculating the security and the trust level can be defined in the following. Additionally, the interpretation of the calculated values is given.

Functions

The overall process of calculating a Feature's trust level is defined as $\{SPR, \{\omega\} \xrightarrow{rF} SL\}_{operation} \xrightarrow{tF} TL_{\varsigma}$. Due to this, there are two functions that need to be defined, the rF which combines a RSPR's rating into a security level and a tF which combines these security levels into a single trust level for the Feature. The basic rF function is therefore defined as mean over all ratings given, with n being the number of ratings stored within the RSPR.

$$x = \frac{\sum_{i=1}^{n} \omega_i}{n}$$

As the result must be within a range of $[-1, 1]$ in order to be conform to the presented approach and provide a scoreable value, the values of this calculation are clamped into an interval of $[-100, 100]$. That is, every value within this range is mapped into the appropriate $[-1, 1]$ range while values below and above this range are interpreted as -1 and 1 respectively. The combined rF is therefore defined as follows.

$$SL = \begin{cases} 1 & , x > 100 \\ x & , -100 \leq x \leq 100 \\ -1 & , x < -100 \end{cases}$$

To get an actual trust level for a Feature, the calculated SLs need to be combined by using the tF. As there are two types of trust level, a phase 1 type and a phase 2 type, there are also two subtypes of the tF defined here. This is necessary in order to be able to calculate usable TL_ζ^{P1} values. The tF_{Phase1} which is being used to calculate the trust level on the Feature Provider is based on adding the calculated SL values. The result of this addition is multiplied by a static factor α_{P1} which is set to $\frac{1}{3}$, thus clamping the final TL_ζ^{P1} into an interval of $[-1, 1]$ again.

$$TL_\zeta^{P1} = \alpha_{P1}(SL_{processSender} + SL_{transmitSender} + SL_{processProvider})$$

To calculate the TL_ζ^{P2}, an extended version of this tF_{Phase1} is used (i.e. tF_{Phase2}) which simply adds the fourth SL value to the addition. Using the previously calculated TL_ζ^{P1} and another scaling factor α_{P2}, which is set to $\frac{1}{4}$, it is defined as the following.

$$TL_\zeta^{P2} = \alpha_{P2}\left(\frac{TL_\zeta^{P1}}{\alpha_{P1}} + SL_{transmitProvider}\right)$$

The interpretation of both trust level types is the same due to the scaling factor mapping both values into the same interval. That is, the trust level of a Feature is interpreted as follows.

- A value of 1 is treated as completely trustworthy. This value is only possible if every single SL reaches the highest possible trustworthiness.
- If the value is between 0.5 and 1 ($0.5 \leq TL < 1$), the Feature is considered to be overall trustworthy. To achieve this kind of result, most of the SLs need to have high values without any critical finding.
- The range between -0.5 and 0.5 ($-0.5 < TL < 0.5$) is interpreted as neutral judgement of the Feature's trustworthiness. That is, the actual interpretation of the value depends on the particular case.
- If the value is lower than or equal to -0.5 ($TL \leq -0.5$), the Feature is considered untrustworthy to a high level. This value can only be reached if there are critical findings or most of the single SL values are very low.

- If the TL calculation returns a -1, the Feature is completely untrustworthy as this value is only reached if all of the SL values are set to minimum.

Based on this interpretation along with the defined security properties and the required functions to calculate the trust level, the evaluation can be carried out. This is done in two steps: the MAP Server is evaluated isolated within the first step while the overall TCADS system is used in the second step. The next section starts with the evaluation of the enhanced MAP Server.

5.6.2 Trusted MAP Server

The MAP Server is tested in two different ways. First a performance evaluation is carried out and second the actual Trust Token generation is tested. The performance evaluation allows to judge about the approach's usability within a practical scenario. More in detail, a Trust Token needs to be generated for every metadata stored within the MAP Server's graph. The impact of this additional elements as well as the required processing power to generate them is tested by evaluating the extended irond against a non extended version. Using the extended version again, the second part evaluates the correct generation of the Trust Tokens for a specific case.

Test environment

To operate on a well defined base for testing, a test environment is introduced here. It consist of the irond in both versions and some auxiliary components which provide required data. Figure 5.15 depicts the test environment and the used implementations. The irond MAP Server shown in this figure refers to both versions, the trust extended and the default implementation, as both versions are compared to each other. Besides the MAP Server itself, the following components and their actual implementations are used.

DHCP Server This server system is provided by the use of the ISC DHCP server implementation [171] in conjunction with the irondhcp MAP Client which operates as Feature Collector. The irondhcp client simply parses the appropriate lease file generated by the DHCP component and publishes these results as Features to the MAP Server.

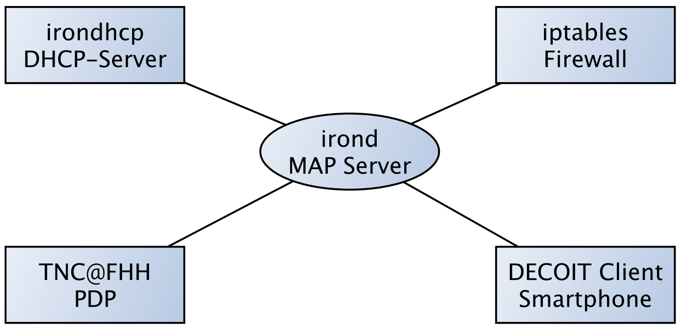


Figure 5.15: Test environment (cf. [166]).

Smartphone Two particular types of smartphones were used within this test environment: a Samsung Galaxy S III (see [172] for more details of the device) and a Samsung Galaxy Nexus device [173]. Both devices were extended by the DECOIT IF-MAP Client acting as a Feature Collector and providing the necessary data from the smartphones.

Policy Decision Point As the Feature tree created by Feature Collectors requires a root element represented by an appropriate access request of the smartphone, a Policy Decision Point publishing this access request is required. Within this test environment, the TNC@FHH TNC implementation [154] was used for this task. This implementation provides two means of building up this access request information. The first way is to use the actual server component, which is rather complex as this requires a full TNC setup. This was only done for testing purposes and not during the actual evaluation. During this phase, the second way which is given by a special IF-MAP enabled script client was used. This script client simulates the appropriate request and publishes it accordingly. This provides an easy and flexible way of testing the smartphones as Access Requestors.

Firewall To gain more information and have a Feature Consumer ready, an iptables firewall [174] was used. This firewall system also used an IF-MAP enabled client in order to publish (i.e. acting as a Feature Collector) and subscribe (i.e. acting as a Feature Consumer) to information.

Using the security properties defined above, the following particular assignment was used while evaluating the MAP Server.

- The DHCP Server uses an expired certificate for establishing the communication channel. This results in the assignment of the communication channel property to the transmit operation between the DHCP Server and the irond.
- The smartphones are both assigned the smartphone property which indicates the special device class.
- The PDP uses a TPM sealed certificate for the authentication. Thus the appropriate property is assigned which indicates the expected state of this system.
- A basic authentication is carried out between the firewall and the irond. Due to this, the basic auth property tagging the firewall system as an infrastructure-based component is assigned.

By assigning these properties, Trust Tokens holding the correct values (ratings as well as trust levels) should be generated by the MAP Server. This is checked in the second step after the performance of the creation process itself was analysed.

Performance Evaluation

The performance evaluation done here is based on the plain performance evaluation for the irond IF-MAP Server described in [175]. Besides the irond components itself, there are four performance evaluation clients used. These clients are simple C programs which perform a defined flow of IF-MAP operations on the particular server. They are implemented as simple MAP Clients using the libifmap2c (cf. [176]) library which provides easy means of implementing C-based MAP Clients. The particular clients used are

- `perf-pulsing-star-ext`,
- `perf-pulsing-star-int`,
- `perf-complete-graph` and
- `perf-rand-graph`.

All of them are described shortly in the following.

Test Client perf-pulsing-star-ext This small client records the time[1] needed to perform all requests sent to the server. In detail, the client establishes an active subscription for its own published metadata on the server. It starts to publish metadata and following this, polls for changes using the active subscription. Due to this, the time recorded is the time between starting the publish operation and receiving the poll result. Thus this time represents the amount of time the MAP Server needs to process the publish request, include the published data within its graph and provide a pollResult. Given the extended irond server, there should be more time needed as more things are to be stored within the MAP Server's graph and prepared for the pollResult.

The method used to publish data consist of establishing links between certain identifiers by the use of publishing metadata on that link. In detail, this test client starts by a ROOT identifier which has N child identifiers. The links on the same depth level are created using a single publish request. Furthermore, the client does not only establish the tree like structure but also deletes it according to the used parameters. Deletion is performed the same way as the publish update: by simply deleting all links at the same depth within one step. The parameters used for this are at the one side the N value which specifies how many identifiers are attached to the ROOT identifier. On the other side, there is the D value which is used to specify the depth of the graph that should be created. Listing 5.11 shows the algorithm which stands behind this testing program. Given all that, an example run of the program is shown in figure 5.16. It is easy to recognise, that each single step of the client only operates on the same depth in the graph for both actions, the update and the deletion of metadata. If the max depth defined by D is reached, the established graph is deconstructed in single steps. Important about this kind of testing client is the single stepped operations and the creation of the links between depth increasing identifiers.

Test Client perf-pulsing-star-int Using the same subscription and publish mechanism like the `perf-pulsing-star-ext` program, this test client is nearly similar to that one but differs within the link creation. The time is again recorded for all requests allowing to compare different MAP Servers to each other. Listing 5.12 shows the algorithm behind the client in a pseudo code form. The difference between this test client and the external client is given by the creation and the removal of the links between the identifiers. While the external variant attaches and removes links step by step on to the deepest identifier,

[1]Wall time is used for this client as well as for the other testing clients.

```
D, N            // depth and root children
ROOT            // root identifier
idents[D][N]    // all identifiers, array as in C
preq            // publish request

subscribe update for ROOT

// building the tree
for d in (0 ... D - 1) do
 for i in (0 ... N - 1) do
  if (d == 0) then
   add update(ROOT, idents[d][i]) to preq
  else then
   add update(idents[d - 1][i], idents[d][i]) to preq
  end if
 end for
 send preq to server, reset preq
 poll server
end for

// removing the tree
for d in (0 ... D - 1) do
 for i in (0 ... N - 1) do
  if (d == (D - 1)) then
   add delete(ROOT, idents[D - d - 1][i]) to preq
  else then
   add delete(idents[D - d - 1][i], idents[D - d - 2][i]) to preq
  end if
 end for
 send preq to server, reset preq
 poll server
end for
```

Listing 5.11: perf-pulsing-star-ext pseudo-code [175].

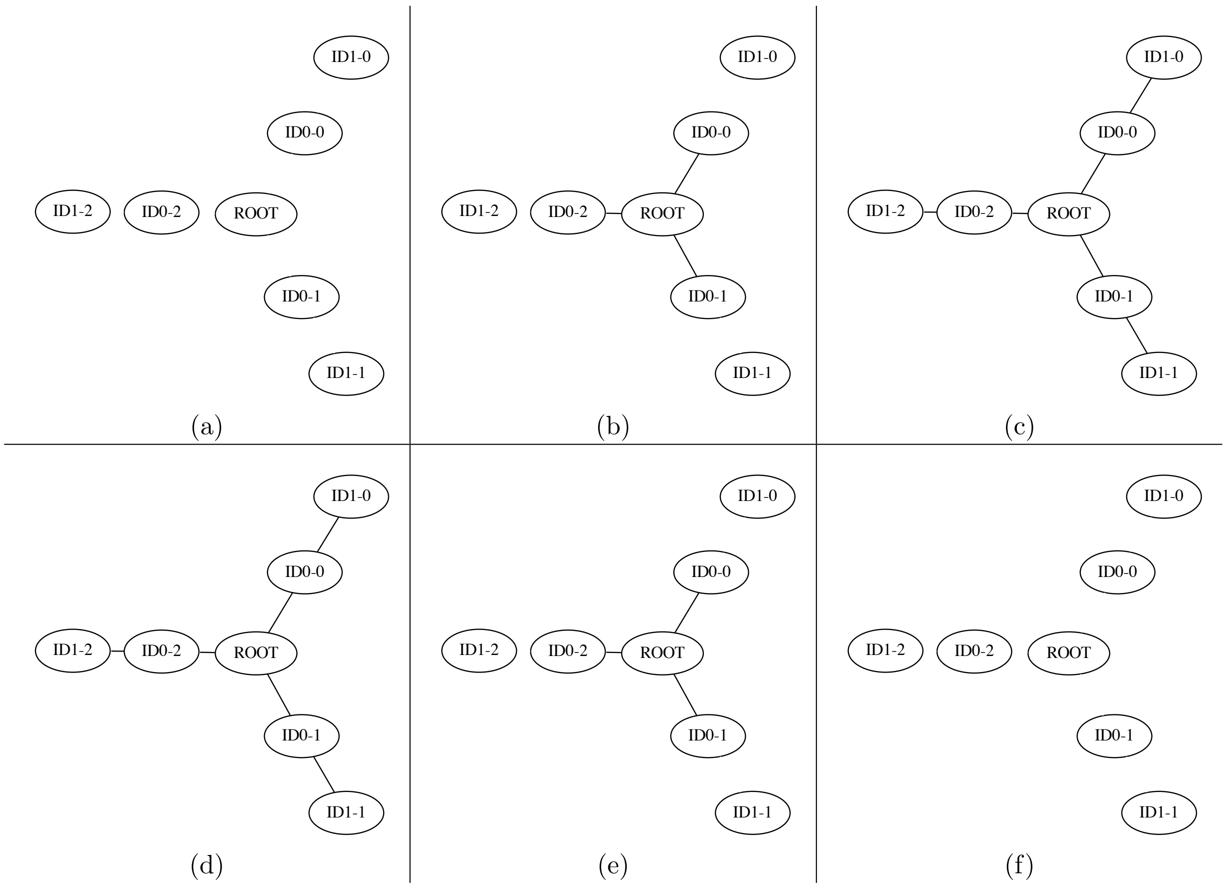


Figure 5.16: Example run of perf-pulsing-star-ext with $N = 3$, $D = 2$ [175].

```
D, N            // depth and root children
ROOT            // root identifier
idents[D][N]    // all identifiers, array as in C
preq            // publish request

subscribe update for ROOT

// building the tree
for d in (0 ... D − 1) do
  for i in (0 ... N − 1) do
    if (d != 0) then
      add delete(ROOT, idents[d − 1][i]) to preq
      add update(idents[d][i], idents[d−1][1]) to preq
    end if
    add update(ROOT, idents[d][i]) to preq
  end for
  send preq to server, reset preq
  poll server
end for

// removing the tree
for d in (0 ... D − 1) do
  for i in (0 ... N − 1) do
    if (d != (D − 1)) then
      add delete(ROOT, idents[D − d − 1][i])
      add update(ROOT, idents[D − d − 2][i])
    end if
    add delete(ROOT, idents[D − d − 1][i]) to preq
  end for
  send preq to server, reset preq
  poll server
end for
```

Listing 5.12: perf-pulsing-star-int pseudo-code [175].

this internal client attaches new links to the ROOT identifier. In addition to that, the previously created links which are also added to the ROOT node only, will be removed and new links are attached to the currently handled identifiers. Figure 5.17 depicts the graph created and deleted by using this test client. More in detail, the external client

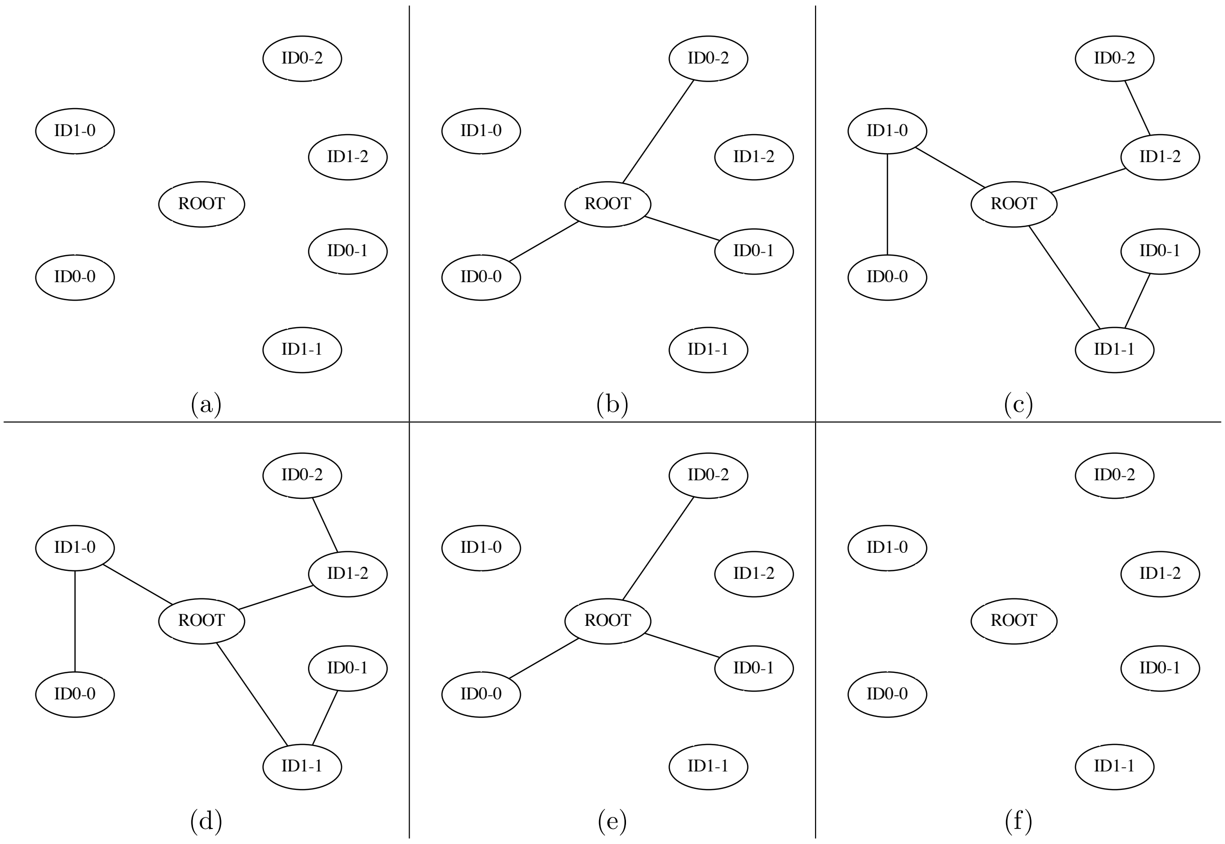


Figure 5.17: Example run of perf-pulsing-star-int with $N = 3$, $D = 2$ [175].

never used a mix of update and delete elements within the same publish request. This internal variant relies on this mixing and increases the amount of work the MAP Server has to do within one step. The two remaining test clients differ in their methods used to stress the MAP Server.

Test Client perf-complete-graph This test client creates a complete graph, i.e. a graph where each identifier is connected through a link to each others identifiers. In difference to the first two star-based test clients, this client creates one subscription per identifier published. This subscription is set to a maximum depth of number of identifiers minus one, so that each subscription basically returns the whole graph. Listing 5.13 shows the basic algorithm that is carried out when using this test client. Due to the fact, that the original version of this client was not applicable with the irond-trust MAPS, the version depicted in 7.9 was used. This version uses another variant of checking the count of the returned metadata and takes the Trust Tokens into account which effectively doubles the metadata returned. Contrary to the previously explained clients, this client steps

```
N              // number of identifiers
idents[N]      // all identifiers, array as in C
preq           // publish request
sreq           // subscribe request

// create subscriptions
for i in (0 ... N − 1) do
  add update subscription for ident[i] to sreq
end for
send sreq to server

// building the graph
for i in (0 ... N − 1) do
  for j in (i + 1 ... N − 1) do
    add update(ident[i], ident[j])  to preq
    send preq to server, reset preq
    poll server
  end for
end for
```

Listing 5.13: perf-complete-graph pseudo-code [175].

forward by performing multiple publish requests. That is, each connection step consists of severe sub steps rendered by appropriate publish requests. Within these requests, the links between the identifiers are established. The graph which is created by this client including the appropriate construction steps are shown in figure 5.18. Although a complete graph is created when running this client, no deletion of this graph is made when the client has finished, thus it records the creation step times only.

This type of client was originally developed to evaluate the performance of the irond server's data model. This is due to the fact, that there is a high amount of internal work to do in order to connect the identifiers to each other from the data models point of view. Furthermore, it is not only stressing the server to build the links to each other but also to maintain these relationships. Using this test client on the trust extended irond should

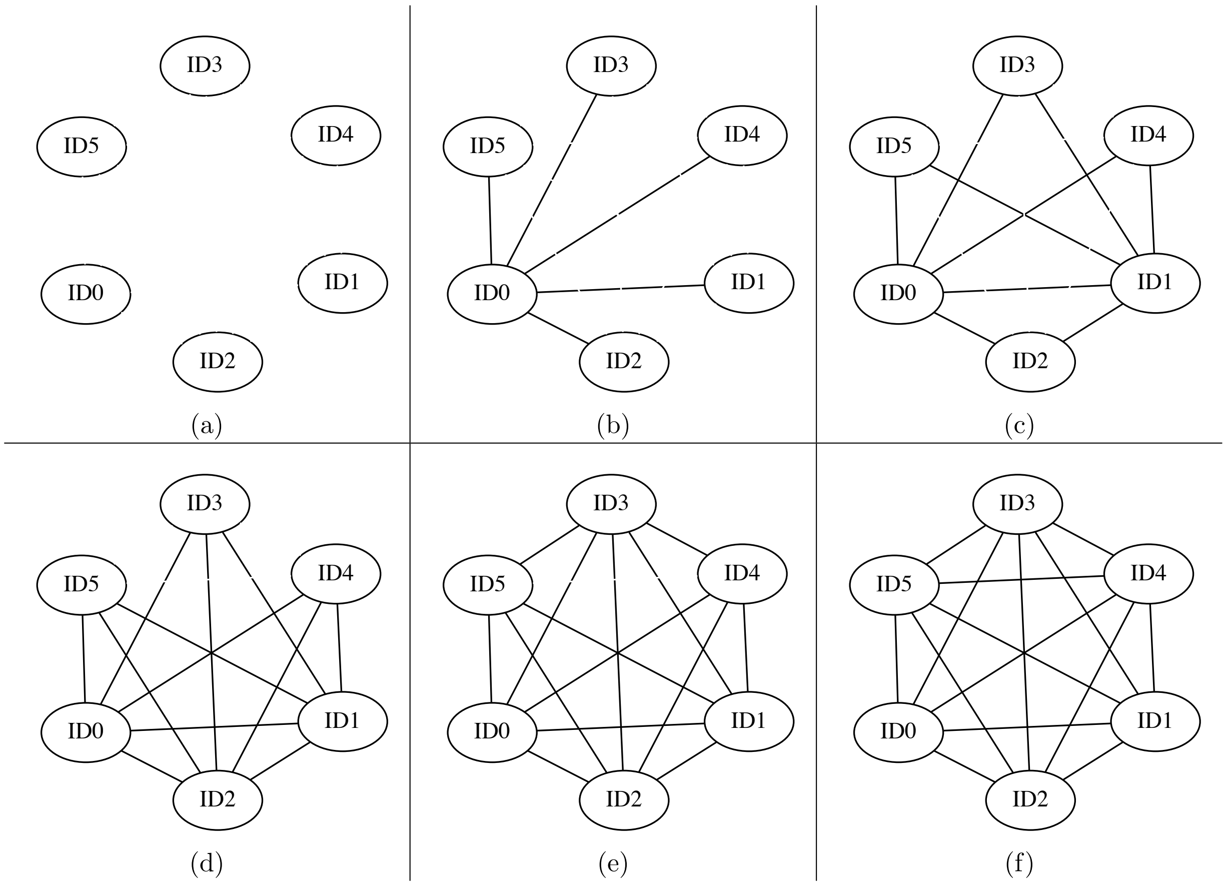


Figure 5.18: Example run of perf-complete-graph with $N = 6$ [175].

uncover possible performance related problems, when combining Trust Token Metadata with such complex graph structures.

Test Client perf-rand-graph The last of the used test clients operates similar to the previously introduced complete graph variant. The algorithm carried out by this test client is depicted in listing 5.14. Instead of connecting the identifiers to each other, they

```
N              // number of identifiers
M              // operations per possible link
P              // probability of delete 0 ... 100
idents[N]      // all identifiers, array as in C
preq           // publish request
sreq           // subscribe request

L = (N * (N - 1) / 2 // number of possible links
                     // between all identifiers

initialize PRNG using N

// create subscriptions
for i in (0 ... N - 1) do
  add update subscription for ident[i] to sreq
end for
send sreq to server

for i in (0 ... (M * L - 1)) do
  I1 <- pseudo random from idents
  I2 <- pseudo random from idents != I1

  if (pseudo-random number) % 100 < P then
    add delete(I1, I2) to preq
  else
    add update(I1, I2) to preq
  end if

  send preq to server, reset preq

  if was not delete then
     poll server
  end if
end for
```

Listing 5.14: pref-rand-graph pseudo-code [175].

are randomly[2] selected and linked or unlinked to each other. The decision, if they are linked or unlinked, is made by their previous linking status. That is, if they are already linked to each other, they will be unlinked and vice versa. Furthermore, this behaviour can also be controlled by the request the client used: it is either an update or delete request. If a delete request is used, the linking and unlinking behaviour is inverted by default. The decision about which request is issued by the client is also made by using the `rand()` function. In order to receive comparable results, the random number generator used can be seeded prior to its usage.

All of these test clients do not regard the overhead generated by invoking these operations using the IF-MAP protocol. That is, the network-based communication delay is not directly treated by the client programs. In order to receive comparable results, they should all run on the same machine with the same environmental settings.

Test Results In order to provide comparable results, all tests were run on the same Intel Xeon-based machine. This machine possessed 8 gigabytes of memory and dual CPUs. The Java version used for the tests was an 1.6 version of the openjdk. Furthermore an Ubuntu-based Linux was used as operating system on the test machine. Both irond version were reset after each run, thus cleaned up and restarted to provide a common starting point. In addition, the test clients were also placed on that machine to avoid network-based influences. Although the client server communication was carried out via IF-MAP, there was no real network traffic involved.

All four test clients described were used to compare the performance of the trust extended irond version against the default irond version. The particular irond implementation that was used for the trust version is based on irond 0.3.4. To provide real world results, this version was compared against the default 0.3.5 implementation, which offers even greater performance. This allows to assess the results in a more critical way.

The first test was carried out using the `perf-pulsing-star-ext` test clients. The results are depicted in figure 5.19. The setup used for this test was given with seven particular runs of the client each with an increasing amount of identifiers and one star-based request. The identifier amount was set to 4, 8, 16, 32, 48, 64 and 128. This can be easily seen when comparing the points at which the response times have been recorded. Interestingly the

[2]In detail, they are chosen by calling rand() on the system, thus the randomness depends on the implementation of this system function.

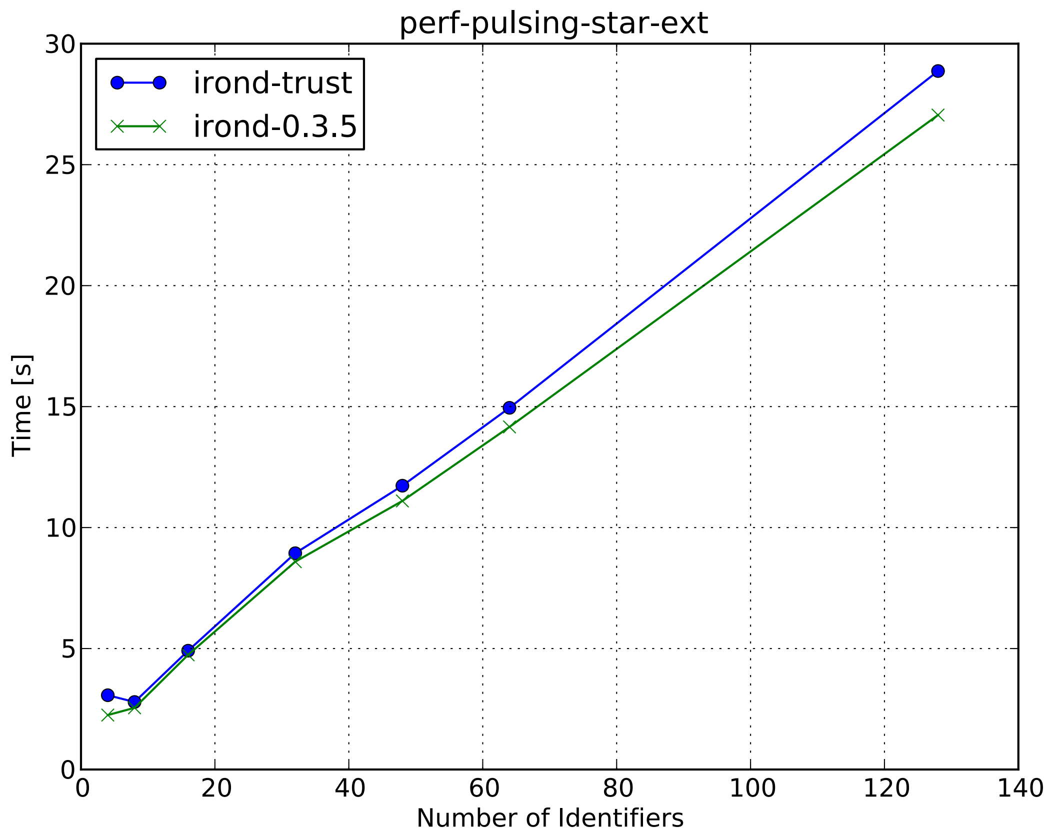


Figure 5.19: Performance comparison using the perf-pulsing-star-ext client.

irond-trust (i.e. the trust extended irond version) is performing equal within the first two steps with small identifier amount. This is rather unexpected and may be due to the way the Java VM operates and pre-caches elements, thus it is considered to be the normal amount of variability of such measurements. The remainder of the graph until the maximum count of identifiers is as it was expected, the irons-trust is somewhat slower than the default irond. However, the difference can be considered to be very small. This small difference is the result of the data model used by the irond. The Trust Token generation doubles the particular amount of metadata within the graph but the irond's internal model handles this in a very effective way.

The second test is based on the `perf-pulsing-star-int` test client. This test client uses another way of constructing the links between the identifiers. The results of the runs which were achieved using this client are shown in figure 5.20. As the graph shows, the

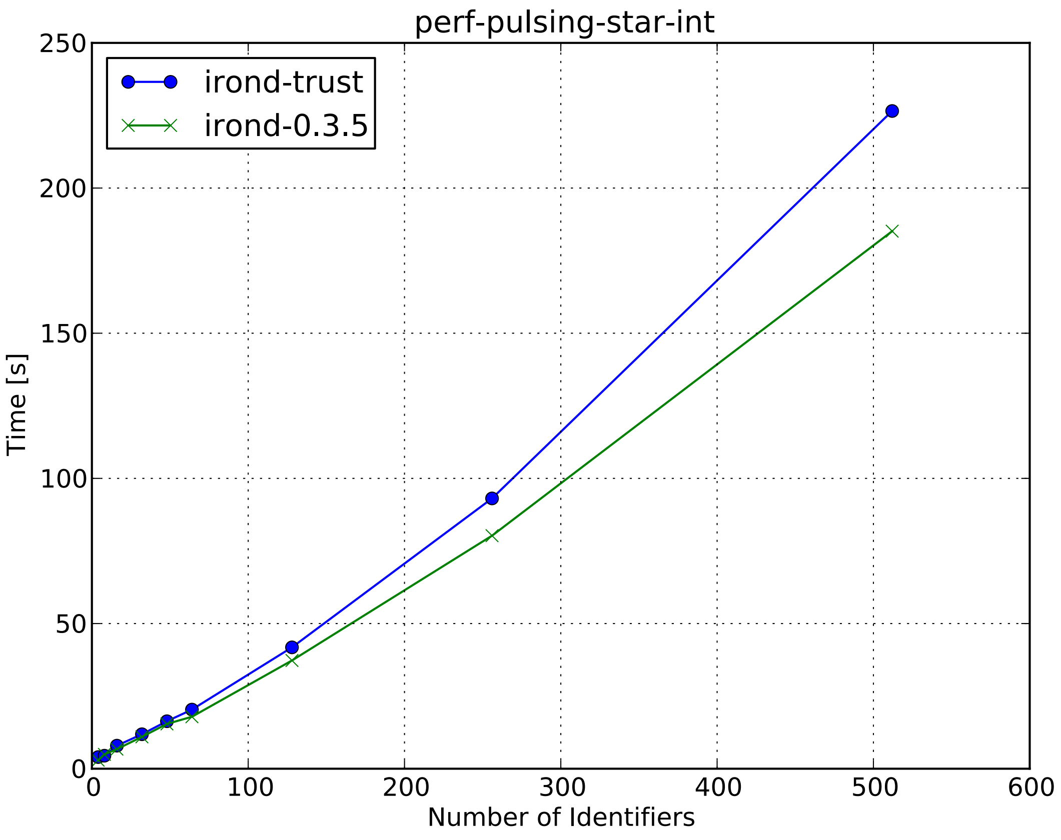


Figure 5.20: Performance comparison using the perf-pulsing-star-int client.

identifier amount used this time was significantly higher than in the first test. This helps to evaluate the performance under circumstances within which a huge amount of identifiers are linked to each other. The particular amounts used started with 4 again and ended at 512. Furthermore, there was only one pulse (i.e. one construction and deconstruction cycle) per run as further test did not show any changes in the results when using more than one pulse. Given the overall graph, the behaviour which can be derived is the same as within the first test. That is, the irond-trust is slower but not in a significant way. Even when considering the last measurement with an amount of 512 identifiers and a complete construction, linking, unlinking and deconstruction cycle, the difference between both versions is about 30 percent. Given the overall time of this operation and the high amount of identifiers, this can be considered as acceptable.

Both of the tests which were evaluated above did only generate one link between two particular identifiers without linking the identifiers to each other. In order to increase the metadata amount, the two following tests were used. First, the `perf-complete-graph` client was used to link the identifiers to each other, thus generating a high amount of metadata. In case of the irond-trust, the amount of metadata generated doubles due to the Trust Tokens. Figure 5.21 depicts the result of this test. Equally to the first two tests,

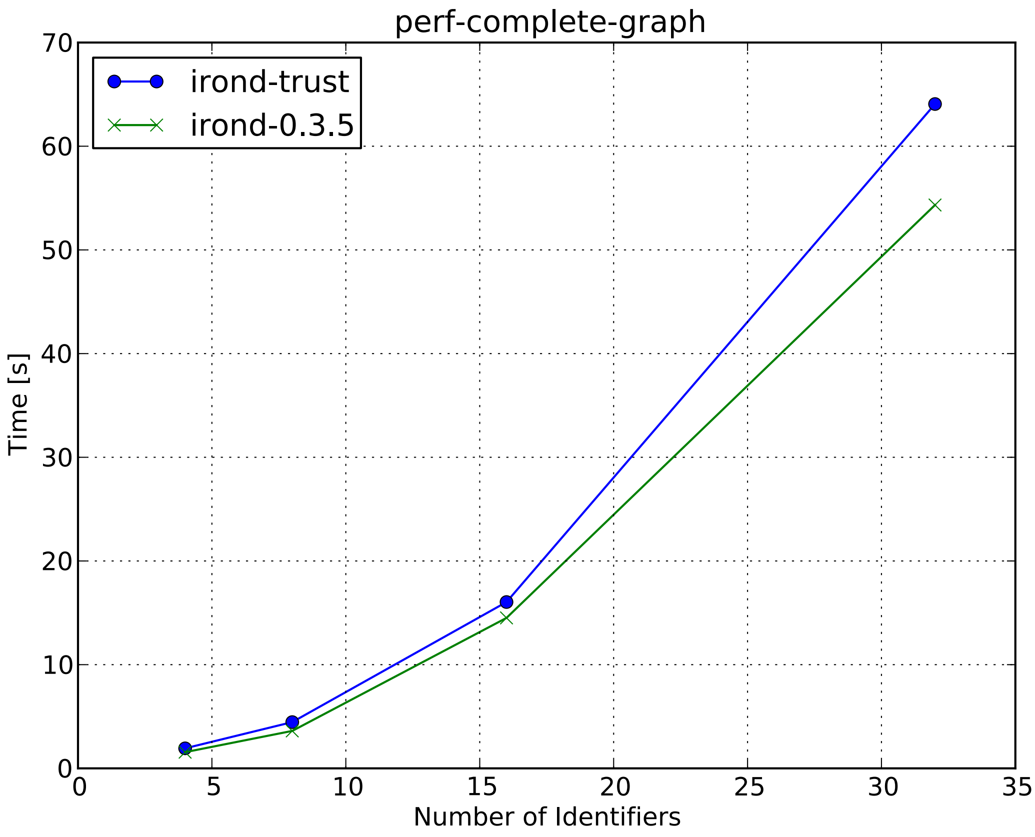


Figure 5.21: Performance comparison using the perf-complete-graph client.

several runs with different identifier amounts were made, starting with 4 identifiers and ranging to a maximum of 32 identifiers. All of these were linked to each other. The results which can be seen in the graph show a similar picture as the first two tests. That is, the irond-trust is again a little bit slower but within an acceptable range. The particular difference with the highest amount of identifiers is about 20 percent. Furthermore, it must be considered that linking an amount of 32 identifiers to each other produces 992 links in case of the irond-trust. Due to this, the result can be considered as reasonable and thus acceptable.

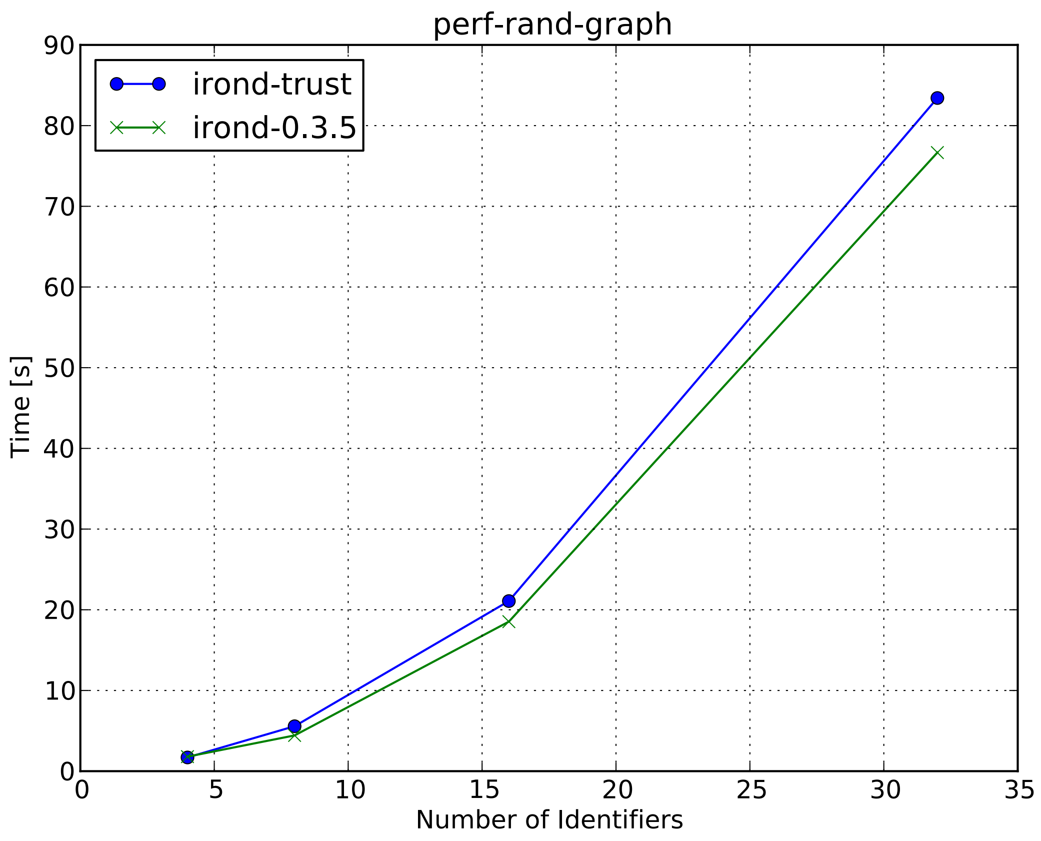


Figure 5.22: Performance comparison using the perf-rand-graph client, first run with an maximum amount of 32 identifiers.

The last performance-based tests that were carried out used the `perf-rand-graph` test client. This client takes a certain number of identifiers and randomly connects them to each other. Because of this randomness, it provides a way of simulating a real world like scenario where publish and delete operations are received by the server in an rather not deterministic way. There were three overall tests done with this client. The first and the second one (depicted in figure 5.22 and 5.23) have the same basic settings. They were carried out twice to show that there are no significant differences between them although they are using random data. This stability is reached due to using the same seed for the random number generator.

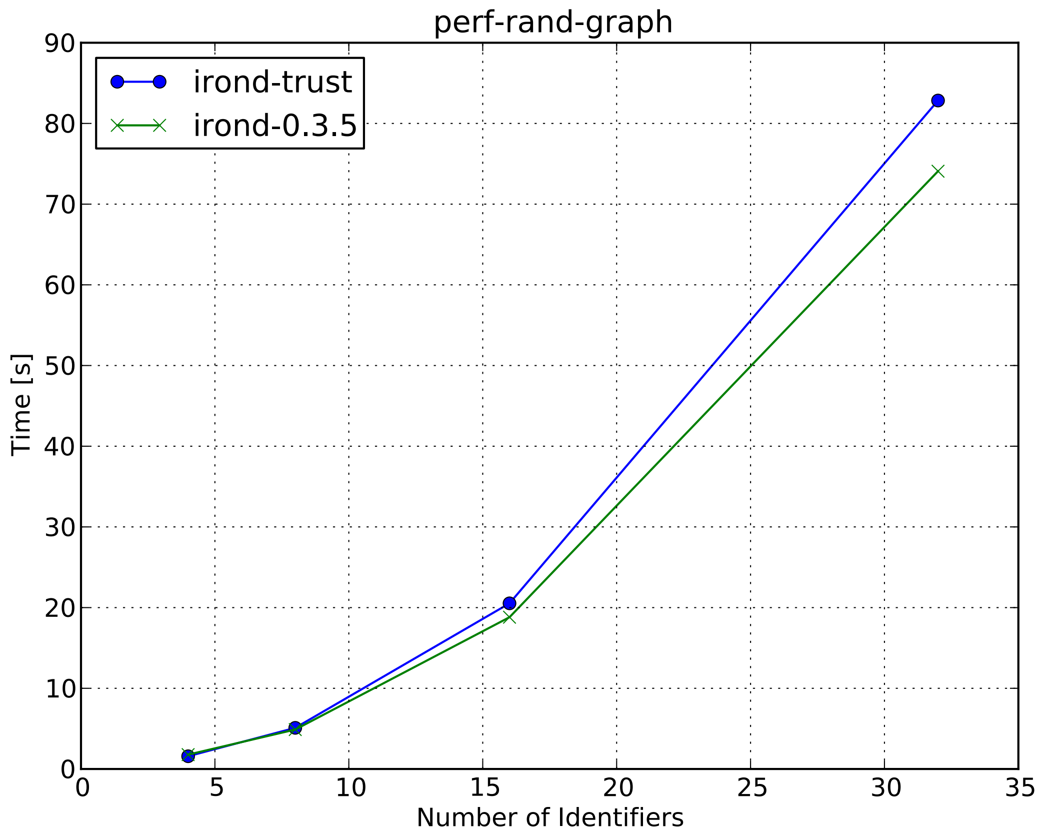


Figure 5.23: Performance comparison using the perf-rand-graph client, second run using the same seed as in the first run and again with an maximum amount of 32 identifiers.

The first particular test is based on the same amount of identifiers used for the complete graph test. As it easy to see in figure 5.22, the difference is again from minor impact. In detail, with the highest identifier amount of 32, the irond-0.3.5 performs about 15 percent faster than the trust extended irond. Comparing this result to the complete graph test, it can be summarised that the random graph test produces overall higher response times.

However, the particular difference in the response times of both irond version are smaller than in the complete graph test.

In order to confirm the results of this first random graph test, a second test with the same parameters was done. The results of this test are depicted in figure 5.23. As it is easy to discover, the results are very similar to the first run. Due to this, the test can be considered as valuable, thus showing and emphasising the small differences again.

As the amount of identifiers used in the previous random graph test was rather limited and to verify that the differences between the two version do not increase in an unexpected way, a final test with a huge identifier amount was carried out. The result of this test is pictured in figure 5.24. The identifier amount used for this test was starting by 4 again

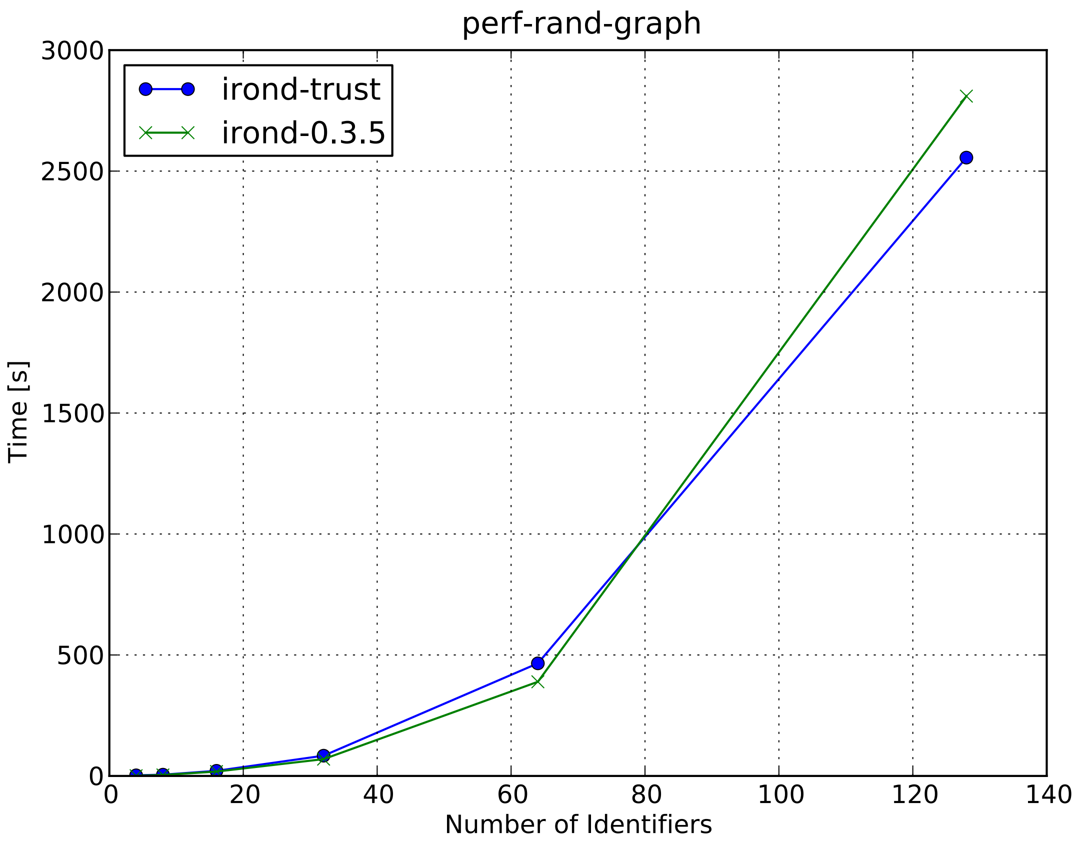


Figure 5.24: Performance comparison using the perf-rand-graph client and using an maximum amount of 128 identifiers.

and ranging to 128 which could theoretically produce 16256 metadata elements for the irond-trust. As the measurement results show, the response time increases to a very high amount, peaking at more than 2500 seconds when using 128 identifiers. This shows that this kind of test setup produced a very stressing situation, which may be difficult to reach in a real world situation (due to the request would be performed in single steps). However, it allows to gain an impression about how both versions behave under such circumstances. Given the graph, it is interesting to see that the irond-trust becomes faster than the default irond between 64 and 128 identifiers. Until that point, the results are as expected and confirm the previously made findings about the extended irond being somewhat slower. However, at 128 identifiers, the irond-trust performed faster within this test case. This is rather unexpected and being considered as measuring inaccuracy. Discussing this, reasons for this result may be either some disturbance on the system itself or the special caching a garbage collection paradigm used within the Java VM the educational ironds were running in.

Summarising these tests, the irond-trust must be considered as about 10 to 20 percent slower than the irond-0.3.5. Due to the doubled amount of metadata to be handled by the trust extended version and the processing of the trust extension itself, this is a very acceptable result. Given this, the irond-trust may also be used for providing a MAP Server's tasks as the response times are still very acceptable. If comparing them to other MAP Servers (cf. [2] for such an comparison), the irond processes requests overall a lot faster than other servers do. Due to this, the irond-trust can still be considered faster than other MAP Servers.

Trust Token generation

This section summarises the evaluation of the particular trust calculation process. That is, the above defined environment was used to simulate trust token calculations. These calculations were checked against the expected (i.e. theoretical) results. This approach showed that the calculated trust tokens matched the expected values. Furthermore, the flow of operations used in this test consist of all necessary operation types. That is, trust tokens are either created or updated. Creation takes place if new metadata is added while updates takes place if particular ratings are changed.

Based on the test described in [166], the following flow of operations is used in order to produce a well defined Trust Token.

1. The DHCP Server publishes an ip-mac metadata which is set to lifetime forever and is linked between an ip-address and a mac-address identifier. The ip address is set to 192.168.0.1 while the mac address is given as aa:bb:cc:11:22:33. This represents the smartphone used within this test. Figure 5.25 depicts the graph which results this step.

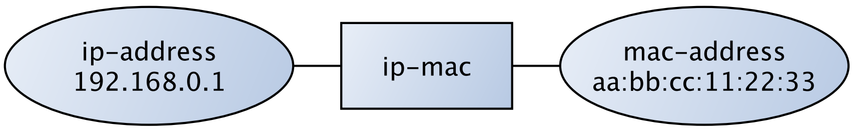


Figure 5.25: Created graph after the first step.

2. Based on the previously set ip address, the firewall subscribes to this ip. That is, the ip-address identifier is used as starting point of the subscription.

3. Continuing after the subscription, the firewall performs the poll operation on the irond. This poll operation returns a searchResult which consists of the three elements that were created in the first step and the appropriate Trust Token as additional metadata. The Trust Token holds all trust-specific data for the ip-mac metadata. It is depicted in figure 5.26.

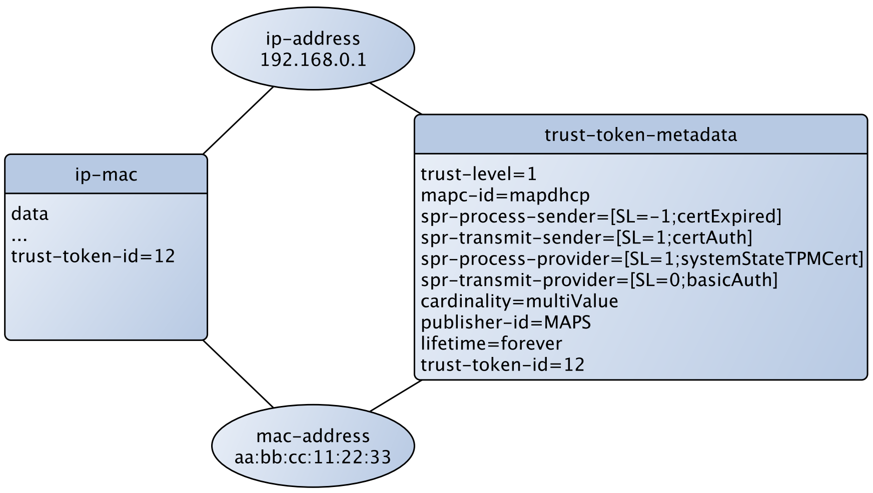


Figure 5.26: Created Trust Token Metadata [166].

4. The PDP uses the same search filter on the MAP Server like the firewall used for the subscription. Doing so, it receives also a searchResult which consists of the elements published within the first step. Although there is also a Trust Token attached, it is not the same as in the previous step. This is due to the different last transmit operation used. However, this is a client centric operation which results in an unchanged graph on the MAP Server.

5. The PDP itself now publishes the metadata access-request-ip and capability. The access-request-ip is published on to the link between the already existing ip-address and a new access-request. This access-request uses the dummy value ar012345678. In addition, the capability with its value *invincible* is published on to the access-request also (see figure 5.27).

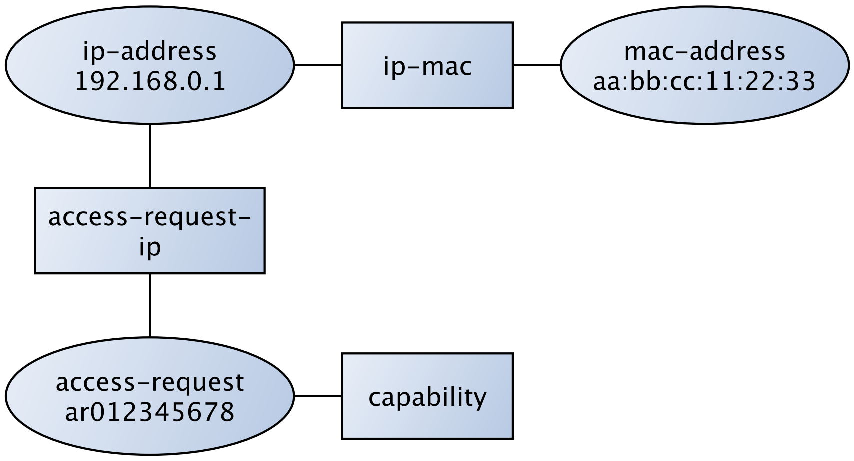


Figure 5.27: Created graph after the PDP publishing step.

6. As the firewall has an active subscription and an open ARC, it receives another pollResult after the last step. This pollResult holds the added elements along with their TT. There are two Trust Tokens included: the one for the access-request-ip metadata as well as the one for the capability metadata. This must also be considered as a client centric operation which does not affect the graph maintained on the MAP Server.

7. The smartphone now publishes using a publish-notify operation event metadata of the type p2p on the ip-address identifier. The resulting graph, which is only temporary as the publish notify-based metadata is not stored is shown in figure 5.28.

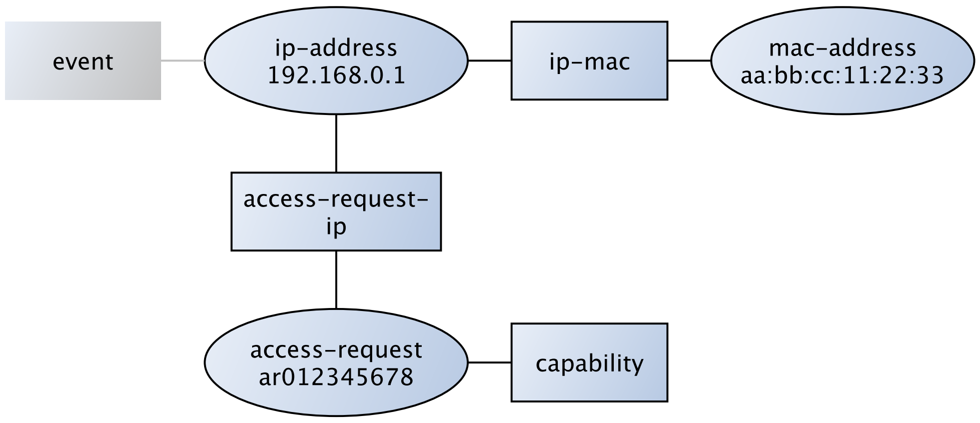


Figure 5.28: Temporary graph including the publish notify-based metadata.

8. Given the subscription of the firewall and the active ARC again, another pollResult is received. This time, a notifyResult is received including the event as well as the appropriate Trust Token. No entry for the published event data is made in the graph due to the type of operation used. The overall graph, which includes all elements for a better understanding is depicted in figure 5.29. In addition to the metadata itself, the trust tokens (depicted as red metadata) are also included along with their appropriate relationships. The temporary elements which where created by a publish-notify operation are grey coloured in order to distinguish them from persistent graph elements.
9. The DHCP Server now performs a delete operation: the access-request-ip metadata as well as the capability metadata is deleted.
10. As the subscription of the firewall is still active, a deleteResult is now received. This result consists of the deleted metadata and their assigned Trust Tokens. In particular, the Trust Tokens now hold the id of the deleting MAP client within the mapc-id field indicating which client is responsible the the operation.

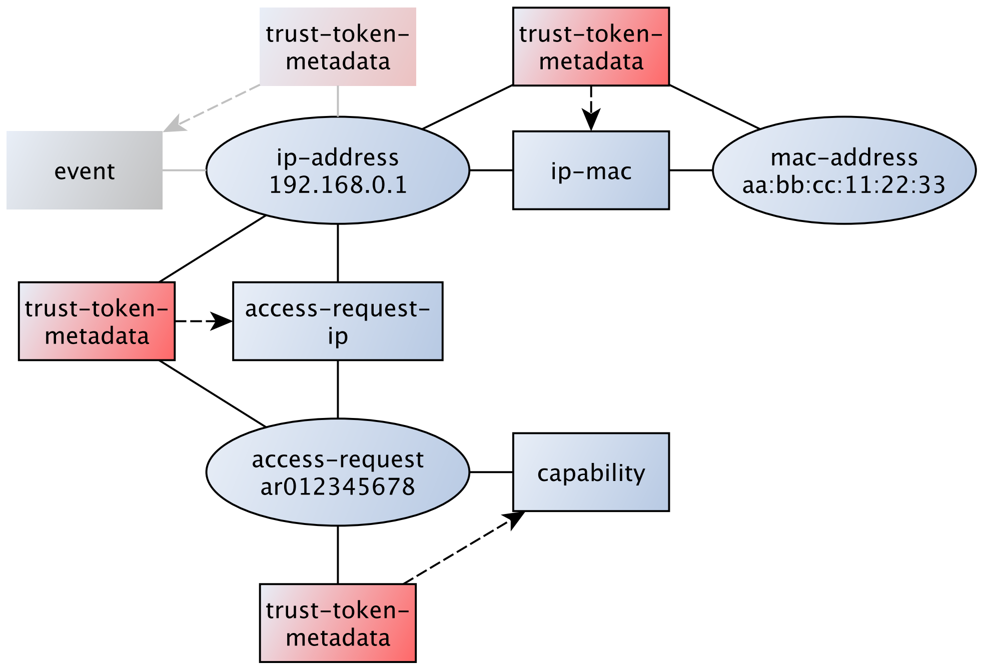


Figure 5.29: Overall graph including the trust token metadata as well as the temporary publish notify elements.

11. The DHCP Server performs an endSession operation thus ending its session. Due to the ip-mac metadata being published with a lifetime of forever, it is not being discarded. Following the endSession, the DHCP Server just performs a newSession operation which establishes another session for it on the MAP Server. Additionally, the rating of certain SPs within the policy is also changed and the irond is triggered to update the rating.

12. A purgePublisher operation is carried out by the DHCP Server. This operation leads to the deletion of the ip-mac metadata and results in the situation were no metadata is left on the graph.

13. Due to the still valid subscription of the firewall, another pollResult of the type deleteResult is received on the ARC. This deleteResult holds the purged ip-mac metadata and the appropriate Trust Token. This Trust Token contains the updated security properties as well as the mapc-id of the purging client.

14. In order to test the TT constraints on the MAP Server, the DHCP tries to publish a Trust Token on its own. This triggers an errorResult message containing an AccessDenied error from the MAP Server. Thus, the DHCP Server is unable to publish TTs.

The presented flow of operations showed all three types of trust token-specific handling: trust token creation, updating an deletion. Comparing the gathered results against the expected results, it can be concluded, that the extends MAP Server perform as expected in terms of trust token handling. Due to this, there are no reservations against using it for the next evaluation step.

5.6.3 TCADS Environment

The given environment from above, consisting of the evaluated trust-enhanced MAP Server, is extended to the following architecture. This architecture now includes all relevant components and is based on the evaluation environment used in [2] for the CADS evaluation. It is depicted in figure 5.30. Given this figure, the blue elements depict general services like the PDP or the vulnerability scanner. All other colours indicate which kind of TCADS role is assigned to that component. That is, the red elements illustrate Feature Collectors, orange elements illustrate the trust extended Feature Provider, yellow is used for the Correlation Engine and green depicts Feature Consumers. In contrast to the environment which was used for a plain CADS-based evaluation, this particular environment is enhanced with the trust-specific components. This is on the one hand the irond-trust as MAP Server and on the other hand the TrustLog enhanced Correlation Engine. Furthermore, an additional Feature Consumer (another ironmonitor instance) is used on the irond-trust's system. It is used to receive rating change instructions from the Correlation Engine.

Testcase: Sensor Sniffing with compromised Feature Collector

To test the TCADS system and evaluate its behaviour, a sensor sniffing attack is launched by the smartphone. In detail, an application installed from an official source (i.e. Google's Play Store, [177]) provides access to the smartphone's camera and microphone sensors. The access method is realised by starting a small web server which allows it to connect to

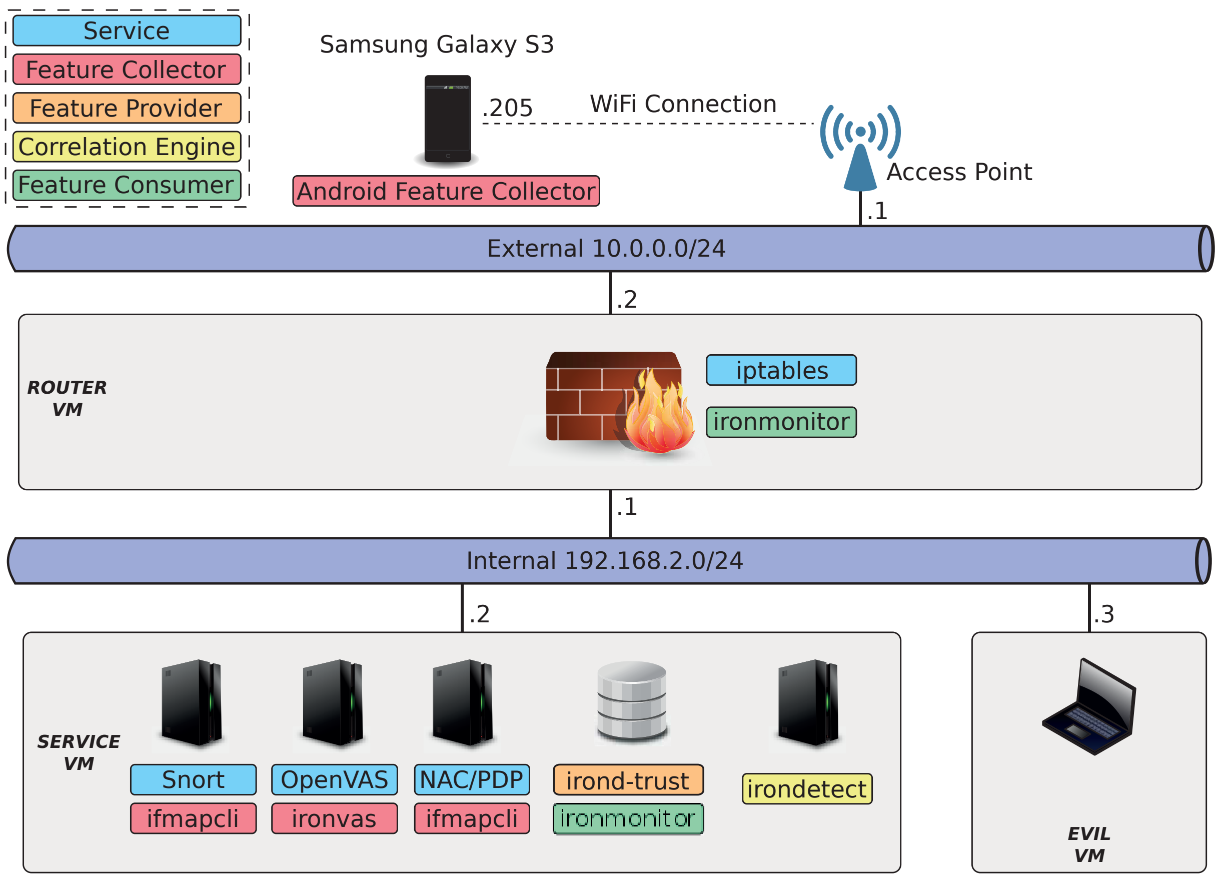


Figure 5.30: TCADS evaluation environment used to simulate a sensor sniffing-based attack (cf. [2]).

the smartphone using a simple web browser. The task of connecting to the smartphone browser-based is carried out on the Evil VM, which simulates the attacker's system. Within this browser, the sensors of the phone are accessible thus a live stream is provided by the installed application. Contrary to the CADS evaluation, the Feature Collector on the smartphone is also compromised. This results in the problem, that this Feature Collector makes a false reporting about the smartphone's properties, in detail about the traffic the smartphone produces in order to stay undetected. When connecting to the app on the phone and starting the live stream the traffic should normally increase drastically. This is no longer being recognised due to the compromised Feature Collector. Using other information sources from the network, in particular the Snort-based IDS the CE can discover the compromise on the smartphone to some extent. Due to this, the CE lowers the smartphone's FC's rating subsequently. Finally, a situation is reached where the trust level trend of this FC runs under a certain threshold, thus triggering the enforcement of the device and blocking access to the live stream.

Test environment

As already described, the test environment is based on the environment used for the CADS evaluation in [2]. Due to this, most of the components are similar to the ones which were used for the basic CADS evaluation. However, a short explanation of the used systems is given in the following. The test environment was established using some physical devices as well as three virtual machines (VMs). As the test was carried out multiple times, the VMs have been deployed using VirtualBox on two different host systems: OS X 10.8 as well as Ubuntu Linux 12.04. Furthermore, the three VMs used simulated different responsibilities within the environment. There is a Service VM which hosts all required services, a Router VM responsible for establishing the network connection between the inner and outer network as well as providing enforcement methods and the Evil VM acting as an attackers device. The components which are housed inside these VMs along with the particular physical devices are explained in the following.

Samsung Galaxy S III This particular device is used as the smartphone within the tests. It is a common device type running a rather actual Android 4.1.1 as operating system. The used App for providing the live stream feature has been installed on it, along with the Feature Collector measuring the device's properties. Given the environment, a network connection is established by the use of the access point, thus a Wifi-based connection is used throughout all tests.

Access Point A Lancom L-54g (cf. [178]) device provides a wireless network to the smartphone. In detail, a 802.11g type network (cf. [179]), a very common type, is used. The access point itself connects to the router VM and forms the outer network (10.0.0.1/24) together with the smartphone.

iptables This service was installed within the Router VM in order to (1) provide access from the outer network to the inner network and (2) to establish an enforcement mechanism which is able to disconnect outer devices on demand. An Ubuntu Linux (12.04) is used to provide routing means. Furthermore, the iptables/netfilter implementation performs the particular enforcement tasks. These tasks are received as Features by the ironmonitor implementation. This implementation is responsible for triggering the appropriate rule changes of the iptables service.

Snort All of the remaining services are located on the Service VM which is also an Ubuntu Linux-based platform running version 12.04. The Snort service is used as IDS system which recognises certain changes, in particular traffic increases of outer network devices. In order to allow easy testing, the ifmapcli implementation simulating Snort's responses is used as a Feature Collector.

OpenVAS This service acts as vulnerability scanner which is responsible for scanning outer network-based devices, in particular the smartphone. Results of these scans are provided as Features by using the ironvas [180] implementation which takes OpenVAS results and publishes them as appropriate Features.

PDP To provide the root for the Feature tree within the MAP Server, an access-request is required. This service is responsible for performing a basic NAC handshake and publishing the appropriate access-request for the smartphone. As with the Snort service, in order to allow for a flexible way of testing, the ifmapcli implementation was used as a Feature Collector providing this required access-request.

irond-trust The Feature Provider used within the tests is given by this service. It is based on the previously evaluated irond-trust. The irond-trust implementation is an enhanced version of the default 0.3.4 implementation of the irond that was used for the basic CADS evaluation. Additionally, the appropriate SPM which defines the required security properties was directly placed on the Service VM. To allow a change of the ratings by the Correlation Engine, the ironmonitor implementation is used to receive appropriate Features and perform the SPM changes.

irondetect This service provides the Correlation Engine and forms the core of the TCADS system together with the irond-trust. The extended irondetect implementation is utilised. It is based on the version that was used for the CADS evaluation but incorporates the TrustLog data model.

In addition to the VM-based components and the physical devices, the Ip Webcam App on the smartphone is used to provide sensor sniffing means. This app is explained shortly in the following.

IP Webcam App The app is provided by the official Google Play store and can be found here [181]. It enables the smartphone's camera and microphone to be accessed remotely. As already stated, this is done by deploying a small web server on the smartphone itself and opening up access to this server by outside connections. This web server simply holds one web page which allows to access the camera's live stream (including audio) by different methods. These methods allow to use for example a Flash-based access or a browser-specific access which utilises the appropriate browser plugin. Due to this, there is eventual no need of installing third party software besides the browser on the machine that is connecting to the app. Furthermore, the app has several convenience settings. Besides the controllability given to the remote clients, there are options to hide the app when running and to auto start the app on boot. Hiding the app means, that is not longer being shown in Android's notification bar, thus making it difficult to be recognised by the user. Starting on boot means that there is no interaction required in order to run the app when rebooting the device. Combining this with the hiding option effectively puts the app into a state where the user may be unaware that it is actually running. However, it is still retrievable from the official app store and is rated rather high. Based on these special properties, a TCADS policy can be formulated which includes all necessary parts to recognise the sensor sniffing and to disconnect the device from the network.

Policies There are two policies which need to be set for the test case: the initial security property setup including the appropriate ratings (i.e. the SPM) and the Correlation Engine policy. Section 5.6.1 defines the initial SPM setup, which is being used here. Based on this the policy is simply created by using the defined properties and their initial ratings. Besides this rather straightforward task, the Correlation Engine policy must be set. In particular, the policy being used is based on the CADS evaluation policy shown in [2] and extended according to the trust requirements. The resulting policy is given in listing 5.15 which is shown below.

```
context {
    ctxWorkingHours := DATETIME > "06:00" and DATETIME < "22:00";
    ctxTrustedInfrastructureComponent := TRUSTLEVEL > "0"
    and DATETIME > "6:00" and DATETIME < "22:00";
    ctxIrondetect := TRUSTLEVEL = "1";
}

hint {

    trafficHintSmartphone := "smartphone.communication.ip.txother"
    "de.fhhannover.inform.trust.irondetectprocedures.TrendByValueCW" "50";
    trafficHintSnort := "ids.snort.event.hightraffic"
    "de.fhhannover.inform.trust.irondetectprocedures.TrendByValueCW" "100";
    trustTrendHint := "trend.measurement.event"
    "de.fhhannover.inform.trust.irondetectprocedures.Trend" "10";
}

anomaly {

    anoHighTrafficSmartphone := trafficHintSmartphone > 0.5 ctxWorkingHours;
    anoHighTrafficSnort := trafficHintSnort > 0.5 ctxTrustedInfrastructureComponent;
    anoLowTrafficSmartphone := trafficHintSmartphone <= 0.5 ctxWorkingHours;
    anoTrustTrend := trustTrendHint > 0.5;

}

signature {
    sigCamera := "smartphone.sensor.cameraisused" = "true" ctxWorkingHours;
    sigSuspiciousApp := "smartphone.android.app.permission.granted!1" = "android.
        permission.RECEIVE_BOOT_COMPLETED"
  and "smartphone.android.app.permission.granted!1" = "android.permission.CAMERA"
  and "smartphone.android.app.permission.granted!1" = "android.permission.INTERNET"
      ctxWorkingHours;
    sigPortOpen := count ("vulnerability-scan-result.vulnerability.port") > "0"
        ctxTrustedInfrastructureComponent;
    sigNoReqForInv := count ( "ids.snort.event.hightraffic" ) = "0";
}

```

```
condition {
    conDataLeakDetected := sigSuspiciousApp and sigCamera and sigPortOpen and
        anoHighTrafficSmartphone;
    conNoDataLeakDetected := anoLowTrafficSmartphone and sigNoReqForInv;
    conSplitResults := anoLowTrafficSmartphone and anoHighTrafficSnort;
    conAlertAndNegativeTrustTrend := anoTrustTrend;
}

action {
    requestForInvestigation := "RequestForInvestigation" "@smartphone.device.ipaddress";
    decreaseSprAction := "trustsprps" "./change-security-property.sh" "$1" "1";
    enforcementIsolate := "enforcement.action" "./drop-client.sh" "$1" "@smartphone.
        device.ipaddress";
    enforcementAllow := "enforcement.action" "./undrop-client.sh" "$1" "@smartphone.
        device.ipaddress";
}

rule {
    dataLeakage := if conDataLeakDetected do enforcementIsolate;
    noDataLeakage := if conNoDataLeakDetected do enforcementAllow;
    sprChange := if conSplitResults do decreaseSprAction;
    dataLeakageTrust := if conAlertAndNegativeTrustTrend do enforcementIsolate;
}
```

Listing 5.15: Evaluation Policy for detection of sensor sniffing attacks with the use of trust enhancements.

As it is easy to recognise, the policy is rather complex and consists of several anomalies and signatures which are used to form conditions and rules. Details of creating a policy for the CE can be found in [2] and will be omitted for improved readability. There are three particular parts of the policy which are different compared to the non-trust enhanced version. The first part is the context block which defines the used contexts for Features. In addition to the standard context types like DATETIME which defines a certain clock-wise time window and the SLIDING context which defines a moving time window, the TRUSTLEVEL context is used. This context allows to setup certain constraints a Feature must fulfil in terms of trust to be processed by the Correlation Engine. There are two trust-based contexts defined within the test case: one for all infrastructure-based services (e.g. Snort or the PDP) and one for the Correlation Engine (i.e. irondetect) itself. The irondetect context is used to verify that Features which (1) change ratings and (2) trigger Correlation Engine-based cascades are fully trustworthy. That is, these Features are proven to being published by the CE itself. Given that, solely by using this context-based trust definition it is possible to define powerful trust-based rulesets.

The second part is given by the anomalies and their appropriate hints. In addition to the CADS evaluation policy, a trust trend anomaly is defined. This anomaly monitors the trust level of the smartphones Feature Collector and is eventually used to recognise the compromise of it in the test case. More in detail, the trend hint operates solely on the process $RSPR$ of the Feature Collector, making full usage of the underlying TrustLog data model. Furthermore, there are two more anomaly elements which monitor the data received from the smartphone's Feature Collector and the corresponding data received from the Snort IDS. If a difference is detected, a rating change of the smartphone's process SPR is triggered.

Finally, the third part is given by the signatures and the conditions using them. They create a cascade whereas the next step of evaluation is only engaged if high traffic from the smartphone is detected. If this high traffic Feature is detected, the ip address of the device producing the traffic is published as Feature and being used by irondetect to launch the next evaluation step.

Given this policies, the particular test flow can be explained including the results after each step.

Test flow

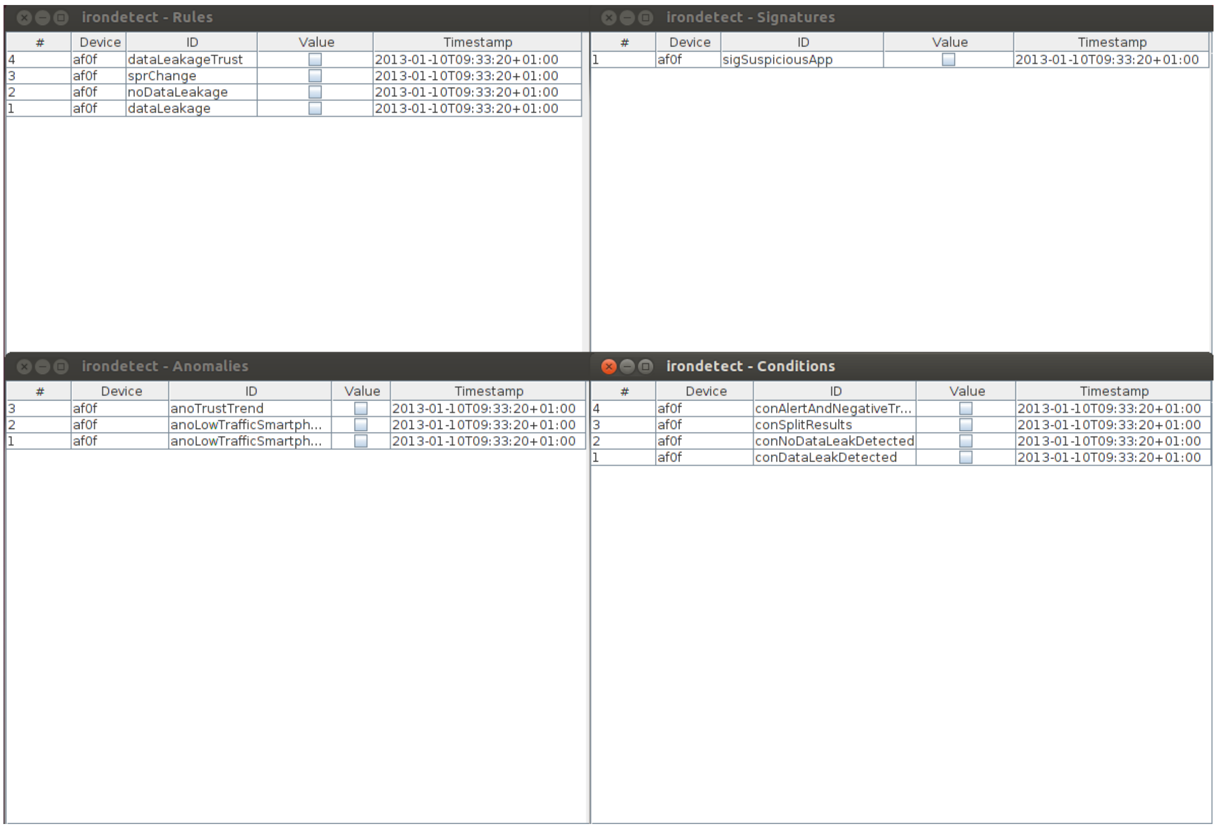
There are four particular steps which are carried out within the test:

1. Bootstrapping the environment, i.e. all required components start up and perform an initial publish to the MAP Server.
2. Start of the compromised Feature Collector on the smartphone. The compromise of this Collector results in it being unable to detect traffic increases on the phone.
3. Start of the webcam app which provides a http-based server on the smartphone.
4. The last step consists of the enforcement procedure after a connection to the smartphone has been established. Due to its complexity, it consists of several sub-steps:
   - Connecting to the webcam app on the smartphone using the http-based server. This results in a huge traffic increase.
   - Detection of this traffic increase only by Snort but not on the smartphone itself. This is due to the compromised Feature Collector.

- Calculation of a trend for the detected traffic, if the trend reaches a threshold, the Correlation Engine lowers the trust level (i.e. the rating of the particular SPR for the process operation on the smartphone) for the smartphone to indicate the different traffic measurements. This is done as the Correlation Engine considers Snort to be fully trustworthy which was checked using Snort's trust context.
- A second trend is calculated by the CE for the trust level change of the smartphone. If the predicted value drops below a threshold, the CE publishes an enforcement Feature to the firewall. The firewall verifies the trust context of this Feature to ensure its validity and blocks all traffic form and to the smartphone.

A detailed explanation of each step, including the particular results on the Correlation Engine is given in the following.

1. The first step consists of bootstrapping the environment and starting the irond-trust as well as the irondetect Correlation Engine. First of all, the irond-trust is started up as it acts as the Feature Provider. Along with it, the appropriate ironmonitor is started. After this, the infrastructure components which are iptables, the PDP and the access point are started up. The smartphone can then be connected to the wireless network which is recognised by the PDP. Due to this, the appropriate access-request which works as root for the smartphone's Feature tree is published by the PDP. Following this, the other infrastructure components are started. In detail, Snort, OpenVAS and as last component the irondetect Correlation Engine is launched. Irondetect immediately starts to evaluate its ruleset-based on the existing Features on the irond-trust. As the smartphone has not yet published a Feature due to the Feature Collector not being started, there are no firing rules. The result of this evaluation can be seen in figure 5.31. If there would be a rule element which was evaluated to be true, this is indicated by a checkmark.
2. The second step consist of starting the Feature Collector on the smartphone. As already stated, this Feature Collector is compromised and only reports the static properties like the installed apps correctly. It does not report dynamic properties, like the traffic generated by the phone correctly, thus it only reports zero as traffic



irondetect - Rules

#	Device	ID	Value	Timestamp
4	af0f	dataLeakageTrust	☐	2013-01-10T09:33:20+01:00
3	af0f	sprChange	☐	2013-01-10T09:33:20+01:00
2	af0f	noDataLeakage	☐	2013-01-10T09:33:20+01:00
1	af0f	dataLeakage	☐	2013-01-10T09:33:20+01:00

irondetect - Signatures

#	Device	ID	Value	Timestamp
1	af0f	sigSuspiciousApp	☐	2013-01-10T09:33:20+01:00

irondetect - Anomalies

#	Device	ID	Value	Timestamp
3	af0f	anoTrustTrend	☐	2013-01-10T09:33:20+01:00
2	af0f	anoLowTrafficSmartph...	☐	2013-01-10T09:33:20+01:00
1	af0f	anoLowTrafficSmartph...	☐	2013-01-10T09:33:20+01:00

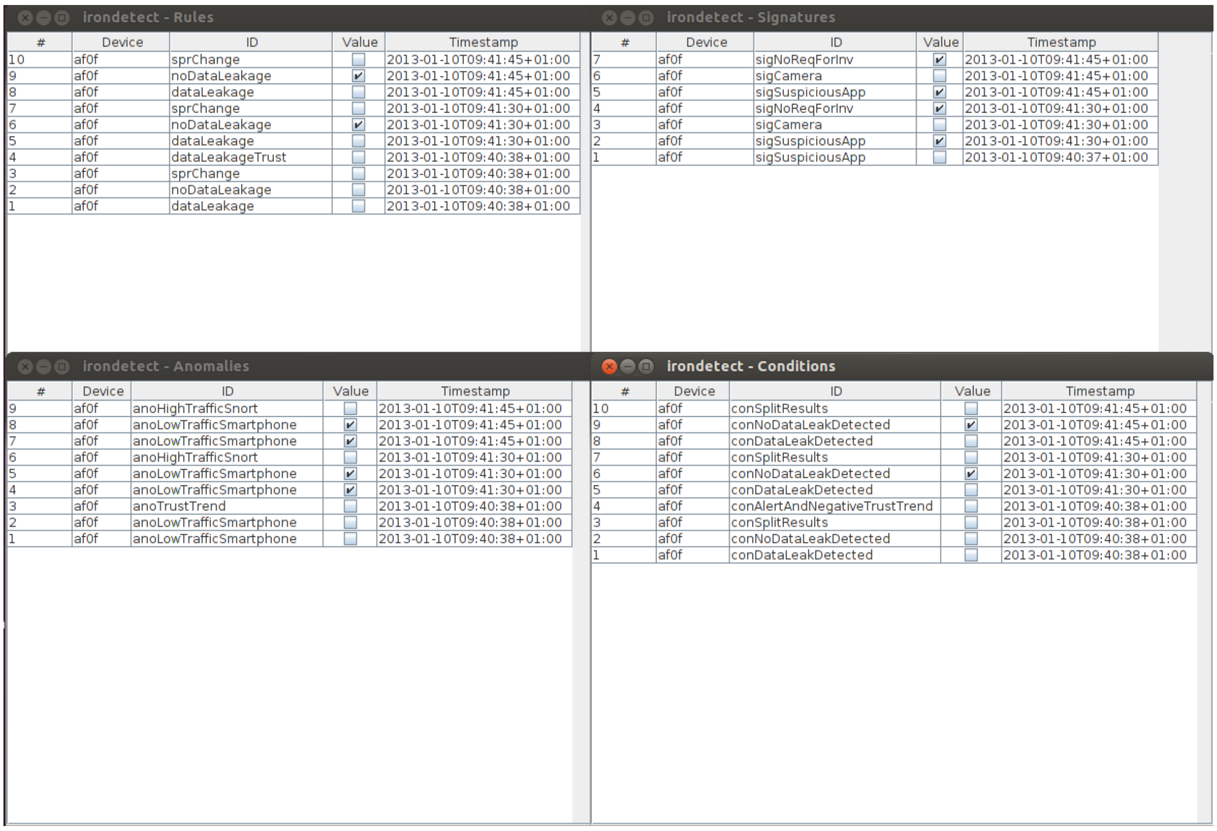
irondetect - Conditions

#	Device	ID	Value	Timestamp
4	af0f	conAlertAndNegativeTr...	☐	2013-01-10T09:33:20+01:00
3	af0f	conSplitResults	☐	2013-01-10T09:33:20+01:00
2	af0f	conNoDataLeakDetected	☐	2013-01-10T09:33:20+01:00
1	af0f	conDataLeakDetected	☐	2013-01-10T09:33:20+01:00

Figure 5.31: Irondetect evaluation after the first step has been finished. No policy elements have been evaluated true.

value. Besides the Feature Collector, the Webcam app is not yet started. This is done in the next step. The result of this step is depicted in figure 5.32. As the picture shows, a suspicious app is detected by irondetect. This is due to the Webcam app possessing permissions (INTERNET, RECEIVE_BOOT_COMPLETE, CAMERA) that are treated as critical. Furthermore, irondetect detects no traffic and due to this, no data leakage from the smartphone.

3. Starting the Webcam app on the smartphones initiates the third step. However, no connection to the provided website is established at this point which results in some of the rules firing but no increase of traffic. As depicted in figure 5.33, irondetect recognises the activation of the camera sensor and the opened port for the website access. In detail, the activated camera is detected by the Feature Collector on the smartphone while the opened port is measured by the OpenVAS vulnerability



irondetect - Rules

#	Device	ID	Value	Timestamp
10	af0f	sprChange	☐	2013-01-10T09:41:45+01:00
9	af0f	noDataLeakage	✓	2013-01-10T09:41:45+01:00
8	af0f	dataLeakage	☐	2013-01-10T09:41:45+01:00
7	af0f	sprChange	☐	2013-01-10T09:41:30+01:00
6	af0f	noDataLeakage	✓	2013-01-10T09:41:30+01:00
5	af0f	dataLeakage	☐	2013-01-10T09:41:30+01:00
4	af0f	dataLeakageTrust	☐	2013-01-10T09:40:38+01:00
3	af0f	sprChange	☐	2013-01-10T09:40:38+01:00
2	af0f	noDataLeakage	☐	2013-01-10T09:40:38+01:00
1	af0f	dataLeakage	☐	2013-01-10T09:40:38+01:00

irondetect - Signatures

#	Device	ID	Value	Timestamp
7	af0f	sigNoReqForInv	✓	2013-01-10T09:41:45+01:00
6	af0f	sigCamera	☐	2013-01-10T09:41:45+01:00
5	af0f	sigSuspiciousApp	✓	2013-01-10T09:41:45+01:00
4	af0f	sigNoReqForInv	✓	2013-01-10T09:41:30+01:00
3	af0f	sigCamera	☐	2013-01-10T09:41:30+01:00
2	af0f	sigSuspiciousApp	✓	2013-01-10T09:41:30+01:00
1	af0f	sigSuspiciousApp	☐	2013-01-10T09:40:37+01:00

irondetect - Anomalies

#	Device	ID	Value	Timestamp
9	af0f	anoHighTrafficSnort	☐	2013-01-10T09:41:45+01:00
8	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:45+01:00
7	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:45+01:00
6	af0f	anoHighTrafficSnort	☐	2013-01-10T09:41:30+01:00
5	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:30+01:00
4	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:30+01:00
3	af0f	anoTrustTrend	☐	2013-01-10T09:40:38+01:00
2	af0f	anoLowTrafficSmartphone	☐	2013-01-10T09:40:38+01:00
1	af0f	anoLowTrafficSmartphone	☐	2013-01-10T09:40:38+01:00

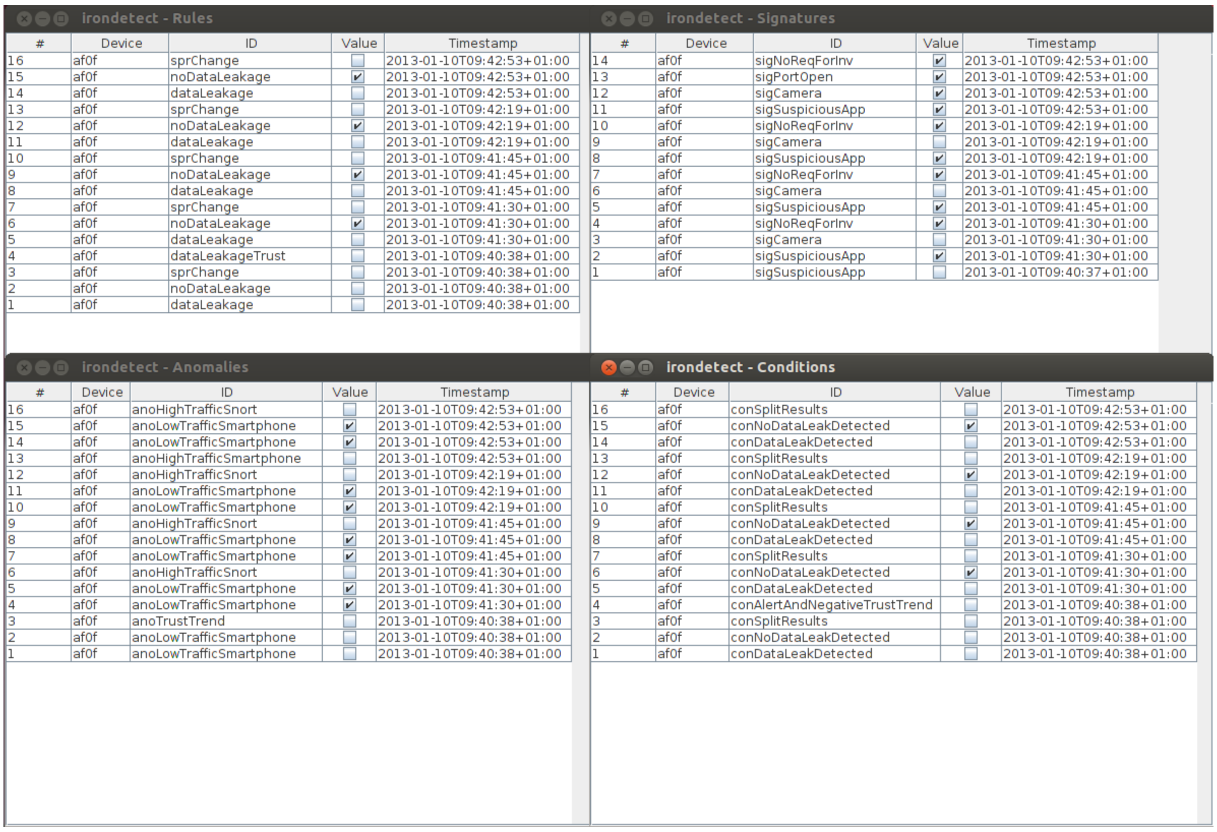
irondetect - Conditions

#	Device	ID	Value	Timestamp
10	af0f	conSplitResults	☐	2013-01-10T09:41:45+01:00
9	af0f	conNoDataLeakDetected	✓	2013-01-10T09:41:45+01:00
8	af0f	conDataLeakDetected	☐	2013-01-10T09:41:45+01:00
7	af0f	conSplitResults	☐	2013-01-10T09:41:30+01:00
6	af0f	conNoDataLeakDetected	✓	2013-01-10T09:41:30+01:00
5	af0f	conDataLeakDetected	☐	2013-01-10T09:41:30+01:00
4	af0f	conAlertAndNegativeTrustTrend	☐	2013-01-10T09:40:38+01:00
3	af0f	conSplitResults	☐	2013-01-10T09:40:38+01:00
2	af0f	conNoDataLeakDetected	☐	2013-01-10T09:40:38+01:00
1	af0f	conDataLeakDetected	☐	2013-01-10T09:40:38+01:00

Figure 5.32: Second step irondetect evaluation results. A checkmark indicates a true evaluation of a certain policy element. In this case, there is no traffic and data leakage but a suspicious app (i.e. the Webcam app) detected.

scanner running periodic scans of the device. As there is no traffic produced yet, there is no anomaly element evaluated true. Finishing this step completes the series of steps which do not result in an enforcement of the device. This is due to the missing traffic. The last step triggering this enforcement is shown next.

4. The Evil VM is now used to connect to the provided web server on the smartphone. Following this, the live streaming of the smartphone's video and audio is started and played back inside the Evil VM's browser. Due to this the traffic generated by the smartphone, in particular in outbound direction, increases drastically. This would be detected by irondetect using the information from the smartphone's Feature Collector under normal conditions. However, as the Feature Collector has been compromised in order to publish false traffic measurements, irondetect is unable to



irondetect - Rules

#	Device	ID	Value	Timestamp
16	af0f	sprChange	☐	2013-01-10T09:42:53+01:00
15	af0f	noDataLeakage	✓	2013-01-10T09:42:53+01:00
14	af0f	dataLeakage	☐	2013-01-10T09:42:53+01:00
13	af0f	sprChange	☐	2013-01-10T09:42:19+01:00
12	af0f	noDataLeakage	✓	2013-01-10T09:42:19+01:00
11	af0f	dataLeakage	☐	2013-01-10T09:42:19+01:00
10	af0f	sprChange	☐	2013-01-10T09:41:45+01:00
9	af0f	noDataLeakage	✓	2013-01-10T09:41:45+01:00
8	af0f	dataLeakage	☐	2013-01-10T09:41:45+01:00
7	af0f	sprChange	☐	2013-01-10T09:41:30+01:00
6	af0f	noDataLeakage	✓	2013-01-10T09:41:30+01:00
5	af0f	dataLeakage	☐	2013-01-10T09:41:30+01:00
4	af0f	dataLeakageTrust	☐	2013-01-10T09:40:38+01:00
3	af0f	sprChange	☐	2013-01-10T09:40:38+01:00
2	af0f	noDataLeakage	☐	2013-01-10T09:40:38+01:00
1	af0f	dataLeakage	☐	2013-01-10T09:40:38+01:00

irondetect - Signatures

#	Device	ID	Value	Timestamp
14	af0f	sigNoReqForInv	✓	2013-01-10T09:42:53+01:00
13	af0f	sigPortOpen	✓	2013-01-10T09:42:53+01:00
12	af0f	sigCamera	✓	2013-01-10T09:42:53+01:00
11	af0f	sigSuspiciousApp	✓	2013-01-10T09:42:53+01:00
10	af0f	sigNoReqForInv	✓	2013-01-10T09:42:19+01:00
9	af0f	sigCamera	☐	2013-01-10T09:42:19+01:00
8	af0f	sigSuspiciousApp	✓	2013-01-10T09:42:19+01:00
7	af0f	sigNoReqForInv	✓	2013-01-10T09:41:45+01:00
6	af0f	sigCamera	☐	2013-01-10T09:41:45+01:00
5	af0f	sigSuspiciousApp	✓	2013-01-10T09:41:45+01:00
4	af0f	sigNoReqForInv	✓	2013-01-10T09:41:30+01:00
3	af0f	sigCamera	☐	2013-01-10T09:41:30+01:00
2	af0f	sigSuspiciousApp	✓	2013-01-10T09:41:30+01:00
1	af0f	sigSuspiciousApp	☐	2013-01-10T09:40:37+01:00

irondetect - Anomalies

#	Device	ID	Value	Timestamp
16	af0f	anoHighTrafficSnort	☐	2013-01-10T09:42:53+01:00
15	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:42:53+01:00
14	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:42:53+01:00
13	af0f	anoHighTrafficSmartphone	☐	2013-01-10T09:42:53+01:00
12	af0f	anoHighTrafficSnort	☐	2013-01-10T09:42:19+01:00
11	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:42:19+01:00
10	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:42:19+01:00
9	af0f	anoHighTrafficSnort	☐	2013-01-10T09:41:45+01:00
8	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:45+01:00
7	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:45+01:00
6	af0f	anoHighTrafficSnort	☐	2013-01-10T09:41:30+01:00
5	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:30+01:00
4	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:41:30+01:00
3	af0f	anoTrustTrend	☐	2013-01-10T09:40:38+01:00
2	af0f	anoLowTrafficSmartphone	☐	2013-01-10T09:40:38+01:00
1	af0f	anoLowTrafficSmartphone	☐	2013-01-10T09:40:38+01:00

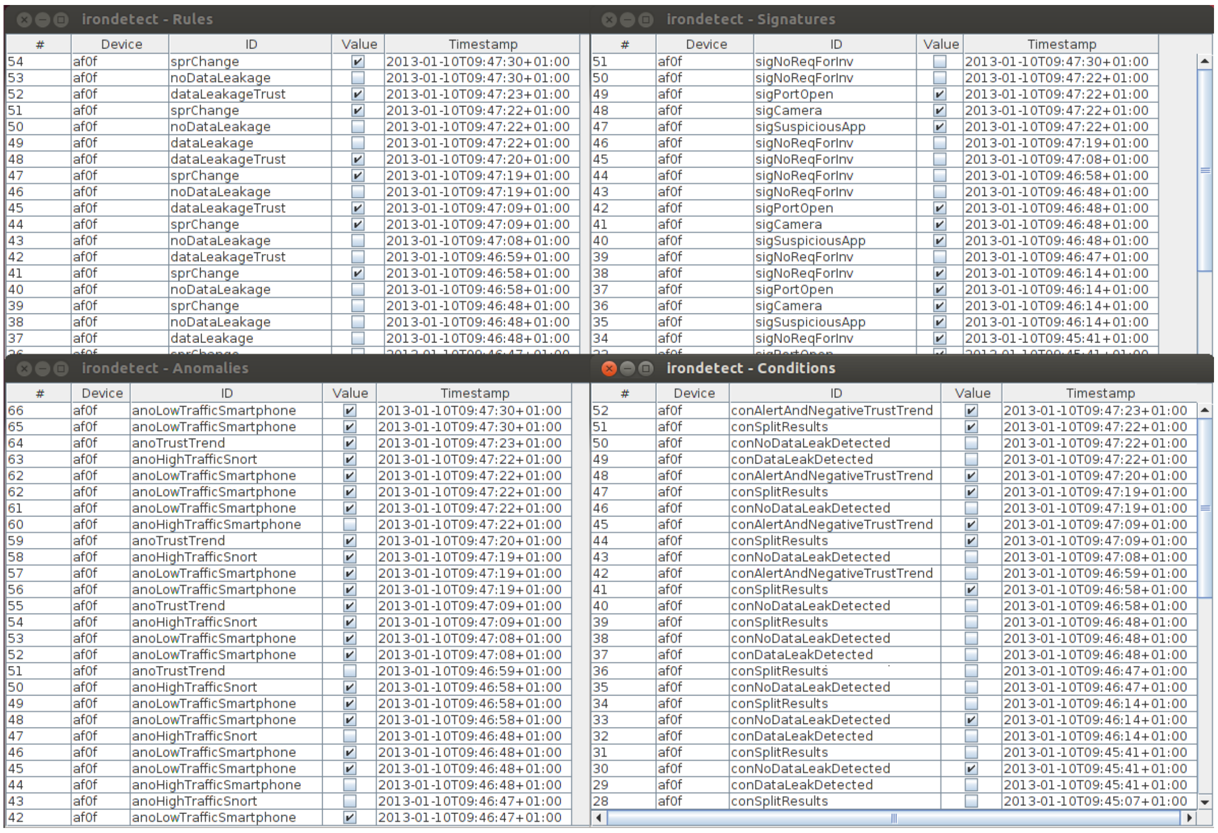
irondetect - Conditions

#	Device	ID	Value	Timestamp
16	af0f	conSplitResults	☐	2013-01-10T09:42:53+01:00
15	af0f	conNoDataLeakDetected	✓	2013-01-10T09:42:53+01:00
14	af0f	conDataLeakDetected	☐	2013-01-10T09:42:53+01:00
13	af0f	conSplitResults	☐	2013-01-10T09:42:19+01:00
12	af0f	conNoDataLeakDetected	✓	2013-01-10T09:42:19+01:00
11	af0f	conDataLeakDetected	☐	2013-01-10T09:42:19+01:00
10	af0f	conSplitResults	☐	2013-01-10T09:41:45+01:00
9	af0f	conNoDataLeakDetected	✓	2013-01-10T09:41:45+01:00
8	af0f	conDataLeakDetected	☐	2013-01-10T09:41:45+01:00
7	af0f	conSplitResults	☐	2013-01-10T09:41:30+01:00
6	af0f	conNoDataLeakDetected	✓	2013-01-10T09:41:30+01:00
5	af0f	conDataLeakDetected	☐	2013-01-10T09:41:30+01:00
4	af0f	conAlertAndNegativeTrustTrend	☐	2013-01-10T09:40:38+01:00
3	af0f	conSplitResults	☐	2013-01-10T09:40:38+01:00
2	af0f	conNoDataLeakDetected	☐	2013-01-10T09:40:38+01:00
1	af0f	conDataLeakDetected	☐	2013-01-10T09:40:38+01:00

Figure 5.33: Evaluation results of the third step. Irondetect recognises the activation of the camera on the device and the opening of the port which is required to access the web server provided by the Webcam app. No traffic increase is measured at this time as there is no connection to the live stream of the app.

detect the traffic increase using this information. As there is the Snort IDS running in the network, there is another way of detecting this increase. Snort monitors all connections for each device within the network, thus it is able to recognise the increase of the smartphone's traffic. Furthermore, these traffic measurements are periodically published as Feature. Due to the fact, that this Feature originates from a infrastructure-based component, it has a rather high trust level which makes it difficult to compromise it. Given this Feature, irondetect uses it for a trend-based traffic evaluation. In detail, a linear regression with a fixed window size is used to estimate the progression of the traffic. If this predicted progression grows over a certain threshold, it is considered to be an abnormal behaviour, thus the anomaly element is being evaluated true. This happens in this step when connecting to the live stream. In addition to that, irondetect recognises the split decision between Snort's measurement and the smartphone's Feature Collector as both are describing the same particular Feature (i.e. traffic of the phone). Due to this, the smartphones Feature Collector is interpreted as suspicious and irondetect reacts with a Feature that lowers the particular rating of the smartphone's process SPR. This Feature is received by the ironmonitor on the Feature Provider which is responsible for carrying out the change. In order to perform this change, the trust level of the Feature instructing the change is verified which makes sure that only irondetect originated Features can be used for such a changing task. Forming a cyclic relationship, the smartphones trust level (i.e. the RSPR of the smartphone's process operation received through the Features published) is also monitored with an anomaly element by irondetect. This anomaly is also trend-based an reacts if the predicted trust level drops below a certain threshold. Note that in detail not the overall trust level is used but the unmapped security level of the process operation. This happens in the test case after some steps and leads eventually to the enforcement of the smartphone which interrupts the live stream. Figure 5.34 shows the situation after the enforcement stage has been reached. On the anomalies side, the low and high traffic difference can be seen easily. This discrepancy is used by the split decision condition which is triggered if both anomalies are evaluated true. This leads to the sprChange signature which triggers the lowering of the smartphone's rating. The trust trend anomaly (anoTrustTrend) reacts if the trust level drops below the threshold which results in the dataLeakageTrust rule to be fired which performs the enforcement.

To explain more in detail, figure 5.35 depicts the particular trend calculations carried out by the anomaly evaluation. The upper two figures (a) and (b) show the calculation results done by the Snort anomaly module. As the calculation is based on several sub steps which may not be time constant, there are always two graphs depicted. The first one shows the calculation results based over the particular calculation steps of the module. The second shows the results over the runtime of the calculation module. As the Snort graph shows, the prediction made by the anomaly module using the linear regression is rather accurate with some small differences. Based on the graph in figure (b), the anomaly becomes evaluated true and triggers the second anomaly module which operates on the SPR of the smartphone and pre-



irondetect - Rules

#	Device	ID	Value	Timestamp
54	af0f	sprChange	✓	2013-01-10T09:47:30+01:00
53	af0f	noDataLeakage	☐	2013-01-10T09:47:30+01:00
52	af0f	dataLeakageTrust	✓	2013-01-10T09:47:23+01:00
51	af0f	sprChange	✓	2013-01-10T09:47:22+01:00
50	af0f	noDataLeakage	☐	2013-01-10T09:47:22+01:00
49	af0f	dataLeakage	☐	2013-01-10T09:47:22+01:00
48	af0f	dataLeakageTrust	✓	2013-01-10T09:47:20+01:00
47	af0f	sprChange	✓	2013-01-10T09:47:19+01:00
46	af0f	noDataLeakage	☐	2013-01-10T09:47:19+01:00
45	af0f	dataLeakageTrust	✓	2013-01-10T09:47:09+01:00
44	af0f	sprChange	✓	2013-01-10T09:47:09+01:00
43	af0f	noDataLeakage	☐	2013-01-10T09:47:08+01:00
42	af0f	dataLeakageTrust	☐	2013-01-10T09:46:59+01:00
41	af0f	sprChange	✓	2013-01-10T09:46:58+01:00
40	af0f	noDataLeakage	☐	2013-01-10T09:46:58+01:00
39	af0f	sprChange	☐	2013-01-10T09:46:48+01:00
38	af0f	noDataLeakage	☐	2013-01-10T09:46:48+01:00
37	af0f	dataLeakage	☐	2013-01-10T09:46:48+01:00

irondetect - Signatures

#	Device	ID	Value	Timestamp
51	af0f	sigNoReqForInv	☐	2013-01-10T09:47:30+01:00
50	af0f	sigNoReqForInv	☐	2013-01-10T09:47:22+01:00
49	af0f	sigPortOpen	✓	2013-01-10T09:47:22+01:00
48	af0f	sigCamera	✓	2013-01-10T09:47:22+01:00
47	af0f	sigSuspiciousApp	✓	2013-01-10T09:47:22+01:00
46	af0f	sigNoReqForInv	☐	2013-01-10T09:47:19+01:00
45	af0f	sigNoReqForInv	☐	2013-01-10T09:47:08+01:00
44	af0f	sigNoReqForInv	☐	2013-01-10T09:46:58+01:00
43	af0f	sigNoReqForInv	☐	2013-01-10T09:46:48+01:00
42	af0f	sigPortOpen	✓	2013-01-10T09:46:48+01:00
41	af0f	sigCamera	✓	2013-01-10T09:46:48+01:00
40	af0f	sigSuspiciousApp	✓	2013-01-10T09:46:48+01:00
39	af0f	sigNoReqForInv	☐	2013-01-10T09:46:47+01:00
38	af0f	sigNoReqForInv	✓	2013-01-10T09:46:14+01:00
37	af0f	sigPortOpen	✓	2013-01-10T09:46:14+01:00
36	af0f	sigCamera	✓	2013-01-10T09:46:14+01:00
35	af0f	sigSuspiciousApp	✓	2013-01-10T09:46:14+01:00
34	af0f	sigNoReqForInv	✓	2013-01-10T09:45:41+01:00

irondetect - Anomalies

#	Device	ID	Value	Timestamp
66	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:30+01:00
65	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:30+01:00
64	af0f	anoTrustTrend	✓	2013-01-10T09:47:23+01:00
63	af0f	anoHighTrafficSnort	✓	2013-01-10T09:47:22+01:00
62	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:22+01:00
62	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:22+01:00
61	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:22+01:00
60	af0f	anoHighTrafficSmartphone	☐	2013-01-10T09:47:22+01:00
59	af0f	anoTrustTrend	✓	2013-01-10T09:47:20+01:00
58	af0f	anoHighTrafficSnort	✓	2013-01-10T09:47:19+01:00
57	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:19+01:00
56	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:19+01:00
55	af0f	anoTrustTrend	✓	2013-01-10T09:47:09+01:00
54	af0f	anoHighTrafficSnort	✓	2013-01-10T09:47:09+01:00
53	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:08+01:00
52	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:47:08+01:00
51	af0f	anoTrustTrend	☐	2013-01-10T09:46:59+01:00
50	af0f	anoHighTrafficSnort	✓	2013-01-10T09:46:58+01:00
49	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:46:58+01:00
48	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:46:58+01:00
47	af0f	anoHighTrafficSnort	☐	2013-01-10T09:46:48+01:00
46	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:46:48+01:00
45	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:46:48+01:00
44	af0f	anoHighTrafficSmartphone	☐	2013-01-10T09:46:48+01:00
43	af0f	anoHighTrafficSnort	☐	2013-01-10T09:46:47+01:00
42	af0f	anoLowTrafficSmartphone	✓	2013-01-10T09:46:47+01:00

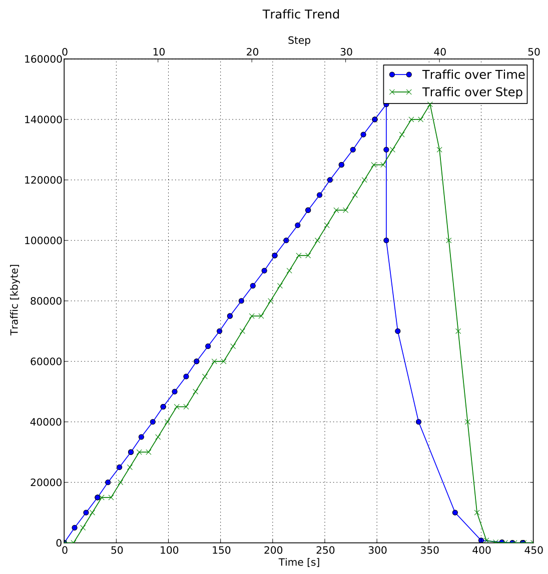
irondetect - Conditions

#	Device	ID	Value	Timestamp
52	af0f	conAlertAndNegativeTrustTrend	✓	2013-01-10T09:47:23+01:00
51	af0f	conSplitResults	✓	2013-01-10T09:47:22+01:00
50	af0f	conNoDataLeakDetected	☐	2013-01-10T09:47:22+01:00
49	af0f	conDataLeakDetected	☐	2013-01-10T09:47:22+01:00
48	af0f	conAlertAndNegativeTrustTrend	✓	2013-01-10T09:47:20+01:00
47	af0f	conSplitResults	✓	2013-01-10T09:47:19+01:00
46	af0f	conNoDataLeakDetected	☐	2013-01-10T09:47:19+01:00
45	af0f	conAlertAndNegativeTrustTrend	✓	2013-01-10T09:47:09+01:00
44	af0f	conSplitResults	✓	2013-01-10T09:47:09+01:00
43	af0f	conNoDataLeakDetected	☐	2013-01-10T09:47:08+01:00
42	af0f	conAlertAndNegativeTrustTrend	☐	2013-01-10T09:46:59+01:00
41	af0f	conSplitResults	✓	2013-01-10T09:46:58+01:00
40	af0f	conNoDataLeakDetected	☐	2013-01-10T09:46:58+01:00
39	af0f	conSplitResults	☐	2013-01-10T09:46:48+01:00
38	af0f	conNoDataLeakDetected	☐	2013-01-10T09:46:48+01:00
37	af0f	conDataLeakDetected	☐	2013-01-10T09:46:48+01:00
36	af0f	conSplitResults	☐	2013-01-10T09:46:47+01:00
35	af0f	conNoDataLeakDetected	☐	2013-01-10T09:46:47+01:00
34	af0f	conSplitResults	☐	2013-01-10T09:46:14+01:00
33	af0f	conNoDataLeakDetected	✓	2013-01-10T09:46:14+01:00
32	af0f	conDataLeakDetected	☐	2013-01-10T09:46:14+01:00
31	af0f	conSplitResults	☐	2013-01-10T09:45:41+01:00
30	af0f	conNoDataLeakDetected	✓	2013-01-10T09:45:41+01:00
29	af0f	conDataLeakDetected	☐	2013-01-10T09:45:41+01:00
28	af0f	conSplitResults	☐	2013-01-10T09:45:07+01:00

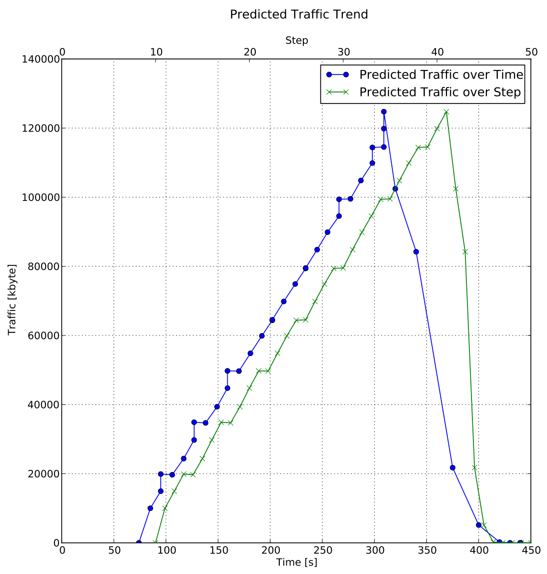
Figure 5.34: Evaluation situation of irondetect after the fourth step. The enforcement of the smartphone has been committed due to the detection of the split decision and the following SPR change of the Feature Collector.

dicts the trend of the rating. The calculation carried out by this module can be seen in figure (c) and (d), with the first being the actual results again and the second being the prediction. The second one is used for the anomaly elements decision, i.e. if the prediction drops below a certain value, which means the rating decreases below a threshold, it returns true as evaluation result.

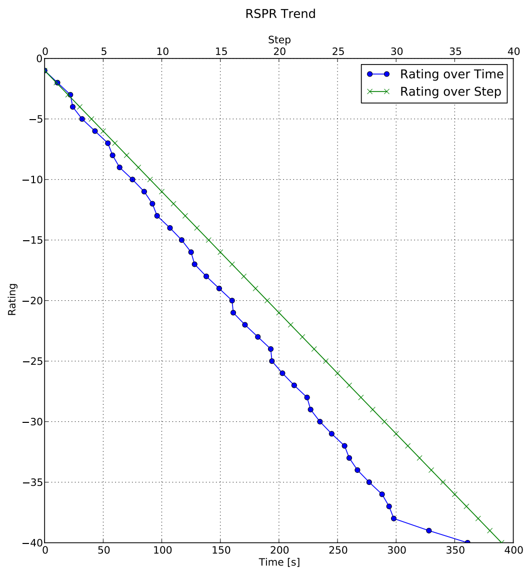
Finishing this test case, the smartphone is no longer able to provide the live stream to the Evil VM or other components. In addition to this, the compromise of the smartphone itself is likely to be recognised as irondetect has lowered the rating of the phone. In a real world use, one or more messages would be provided to the administrator who is then able to take care of the smartphone. Besides that, it is easy to see that the attack has been suppressed very fast and rather effective. Given the nature of the attack, it is unlikely that the short



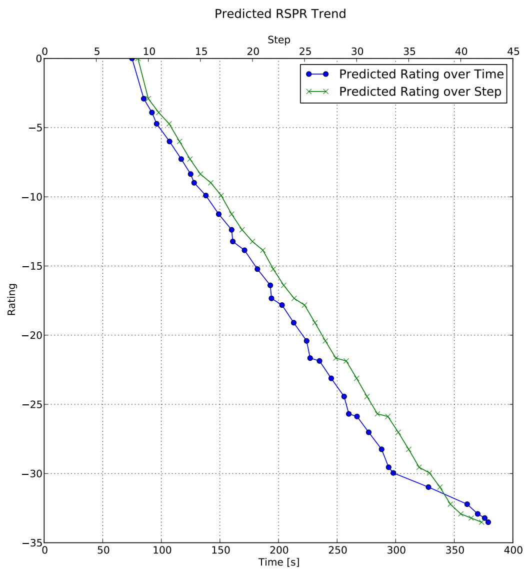
(a)



(b)



(c)



(d)

Figure 5.35: Trend calculations used for both anomaly evaluations.

amount the attacker had access to the phone is enough to gain substantial information. Even more interesting, the system was now able to deal with false information provided by a compromised Feature Collector. This allows for a sophisticated usage profile, where a Feature Collector may not be considered trustworthy at any time.

5.7 Summary

This chapter presented a proof of concept implementation for the conceptual approach. This proof of concept is based on two parts: the conceptual part and the actual implementation. The conceptual part first reviews the IF-MAP protocol of the TCG, which acts as a technological basis for the proof of concept. Instantiating the trust model from the previous chapter, a trust layer for IF-MAP is presented. By combining this layer with the domain-specific extension (TCADS), a particular component-based approach proof of concept is introduced. It consists of a MAPS which is trust-enhanced and acts as a Provider, MAPCs and a trust-aware Correlation Engine, both acting as Senders as well as Receivers. These enhancements not only provide the basic trust concepts but also the snapshot part of the TCADS approach as well as the possibility to express a Feature's trust level by the use of a contextual parameter. Using all parts of the implementation and deploying it within a real world like scenario allowed an overall evaluation of the approach based on the proof of concept. This was done in two ways: First the trust enhanced components (i.e. the irond-trust) have been evaluated while in the second part the overall environment was used to detect a real world like attack.

The component-based evaluation provided two useful results. The first statement which can be made was the acceptable performance. That is, the irond-trust which uses the Trust Token approach can be considered to be as useful as the irond without these modifications. The test results clearly showed, that there is some difference in terms of performance but this difference is small enough to treat it as not of significance. Furthermore, when comparing to other MAP Servers, irond-trust is still to be considered fast. The second result of the components evaluation showed that the Trust Token generation worked as intended. Every possible IF-MAP operation was tested against the irond-trust with all of them providing the expected results. Both of these evaluations showed that the enhanced irond is able to perform as expected and can be used to substitute the non trust irond.

The second part of the evaluation used the complete TCADS system in order to detect and enforce a sensor sniffing attack which compromised the smartphone itself. In detail, the Correlation Engine was not only able to detect the sensor sniffing itself but also the compromise of the smartphone. The successful completion of this complex test case showed clearly the possibilities of the TCADS system. Furthermore, the gearing between the irond-trust and irondetect provides a powerful mechanism for detecting not only signature- and anomaly-based situations but also trust-based attacks. While the policy definition for such cases must be considered rather complex, the possibilities which are provided by the complete TCADS system cover a wide area of use.

Given the requirement R6, which demands the ability to seamless integrate the concept into an already existing environment, the proof of concept showed that this is possible. However, it depends on the particular implementation which is being used. In the case presented here, the implementation is based on protocols and systems which were designed with a high level of interoperability in mind. Due to this, integrability is achieved by simply using the Feature-based IF-MAP data structure and adopting it for every component. Even if is impossible to directly extend a component to use the IF-MAP data structure, the ironmonitor component provides a way of integrating this component as it can be customised as an adapter between a component's internal data structure and the used IF-MAP data structure. That is, if using the proof of concept shown here, requirement R6 can be considered fulfilled also on the level of implementation.

6 Conclusion and Future Work

This chapter concludes the thesis. First, an overall summary of the work which has been done is given. Next, an assessment of the results against the research questions is presented. Finally, an outlook is given. This outlook includes further questions as well as promising directions for future developments.

6.1 Conclusion

This thesis provides two main contributions: (1) a generic trust model for smartphone environments and (2) a domain-specific extension which provides a trust-aware decision making system. Chapter 1 outlined the problems which arise when integrating smartphones into network-based business environments. The situation where smartphones can be administered by the owner of such a network was described, showing that even this case suffers from limitations in terms of integrating smartphones. In addition to that, the chapter described the even more critical approach of bring your own device, where the integration of such external devices becomes more valuable. Continuing this problem description, current examples where given. These examples emphasised the need for a solution. Based on this, approaches which try to give a solution where named and described. However, analysing these approaches in detail, it was pointed out, that none of them considered the trustworthiness of the collected data. As this collected data is used as basis for the decision making process, it was made clear, that more work needs to be done in order to use such systems effectively. Based on these findings, three research questions were identified which are assessed later in this chapter. The second chapter introduced the relevant scenarios for this thesis. All presented scenarios are use cases from the ESUKOM [21] research project. Prior to the detailed scenario description, the reference infrastructure was introduced. Concluding this chapter, a set of requirements that need to be fulfilled in order to solve the named problems has been introduced. Using this set of requirements as assessment criteria, chapter 3 provided an in depth look at the related work that has been done in the relevant research areas. Furthermore, promising technologies addressing some of the named problems were analysed. The reserach-based related work analysis itself consists of trust centric views as well as IDS centric views and a special view on the CADS approach, which can be considered as smartphone-capable IDS. From the technologies point of view, Trusted Computing is one promising approach in order to derive trust for a particular system and was therefore analysed in depth. Following this, the stated requirements were assessed against the findings made and it was pointed out that there is no solution available which fulfils all requirements in a satisfying manner. Due to this, chapter 4 provided a solution for all of the requirements. In detail, the first part of the concept introduced a generic trust model. This trust model gives not only a definition of trust itself, but also provides means to specify, derive and calculate as well as evaluate trust. The trust specification and derivation part is based on the introduction of security properties upon role-based generic operations. These properties can be rated in order to provide a basic trust measurement. Combining all rated properties into one value introduced the trust level which expresses trust per data and not per component, thus addressing the calculation and the evaluation of trust. Furthermore, a data model along with a state machine using this data model was provided by the trust model. The second part of the concept introduced the domain-specific extension, called TCADS, which is based on the generic trust model. More precisely, the generic trust model was used to build a trust-aware decision making system for the particular domain of smartphone environments. Finally, the generic trust model and the domain-specific TCADS system were assessed against the requirements. This assessment showed the great potential as there was only one requirement which could not be fulfilled completely. In particular, this was due to the requirement's fulfilment being very dependent on the implementation. Chapter (5) introduced a proof of concept implementation for the conceptual parts which allowed to show that the requirement can be fulfilled by a tailored implementation. The implementation was based on the CADS reference implementation and the IF-MAP protocol of the TCG. The irond MAP Server was extended providing the defined trust means as so-called Trust Tokens. Furthermore, the data model was placed on the used Correlation Engine, irondetect, and enabled the remaining trust concepts like the history function. Using this implementation, an evaluation was done in the same chapter. First, the used security properties and their appropriate functions were defined. The performance of the enhanced MAP Server, the irond-trust, was evaluated next. This evaluation only showed a minor decrease in processing time, thus being overall successful. The Trust Token generation including every possible operation was evaluated next and showed the expected results. At this point, the irond-trust was able to replace the non trust extended irond. Based on this, the complete TCADS environment was tested within a real world scenario by a sensor sniffing attack. Running through the attack, TCADS was able to prevent the dropping of the sniffed information in a very satisfying manner. As TCADS is based on the generic trust model, the overall approach could be proved not only very promising but also applicable.

Referring to the first chapter, three essential research questions have been stated. These research questions can now be answered which is done in the following.
Which data and characteristics can be used to derive trust? This question is mainly answered by the security property approach. More in detail, the introduced security properties provide a reasonable way of expressing the most basic building block for the overall trust model. They allow to define arbitrary properties which influence the trust derivation for either a platform or a communication channel. By combining them, trust is placed on a Feature rather than on a particular system as they provide a holistic view on the complete environment. This is a very important point as all of the known existing approaches derive trust from only one particular source (or a group of particular sources). Given an example of this, the approach presented in this thesis is also applicable for a scenario with multiple network-based hops. That is, if the Feature is transmitted trough several other systems before being received on the Provider, the generic trust model allows to recognise and evaluate this. In detail this is done by defining appropriate properties for the transmit operation between the Sender and the Provider. These properties may then for example be used to incorporate the trustworthiness of the systems which are in between the Sender and the Provider. It would also be possible, e.g. in the case a secured tunnel is used between Sender and Provider, to simply ignore the intermediate steps of the transmission. The concept of basing trust on particular properties measuring a defined set of operations in terms of there trustworthiness allows for this high level of flexibility. Given this and the abstract model, the approach can be extended to arbitrary scenarios and environments by using the model instantiation defined. As the evaluation showed, security properties and their flexible way of definition is well suitable in a practical case. That is, it could be shown that arbitrary sources for trust derivation may be used, thus

it is possible to tailor them for the actual use case. Given the test case of sensor sniffing, there was for example a property used which expresses the vulnerability level of a system. In the particular case of the smartphone, this level increased drastically when the port for the web server was opened up on the device. That is, the generic trust model provides a very flexible way of deriving trust which can be tailored to fit a wide range of possible scenarios, even more than the scenarios shown here.

How can trust be defined and calculated? In order to answer this question, the thesis first provides a generic definition of trust which is later refined. In detail, this was done by leveraging the trust definition of the TCG and applying it on the trust level concept. The introduction of ratings and the related calculation steps answers the remaining part of this question. As with the security properties, a very flexible method is given to define the particular level of trustworthiness as well as the resulting overall trust level and its calculation. The particular level of trustworthiness, which is represented by the appropriate rating, can be used to express trustworthiness as well as untrustworthiness within the same scenario and environment. This allows for example, that trust values may drop under a certain threshold which may be lower than the initial level of trust. For example, if Features are transmitted using a particular communication library which was considered trustworthy at time of transmission a rating which expresses this trust is given to the transmit operation. If the evaluation of this library changes, e.g. due to a recently discovered bug, the rating may be set to a specific value indicating this problem (i.e., a value that is lower than the initial lowest value). A recalculation of the trust levels for all Features, including the ones which have already been transmitted, affected by this problem would take place. Additionally, the concept provides not only the final trust level for further usage but holds all intermediate steps ready. It is therefore also possible to base a trust evaluation on a certain aspect, for example on a platforms trustworthiness without considering the transmission steps. Due to this, most of the already existing approaches can either be used within solutions that are based on the generic trust model, like TCADS, or completely replaced by the trust model. The evaluation clearly showed, that tailoring these conceptual parts provides a most satisfying way of using the trust model in different scenarios. Along with the calculation, the ratings of the particular properties can also be defined per scenario. Looking back to the evaluation, this allows to have a fine grained distinction between particular properties of different devices.

How is it possible to use trust in the decision making process? As already stated, the trust level itself is used to express the overall trustworthiness. Combined with the domain-specific extension, it therefore answers the third question. In addition to the plain value of the trust level, the intermediate steps are also provided. This allows to only use certain aspects of the trust calculation, thus providing a very high flexibility. Furthermore, the generic trust model provides two more concepts: a flexible data model and a sophisticated state machine using the trust and operational models. The data model allows to actually implement the trust model as it defines the relationship between the different parts of the trust calculation and the Feature the trust is provided for. Along with that, the relationship between the snapshot concept is also defined within the data model. As the name states, the model only defines the entities holding certain data but not the mechanisms operating on this data. This is given by the state machine which combines all provided algorithms into a single overall picture. It consists of three different operational paths, one for each request type. Due to this, the Provider which implements this state machine is fully described. Besides that, the request type which addresses the relationship between the decision making system and the Provider is also given by this state machine as one request type defines the data exchange between these two systems. The domain-specific extension, TCADS, provides a deployable trust-aware decision making system. It contains all necessary functions to use the possibilities of the generic trust model in order to use the trust concept within the decision making process. The extended policy provided for the domain-specific extension addressing the Correlation Engine allows to use the received data from the Provider in terms of trust evaluation. The proof of concept implementation used throughout the evaluation showed, that all parts of the domain-specific TCADS system work geared together, thus providing trust means and a solution for the problems outlined in the introduction.

Given all this, it can be summarised that the generic trust model along with its domain-specific extension, TCADS, provide a novel solution to the problems described. In detail, a very flexible way of defining trust derivation, calculation and usage is introduced which overcomes the limitations described within the state of the art. Furthermore, the trust derivation includes every part of the information handling, thus forming an holistic system which not only addresses small problematic parts. In addition to this, the trust derivation and calculation is given in a dynamic fashion. That is, trust may be changed while the system is running allowing to tweak the trust derivation to the needed level. Besides the calculation, the particular base of trust is defined in a flexible way which allows to use arbitrary sources for trust derivation. This allows to use the system in a wide area of scenarios. Considering the domain-specific extension, TCADS is able to operate and correlate directly upon the trustworthiness including historic changes. Going one step forward, this is not only possible for the defined scenarios but for all scenarios which can be mapped to the generic trust model presented within the concept. The steps necessary to this can be distinguished into two groups: conceptual and implementation. On the conceptual side, there are only the steps of mapping the generic trust model correctly into the environment. If these steps are completed, an implementation must be provided. This is done by either creating an own conceptual domain-specific extension and implementing it or by leveraging the already provided domain-specific extension. Using TCADS, it is necessary to implement the Feature Collector on the appropriate platform along with the TCADS Feature Provider (in particular the state machine) and the Correlation Engine. Doing so allows to use TCADS in arbitrary environments where the role-based mapping of the generic trust model is applicable for. That is, the TCADS system can be used for a wider set of tasks. Providing an trust-based authorisation system or for making decisions within a home automation environment are just two examples. Although these cases are not relevant for this thesis, they show the possibilities of the developed system as it may be used as a generic, trust-enabled decision making engine allowing for sophisticated tasks.

6.2 Future Work

While the current state of the concepts introduced in this thesis can be fully used, with the generic trust model providing general purpose trust means and TCADS providing a complete trust-based decision making system, there are some points which may be investigated further. Namely, there are three points which leave room for improvement. They are described in the following.

Gearing between Feature Provider and Correlation Engine TCADS is based on two core components: the Feature Provider as well as the Correlation Engine. While the Feature Provider is responsible for most parts of the trust calculation and derivation, the Correlation Engine is used as main decision making component. In order to realise sophisticated tasks like the recognition of the sensor sniffing attack presented in the evaluation, they work together in a very fine grained manner. In detail, decision cascades including trust are formed between them. From the architecture's point of view, both of them are still distinct sub systems. This leads to the fact, that the Provider needs to verify the trustworthiness of the Correlation Engine which is rather unnecessary as both are core components required for a functional TCADS environment. Due to this, the gearing between these two core components should be looked at again and if possible improved. An outcome of this could be to form one high level component which provides the same flexibility as the current approach but combines both the Provider and the Correlation Engine into one aggregated engine. This would also reduce the attack potential.

Policy combination and language Looking at the gearing of the Provider and the Correlation Engine, the policies must be considered as well. The current approach uses two different policies, one coming from the generic trust model being the SPM with the properties and ratings while the other being the CE policy of the domain-specific extension defining the ruleset. If combining the Provider and the CE into one single component, the policies should also be integrated to each other. This would allow for a single point of control. Furthermore, the used language should also be looked at. At the moment, both policies rely on self defined languages. Although they are completely functional for the presented cases, they can be improved. This would also allow for an easier integration into environments which rely on certain policy languages and which use a central policy distribution instance.

SP derivation and SPRM definition The last point aims at an easier definition of the used security properties for the generic trust model. Although the thesis provides a small guideline to set such properties, an analysis defining core properties which are found in every environment could be made. Using these core properties, the trust model could be extended by a security property construction kit which allows to assign the properties and their ratings in a very easy fashion. Furthermore, this kit would also allow to generate the required $SPRM$ functions to measure these properties.

Further development of these three points allows for an easier integration of the overall system and would be one step towards the development of a product like system which would be usable for a wide range of tasks. The current version of the generic trust model and the domain-specific TCADS approach provides a very reasonable and promising basis for such an extension.

7 Appendix

7.1 Publications and Contributions

- Trusted Service Access with Dynamic Security Infrastructure Configuration. [182]
- On Remote Attestation for Google Chrome OS. [151]
- TCADS: Trustworthy, Context-related Anomaly Detection for Smartphones (Best Paper Award). [18]
- Trustworthy Anomaly Detection for Smartphones. [19]
- ESUKOM: Smartphone Security for Enterprise Networks. [183]
- Towards permission-based attestation for the Android platform. [149]
- Interoperable device identification in Smart-Grid environments. [184]
- Towards Trustworthy Networks with Open Source Software. [113]
- Interoperable Remote Attestation for VPN Environments. [135]
- tNAC - Trusted Network Access Control. [126]
- Countering Phishing with TPM-bound Credentials. [185]
- Privacy enhanced Trusted Network Connect. [120]

7.2 Trusted Platform

As defined by the TCG, a Trusted Platform is a system, which fulfils the necessary requirements. There are three of these requirements, namely providing so-called protected capabilities, attestation and integrity measurement, logging and reporting. The basic architecture of a system fulfilling these requirements is shown in figure 7.1. The three re-

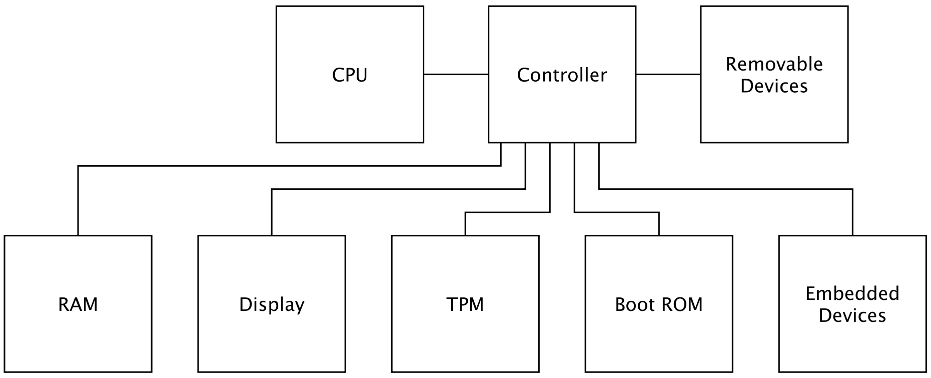


Figure 7.1: Architecture of a Trusted Platform [106].

quirements are mainly fulfilled by the abilities the TPM provides to the system. They are explained in short in the following.

Protected Capabilities The Trusted Platform provides a set of well defined operations, which realise the most security critical tasks. These operations, called protected capabilities, are well secured against compromise. They are for example used when accessing critical locations in memory or when storing a certain measurement, i.e. a hash-based, value. The use of TPM in order to perform a quotation of a PCR's value is for example carried out by invoking such a protected capability.

Attestation Attestation describes the process of attesting information in order to verify their authenticity. As already stated, this is done by using the TPM's key hierarchy which allows it to securely determine if the information set has been signed by a valid TPM. Furthermore, the attestation mechanism is used to verify the integrity of a system. This is done by quoting the TPM's PCRs and singing them using an appropriate key. This signed message is then provided to a verifier which can (a) check the integrity itself and (b) check the authenticity which should indicate that the measurement comes from a valid TPM. The verifier may also be a remote party which receives this message, i.e. the integrity report, via network-based communication. In this case, a remote attestation is performed which is being discussed in detail later in this section.

Integrity Measurement, Logging and Reporting To allow for a verification of a system's integrity, the trusted platform provides appropriate means. First of all, it allows to actually perform the measurement in a secure way by utilising the TPM's capability. In addition to this measurement, the platform records the single steps which lead to the final measurement value, i.e. the final PCR value. Although it is not defined that this logging has to be to performed in a secure way, it is possible to compare the steps against the actual values of the measurement, if there is any difference the integrity of the system is no longer given. Furthermore, a possibility to report the results is provided by such a platform. This allows to use the results and to perform comparison against well known values or against the expected log.

A Trusted Platform is required (i.e. must) to fulfil all three expectations. That is, it needs to provide technical means to perform the desired operations. This cannot be achieved by only incorporating specialised hardware like the TPM, it must also be software-based. This is realised through several, trusted platform tailored extensions. The first one is given by a specialised BIOS which supports the use of the trusted platform-specific abilities, like performing a measurement by using the TPM. Second, the boot system and all following components must support the trusted platform paradigm. While this may be easy for the limited code base of a boot loader, it grows very complex for the operating system itself. Due to this, the use of a specialised operating system, commonly called a trusted operating system (trusted OS) is required. This trusted OS provides means to access the trusted platform's capabilities, like accessing the TPM for a sealing operation. The trusted OS is also responsible for measuring high level applications and for actually starting the integrity verification by invoking the quotation of a TPM's PCR.

Given all that, it is possible to establish a so-called Chain of Trust, which is another basic building block of trusted computing.

7.2.1 Chain of Trust

The Chain of Trust concept allows to put trust in high level measurements, for example of measurements taken from user space applications. Given this concept, it can be considered as trust calculation method, thus being relevant for the fulfilment of R1. This is done by measuring each single software component of the system prior to the components starting process. A simplified version of this process is depicted in figure 7.2. As

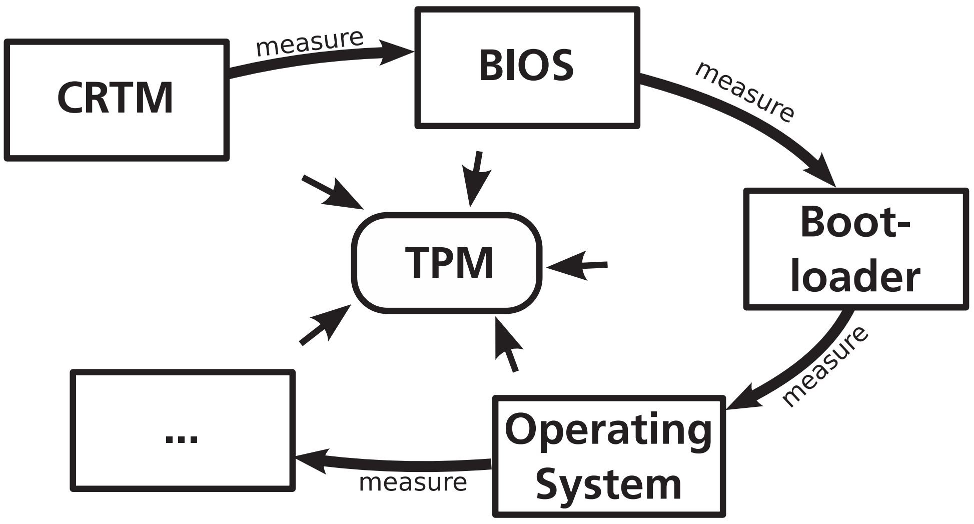


Figure 7.2: Chain of Trust [113].

already explained, the first measurement is triggered by the CRTM, which is residing in a hardware-based component. The CRTM consists of executable code which instructs the very first measurement. Before the measurement actually takes places, the TPM's PCRs are initialised into a well known state. I.e., the initial values are known and must differ from the specifications. The code of the CRTM measures the system's BIOS and extends these values into the appropriate PCRs. This is done before the BIOS is actually started to recognise changes that may have been made to the BIOS itself. After this step, the control is handed over to the BIOS which starts up and is responsible for performing the next measurement. The next component which is to be started is represented by the boot loader. Due to this, the BIOS measures the boot loader and afterwards hands over control to the boot loader by executing it. The boot loader can now measure the operating system before actually starting it. If this is done, the operating system starts and, as long as it provides the capabilities of a trusted OS, further measurements of high level applications can be performed. By reaching this point, a chain of measurements has been created beginning with the CRTM and ending on the last measured application. The trust implication which is behind this concept is that if each of the measurements taken is as expected i.e., the values are the same as the reference values, all measured components are trustworthy, thus a transitive trust relationship has been established. To verify the measurements taken, there need to be reference values for each of the measured components, which might be rather complex due to the high amount of possible components and their trustworthy states. Furthermore, this type of Chain of Trust can only be reset by a full system reboot. It is therefore also based on an approach called Static Root of Trust for Measurement (SRTM) as it creates a Static Chain of Trust.

To overcome this problem of high complexity and fully rebooting the system for a reset, there is an enhanced conceptual version of the Chain of Trust which is based on the so-called Dynamic Root of Trust for Measurement (DRTM). The DRTM approach allows to establish arbitrary new Chain of Trust instances at runtime without the need to reboot the system. It is another hardware-based concept as it must be supported by the CPU. This is due to the fact, that the CPU is responsible for providing an isolated environment in which the Chain of Trust can be established. Intel, with their Trusted Execution Technology (TXT, [186]), as well as AMD with their Secure Virtual Machine Architecture (SVM, [187]) provide such environments on chip. These isolated environments allow it to establish a Chain of Trust which can be used to attest particular aspects of the system, without the need to measure the whole system. This may be used to only spawn a Chain of Trust if necessary for an application, e.g. for carrying out home banking tasks.

If deriving trust, it needs to be taken into account which one of the two types was used to establish the actual Chain of Trust. While dynamic allows the definition of a particular trustworthy environment on a system, static allows to derive the overall trustworthiness of the system. Although the second option seems to be more promising, it is rather questionable if it is possible to establish a complete static Chain of Trust in a real world scenario. This is due to the high complexity.

7.3 Detailed Architecture of TNC

The Network Access Layer is responsible for realising the basic network-based communication. The communication may be realised by utilising different technologies, for example directly through 802.1X or indirect through a VPN tunnel. There are three building blocks located in this layer, one in each component. On the AR's side, there is the logical block of the Network Access Requestor, which is usually software responsible for handling the basic network connection without TNC-specific properties. The Policy Enforcement Point consists only of this block as it only operates on this layer. This is due to the PEP being only responsible for allowing or denying access on a network-based level. As third, the Network Access Authority is residing on the PDP's side. In contrast to the AR, it is not only responsible for providing basic network-based communication but is also responsible to include third party decisions into the overall access decision for an AR. For example, the step of an user authentication is evaluated at this point as it is not TNC-specific. There are two interfaces defined at this layer by the TNC architecture: IF-T for the network-based communication and IF-PEP for the enforcement decisions the PEP has to carry out. IF-T is defined in two different specifications. The first one [188], realises the communication-based on EAP and 802.1X. It is mainly used when directly accessing a network and leveraging port-based access control. It relies on the mechanisms provided by the combination of 802.1X and EAP as there is no ip-based connection available for communicating with the PDP. The second specification defined in [136] allows to base the communication on an already existing ip-based network connection. It uses this connection and establishes a TLS secured tunnel. It is mainly useful in VPN environments where the AR has already a connection to the VPN gateway. Decisions made by the PDP are communicated to the PEP by leveraging the RADIUS-based IF-PEP [189] interface. This is done by using the protocols ability to control the access state of the PEP. All PEPs which are used must be able to understand this protocol to enforce the access decisions.

In order to realise the TNC-specific communication, the Integrity Evaluation Layer is used. This layer provides means to negotiate the results of TNC-based measurements made on the AR. It consists of two components: the TNC Client (TNCC) on the AR and the TNC Server (TNCS) on the PDP. The TNC Client is responsible for collecting and encapsulating the measurements made on the AR. This is done by receiving the measurements through a well defined interface from the layer located above. The encapsulation is done by leveraging the IF-TNCCS interface. This interface defines the actual communication between the TNC Client and the TNC Server. There are two versions of the protocol defining the interface. The first version, which is specified in [190] uses a type length value (TLV) based encoding in it's actual revision. Legacy versions of this specification were based on the use of XML encoding. The second version [191], is used in order to provide compatibility with Microsoft's Network Access Protection (NAP, cf. [192]) concept. That is, it is based on the exchange of so-called Statement of Health reports and allows to use a TNC Client in conjunction with a Microsoft NAP server. After the results, i.e. the measurements, have reached the server, they are provided to the layer above where an in detail evaluation takes place. The result of this evaluation is given back to the TNC Server, which is responsible for providing the final result to the NAA below. The final result may not solely be based on the evaluation of the highest layer but also on a policy which is located on the TNC Server. Furthermore, the result is not only given to the NAA for enforcement purposes but also provided to the TNC Client. The interfaces to the Network Access Layer are not specified. In particular, this is due to the fact, that the communication at this level is very platform-specific and may not be specified. In difference, the interfaces to the upper layer are well specified to allow for a high level of interoperability.

The upper layer, the Integrity Measurement Layer, consists of the Integrity Measurement Collectors (IMCs) on the AR's side and the Integrity Measurement Verifiers on the PDP. The interface between these two components and the layer below is specified, as already written. These specifications [193, 194, 195, 196] define the communication between the TNC Client and the IMCs on the AR as well as the communication of the TNC Server and the IMVs on the PDP. Thus, they allow to use arbitrary IMC/Vs with arbitrary server or client components. Going back to the Integrity Evaluation Layer, the IMCs on the AR are responsible for actually performing the measurements on the platform. There may be an arbitrary amount of such IMCs as long as there is the same amount of IMVs on the other side. This is due to the fact, that there needs to be a component which is possible to evaluate the measurement, which is the appropriate IMV on the PDP's side. Commonly, IMCs and IMVs are provided as pairs of each other, thus each IMC has a corresponding IMV. Two different types of communications may be carried out between them. The first one is unspecified and can only be used between pairs which belong exactly to each other. This allows to measure arbitrary things and communicate the results in the most suited manner. Although this provides a high flexibility and discretionary, it limits the interoperability. To circumvent this, the second method that may be used is the standard interface IF-M [121]. It provides a standardised message exchange, which allows to combine different IMC/V pairs as long as they implement the IF-M standard.

Besides these three layers, there is an extra component called Platform Trust Service (PTS). It is responsible for realising the access to trusted platform-specific functions, in detail to provide a method for accessing the TPM's capabilities. Although the TNC architecture is usable without the PTS enhancement, there would be no way of leveraging the abilities of trusted computing, thus TNC would only be another NAC approach which is not capable of providing a way to put trust into the measurements made on the client (cf. [197]). The PTS is also a well specified interface [119]. In detail, it is located above the Trusted Software Stack (TSS) which itself is located above the TPM. The TSS is usually provided by the operating system as it is an interface which allows to use the capabilities of the trusted platform and the trusted OS itself. The PTS leverages the functions of the TSS by using its functions and due to this, it exposes this functions in a well defined manner for the TNC architecture. It is usable through the two upper layers. The IMCs may use it to extend their measurements made into the TPM, thus integrating them into the chain of trust. The TNC Client can use them to measure the IMCs prior to their execution, thus including also these components into the chain of trust. Note that the TNC client itself must be measured by the operating system. Doing this, all three parts, the TNC stack itself, the appropriate IMCs and their addressed components are measured securely and are part of the chain of trust. This is the main benefit of using TNC as it allows to perform trustworthy measurements. Trustworthy means, that it can on the one hand distinguished between expected and unexpected measurement values and on the other hand it is possible to recognise compromised measurement due to the chain of trust concept.

Given these architecture and their components, the basic message flow which is carried out when negotiating the integrity of a client can be constructed. The simplified flow is shown in figure 7.3 with the particular steps described in the following. Note that the actual enforcement communication is not explicitly addressed.

1. The first step starts the handshake. It is initiated from the side of the TNCC and instructs the IMCs on the client (i.e. on the AR) to measure all necessary components. The overall handshake process is also referred to as integrity check handshake (cf. [96]). The IMCs perform the actual measurements.

2. Their results are communicated back to the TNCC in the second step. The TNCC encapsulates the results in the appropriate version of the TNCCS protocol.

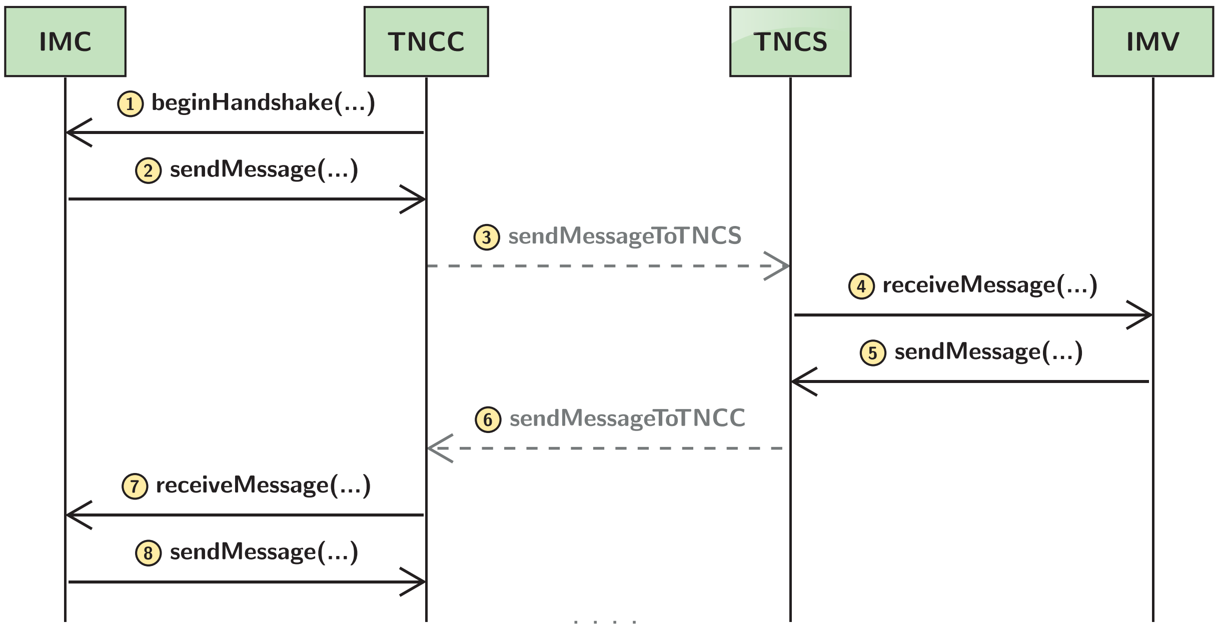


Figure 7.3: Simplified TNC message flow [120].

3. The message which results from this encapsulation is then communicated to the TNCS. This is done by using the layer below, the Network Access Layer which is responsible for performing the actual network-based communication.
4. The TNCS unmarshalls the received message and provides the single parts, which represent the single measurement result per IMC, to the appropriate IMV. The IMVs are then responsible for evaluating the received results.
5. Each of the IMV provides a suggestion to the TNCS. This suggestion includes the evaluation result of the measurements. Based on these suggestion and the combination of them, the TNCS may make a decision about the clients access. In the case shown, a second round is triggered before the final decision is made. This is the case if some information are missing.
6. The TNCS sends a message back to the TNCC. This message includes the request for the missing information, thus this gives the AR the possibility to measure more detailed and provide the missing information.
7. The TNCC receives this message and forwards the measuring request to the appropriate IMCs.

8. After these IMCs have completed their measurement, the measured values are given back to the TNCC which encapsulates them and provides them back to the TNCS. The TNCS uses the evaluation of the IMVs appropriately and may then be able to make a decision.

This message flow may take several rounds to find a real result where the TNCS can base a decision upon. Furthermore, the TNCC measures the IMCs prior to their execution to ensure the chain of trust. The final decision about network access is based on the suggestions of the IMVs, the decision of the TNCS, which may include chain of trust information and on other properties from the NAA. There are several policies controlling this behaviour which are summarised shortly in the following.

7.4 System State Sealed Authentication Certificate

In oder to derive appropriate Security Properties through the SPRM, there must be a method to reason about the trust state of a certain MAP Client. This is done by a special certificate which can be used to determine this trust state of a specific MAP Client. The findings made by the use of this approach can then be used to assign the correct Security Properties for the process operation of this client. It is necessary to provide an approach of this, as the process operation of the Client is the only operation which is not directly visible by the SPRM. This is due to the fact, that all other operations are carried out by or with the MAPS where the SPRM is placed on. While a remote attestation would also be very useful for determining properties of the client's process, the method used here is rather straightforward and provides a complete coverage for the implementation. The approach itself is based on mechanisms provided by Trusted Computing, in particular on sealing data to a certain system state.

7.4.1 Overview

In detail, this method is based on the authentication mechanism provided by IF-MAP. To publish data and to subscribe for certain information, an MAPC has to authenticate itself. This prevents unauthorised access to the MAPS's graph. As IF-MAP uses SOAP and the HTTPS protocol for communication, there are two distinct authentication mechanisms defined: the basic authentication and the certificate-based authentication.

- The basic authentication provides a rather simple authentication method, which uses a username and a password, i.e. a MAPC gets access to the MAPS if it knows the appropriate credentials.
- The second authentication method is based on a certificate. That is, the MAPC has to present a valid (possibly correctly signed by an appropriate authority) certificate to the MAPS. If this certificate is presented and verified, the MAPS grants access to the MAPS.

Based on this two authentication methods, the method proposed here works the following way. The authentication certificate is issued to each MAPC in the network by the MAPS. The certificate is not simply provided but only distributed to the MAPC in a defined way and under the precondition of a valid integrity state. This means, the certificate is only given to the MAPC if (1) the MAPC's integrity state is valid (this is defined by a third party, e.g. the networks administrator) and (2) the MAPC has a mechanism to bind the certificate to this valid integrity state. This ensures, that the certificate can only be accessed by the MAPC if it is in the same state as when the certificate was provided to it, thus effectively countering unwanted system changes. When the MAPC wants to perform MAP operations, it accesses the certificate and presents it to the MAPS. The MAPS recognises this certificate and gives this information to the SPRM which attaches a TT to the MAPC which presented this certificate. If the system state of this MAPC changes, it can no longer access the certificate. As the certificate for authentication is missing and only the simple authentication is available at this time, it will be recognised by the MAPS. The MAPS can then change the appropriate value of the TT by the use of the SPRM. The exact process, is presented in the following.

7.4.2 Initialisation Phase

Aim of this phase is the attachment of a TT to a MAPC publishing data. To achieve this, certain steps are necessary. Figure 7.4 shows these steps. As already explained, this method uses the concept of indirect trust. Thus, the trustworthiness of the data published by a MAPC is derived from its system/integrity state. Therefore, the first step is the definition of a trustworthy system state of the MAPC. This definition is strongly related to the actual implementation and needs therefore to be set while deploying the system. Usually,

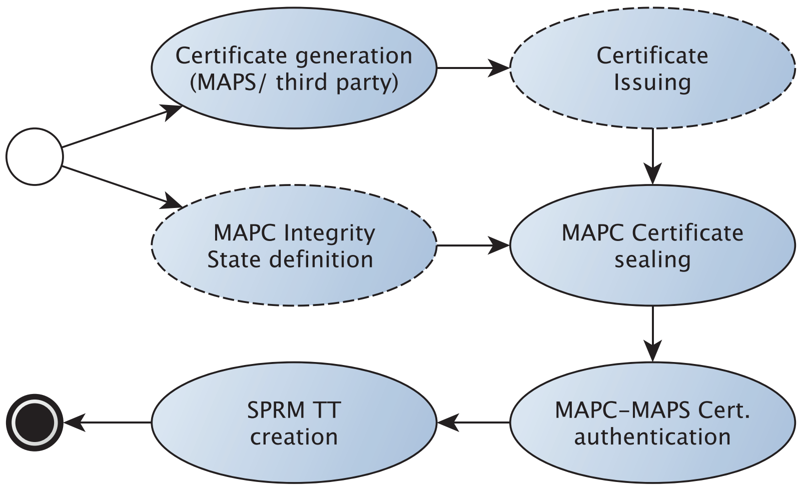

Certificate generation (MAPS/ third party)
Certificate Issuing
MAPC Integrity State definition
MAPC Certificate sealing
SPRM TT creation
MAPC–MAPS Cert. authentication

Figure 7.4: Initialisation of the Certificate.

the state will be defined by a third party (like the networks administrator). It will be chosen in a way, that no compromising element such as malware or a computer virus, is being included. An easy but not complete way of achieving a benign state would be to keep the system always up to date. If such a state is defined, it should be applied to the system. After this is done, the system can be treated as trustworthy for this case and therefore (which is a core element of this method) data published by this system can also be treated trustworthy.

The next step is to issue, provide and bind (seal) the MAP certificate, allowing to authenticate the MAPC. This issuing process is commonly known and therefore not explained in detail here. After the certificate has been issued, it is registered on the MAPS side, thus giving the MAPS the information that (1) the certificate is known and (2) data provided with this certificate can be treated trustworthy. The next task is providing the certificate, which is rather straightforward. The actual implementation should provide a secure way to exchange this certificate. However, the part of binding it to a system's state is more interesting: The binding is done by using special mechanisms which are provided by Trusted Computing. That is, to perform this binding, which is called sealing, the system needs to be capable of performing those operations. This means, that such a system must provide the abilities of a trusted platform. In detail, after the certificate has been received, it will be sealed on to the defined system state. The certificate is therefore taken and encrypted, i.e. it is encrypted with a key stored inside the platforms TPM. This key is not accessible outside of the TPM and one is unable to retrieve this key out of the TPM. Besides this, the usage of this key is bound to certain Platform Configuration Register values. As these PCR values reflect the system's state, the certificate is effectively sealed to the defined state. Having done this, it is not possible to access the certificate if the platforms state has changed compared to the state which was defined as trustworthy.

With this sealed certificate, the MAPC may publish data to the MAPS. The MAPS can distinguish between the authentication type (sealed certificate or simple authentication) and if the connection was authenticated using the certificate, the next step is initiated. This is described in the following.

After the connection is established, and the authentication takes place, the MAPS verifies the certificate. If the certificate is known to the MAPS, access is granted. Otherwise the MAPC is unable to publish data using this authentication type. If the certificate was considered valid by the MAPS, it checks if there is already a TT entry for the metadata published. If there is one, the update phase begins (explained below). If there doesn't exist a TT for the metadata of the MAPC, the MAPS starts the process of creating it. It takes several information which are assigned as Security Properties to do so:

- The identity of the MAPC. This is uniquely identifying the MAPC thus binding the TT to be generated globally to this MAPC.
- Some time information, which may be used for reasoning.
- Trust Anchor information. This is known due to the method used, i.e. it is always the MAPCs TPM when using this method.

The MAPC then creates a new TTM including the above stated SPs in the SPR fields. Along with these values, the Trust Level is put into the TTM. After the TT entities are created, the MAPS publishes them into the MAP graph, thus finishing the initialisation.

7.5 Client Side Policy

The following two figures show the conceptual model of a client side policy and the enhanced architecture used. An example Client-side Policy is depicted in a simplified form in the listing (7.1) shown last.

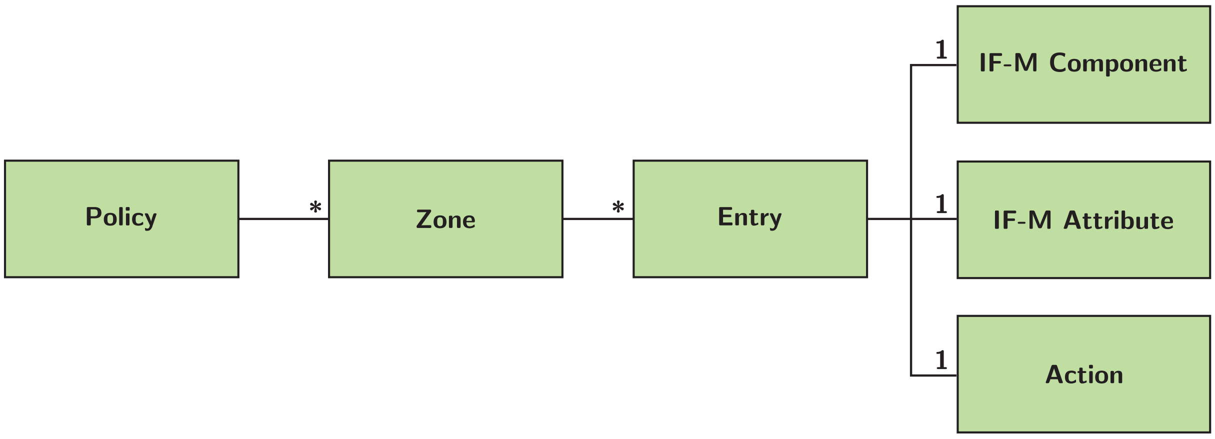


Figure 7.5: Client side policy definition [120].

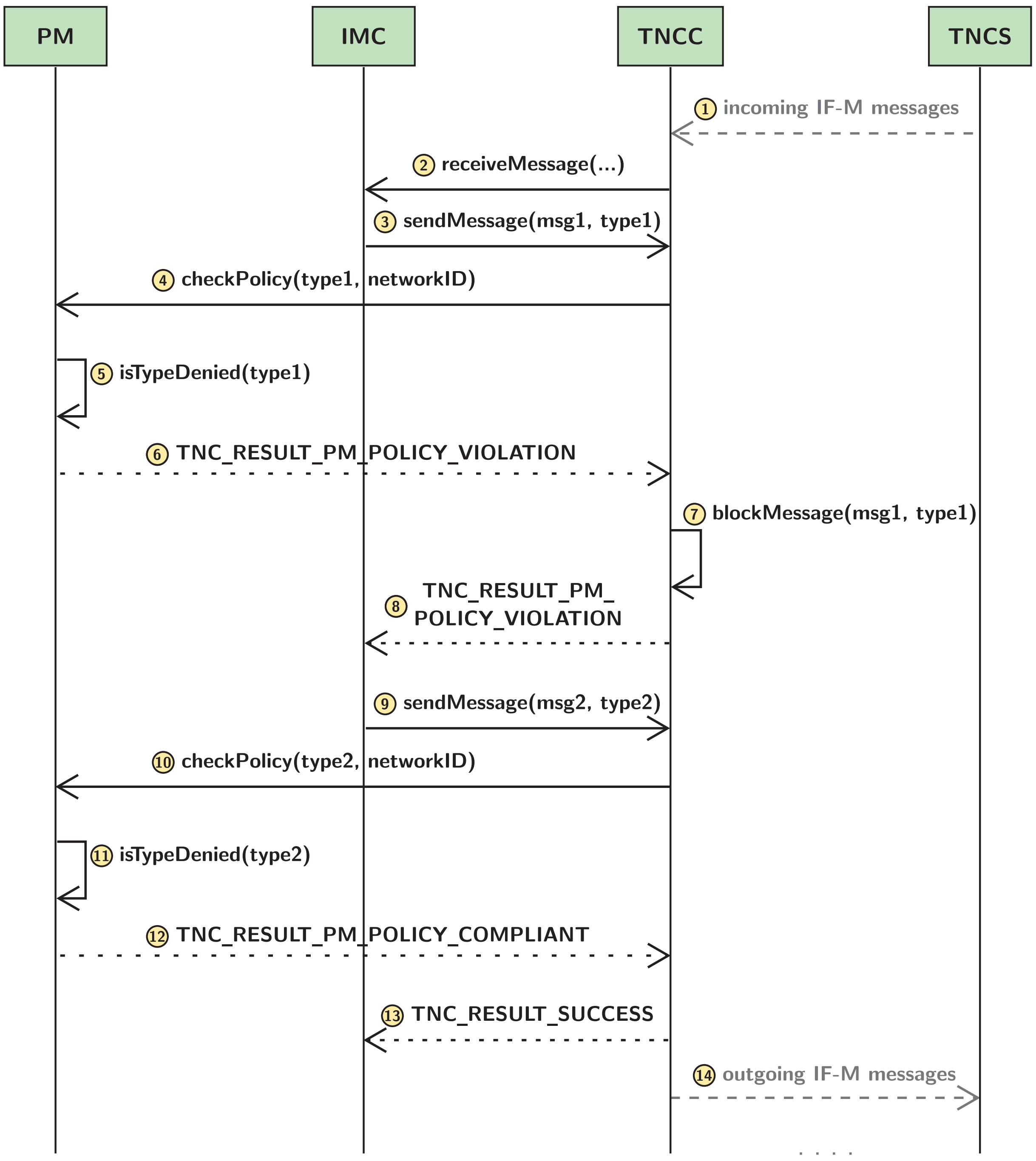

PM
IMC
TNCC
TNCS
1 incoming IF-M messages
2 receiveMessage(...)
3 sendMessage(msg1, type1)
4 checkPolicy(type1, networkID)
5 isTypeDenied(type1)
6 TNC_RESULT_PM_POLICY_VIOLATION
7 blockMessage(msg1, type1)
8 TNC_RESULT_PM_ POLICY_VIOLATION
9 sendMessage(msg2, type2)
10 checkPolicy(type2, networkID)
11 isTypeDenied(type2)
12 TNC_RESULT_PM_POLICY_COMPLIANT
13 TNC_RESULT_SUCCESS
14 outgoing IF-M messages
. . . .

Figure 7.6: Extended TNC message flow [120].

```
<policy>
  <!-- zone for work network -->
  <zone name="work" network="141.71.30.0/23">
  <!-- list of all allowed IF-M messages -->
  <allow>
    <!-- an entry -->
    <entry>
      <!-- list of components -->
      <ifm-component>
        <vendorID>TCG<vendorID>
        <component>Anti Virus<component>
      </ifm-component>
      <ifm-component>
        <vendorID>TCG<vendorID>
        <component>Operating System<component>
      </ifm-component>
      <!-- list of attributes -->
      <ifm-attribute>
        <vendorID>TCG<vendorID>
        <attribute>Product Version</attribute>
      </ifm-attribute>
      <ifm-attribute>
        <vendorID>TCG<vendorID>
        <attribute>String Version</attribute>
      </ifm-attribute>
    </entry>
    <entry>
      <ifm-component>
        <vendorID>TCG<vendorID>
        <component>Firewall<component>
      </ifm-component>
      <ifm-attribute>
        <vendorID>TCG<vendorID>
        <attribute>Operational Status</attribute>
      </ifm-attribute>
    </entry>
  </allow>
  <!-- list of all denied IF-M messages -->
  <deny>
    <entry>
      <ifm-component>
        <vendorID>TCG<vendorID>
        <component>*<component>
      </ifm-component>
      <ifm-attribute>
        <vendorID>TCG<vendorID>
        <attribute>*</attribute>
      </ifm-attribute>
    <entry>
  <deny>
  </zone>
</policy>
```

Listing 7.1: An example Client-side Policy expressed in XML [120]

7.6 Interoperable Remote Attestation in VPN Environments

Detailed architecture including the additionaly introduced components is shown in figure 7.7.

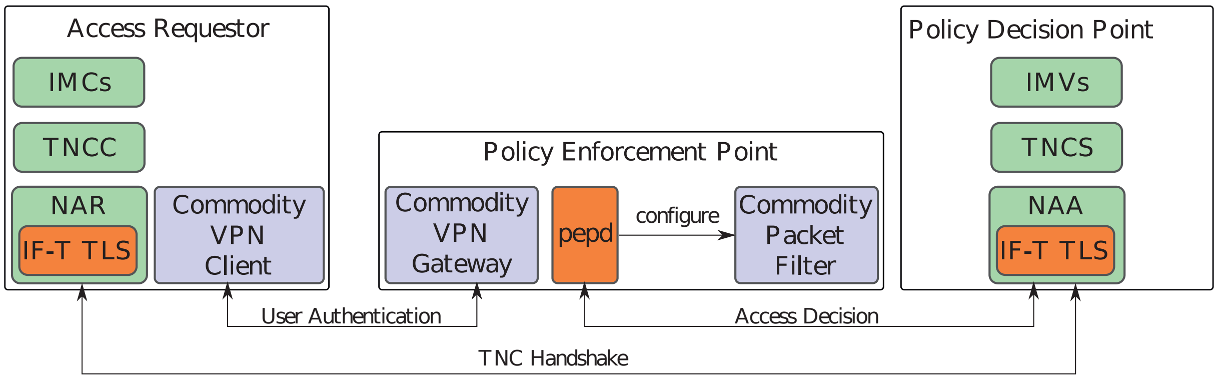


Figure 7.7: Detailed architecture for VPN environments [135].

7.7 Permission-based Attestation

Figure 7.8 and 7.9 depict the complexity of the permission structure in different situations on a HTC Desire device.

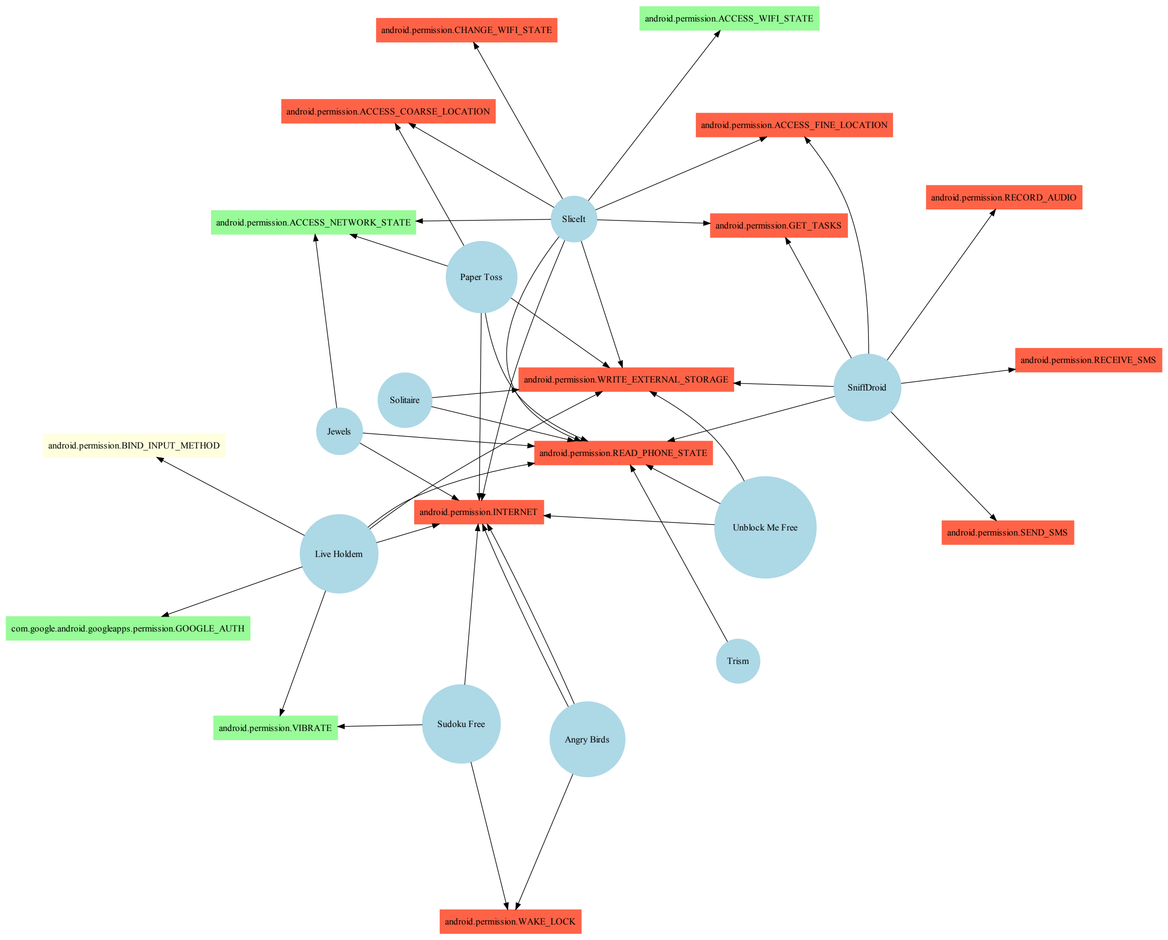

android.permission.CHANGE_WIFI_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.RECORD_AUDIO
android.permission.ACCESS_NETWORK_STATE
SliceIt
android.permission.GET_TASKS
Paper Toss
android.permission.RECEIVE_SMS
android.permission.WRITE_EXTERNAL_STORAGE
SniffDroid
Solitaire
Jewels
android.permission.BIND_INPUT_METHOD
android.permission.READ_PHONE_STATE
android.permission.INTERNET
Unblock Me Free
android.permission.SEND_SMS
Live Holdem
com.google.android.googleapps.permission.GOOGLE_AUTH
Trism
android.permission.VIBRATE
Sudoku Free
Angry Birds
android.permission.WAKE_LOCK

Figure 7.8: Permission graph of a HTC Desire device with top ten free games installed, only app-specific permissions shown.

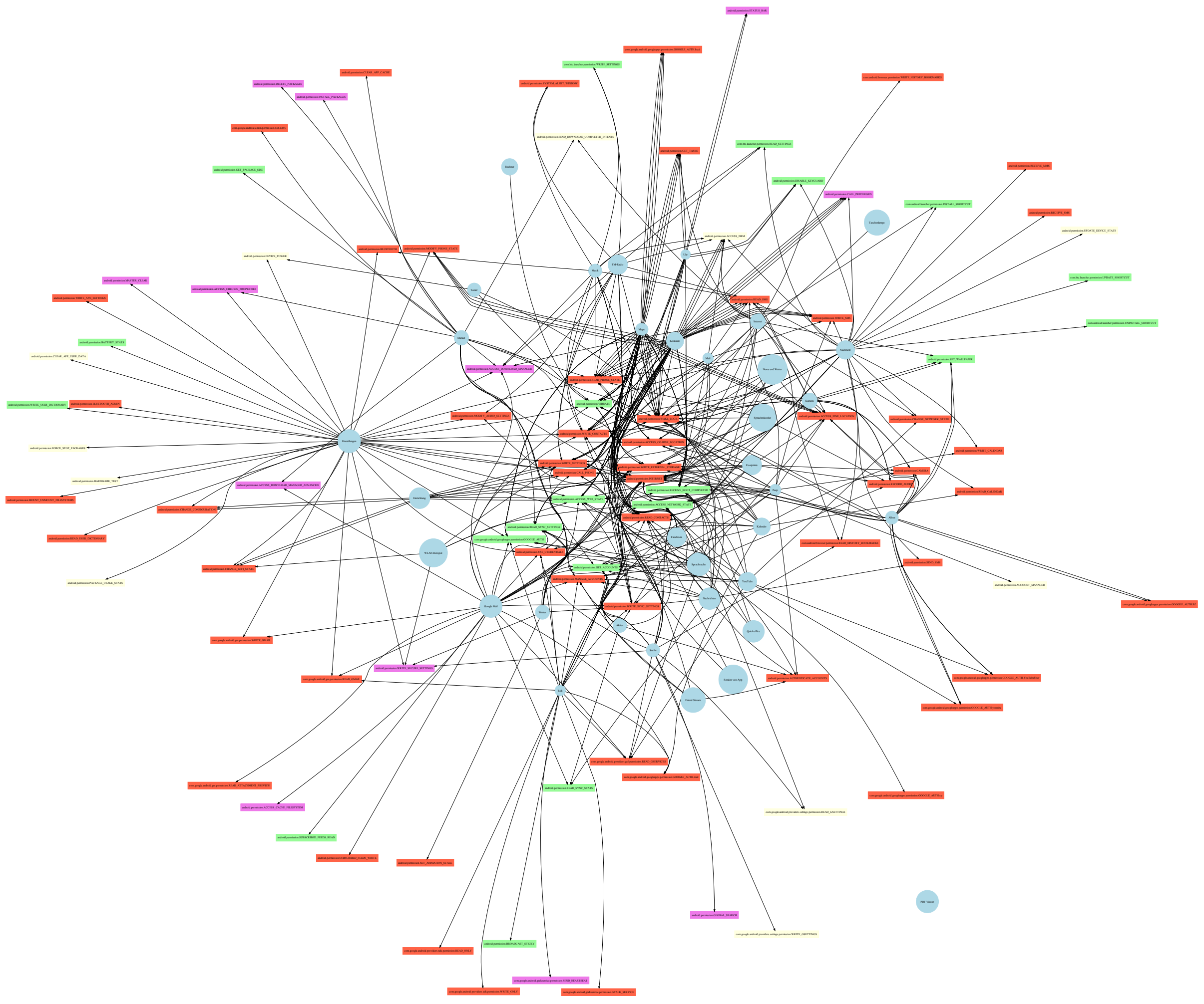


Figure 7.9: Permission graph of the system apps installed on a HTC Desire device.

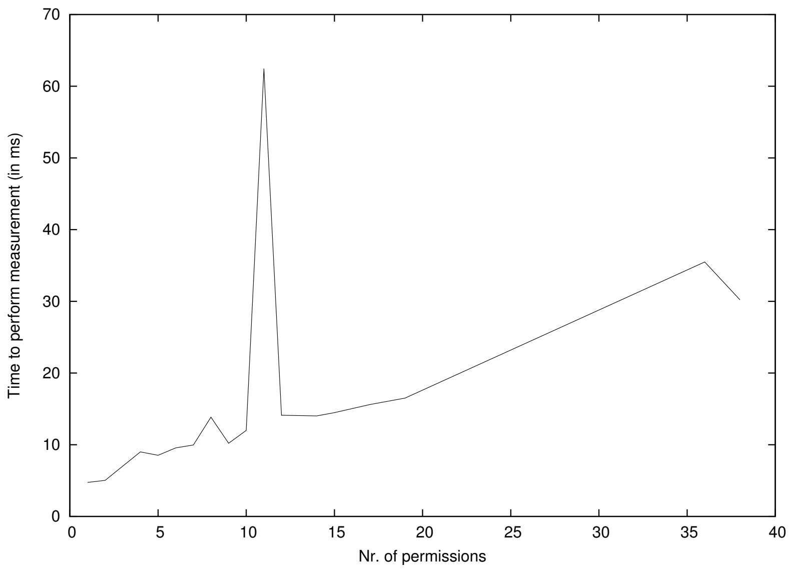


Figure 7.10: Time required to measure permissions [149].

Performance

Performance figures are shown in table 7.1 and figure 7.10.

7.8 On Remote Attestation for Google Chrome OS

The trusted boot process is shown in figure 7.11.

The conceptual architecture of the approach is depicted in figure 7.12.

The flow of operations, based on these extension is depicted in figure 7.13.

Table 7.1: Performance impact of Permission-based measurement of application packages (Abstract) [149]. n_p is the number of permissions the application requests. Time values are given in ms, PML Entry size in Bytes.

		$t_{measure}$			t_{extend}			PML
Application Package	n_p	avg	min	max	avg	min	max	$Size$
com.android.soundrecorder	4	11.54	7	30	7.08	6	37	205
com.android.voicedialer	8	13.84	8	26	7.12	6	14	344
com.android.launcher	11	62.44	52	77	6.78	6	18	465
android	1	6.87	4	14	6.83	6	11	113
com.android.providers.contacts	10	12.82	9	23	6.68	6	49	474
com.android.settings	36	35.50	28	48	6.92	6	38	1464
com.android.quicksearchbox	6	10.48	7	22	6.51	6	11	299
com.android.protips	1	5.9	4	12	6.65	6	39	79
...								
com.android.providers.settings	1	4.63	4	7	7.26	6	47	90
com.android.magicsmoke	2	5.42	4	8	8.17	6	49	146
android.tts	2	4.53	4	9	7.65	6	43	128
com.android.mms	17	15.6	14	23	7.58	6	57	636
com.android.provision	2	3.99	3	6	7.42	6	46	143
com.android.providers.media	4	6.48	5	9	7.7	6	57	223
com.android.certinstaller	1	4.70	4	7	7.58	6	49	113
com.android.providers.downloads	9	9.90	9	18	7.51	6	51	435
com.android.server.vpn	1	4.53	3	8	7.38	6	45	96
...								
com.android.providers.settings	1	4.04	3	6	7.41	6	49	90
android.tts	2	4.21	3	9	8.13	6	47	128
com.android.magicsmoke	2	5.14	4	7	7.64	6	40	146
com.android.mms	17	13.92	13	17	7.77	6	43	636
com.android.provision	2	3.98	3	6	8.94	6	47	143
com.android.providers.media	4	5.97	5	8	7.54	6	48	223
com.android.providers.downloads	9	9.11	8	11	7.62	6	41	435
com.android.certinstaller	1	4.07	3	6	8.46	6	48	113
com.android.server.vpn	1	4.1	3	6	8.7	6	46	96
Sum			463.24		324.89			13828
Average		10.82			7.29			

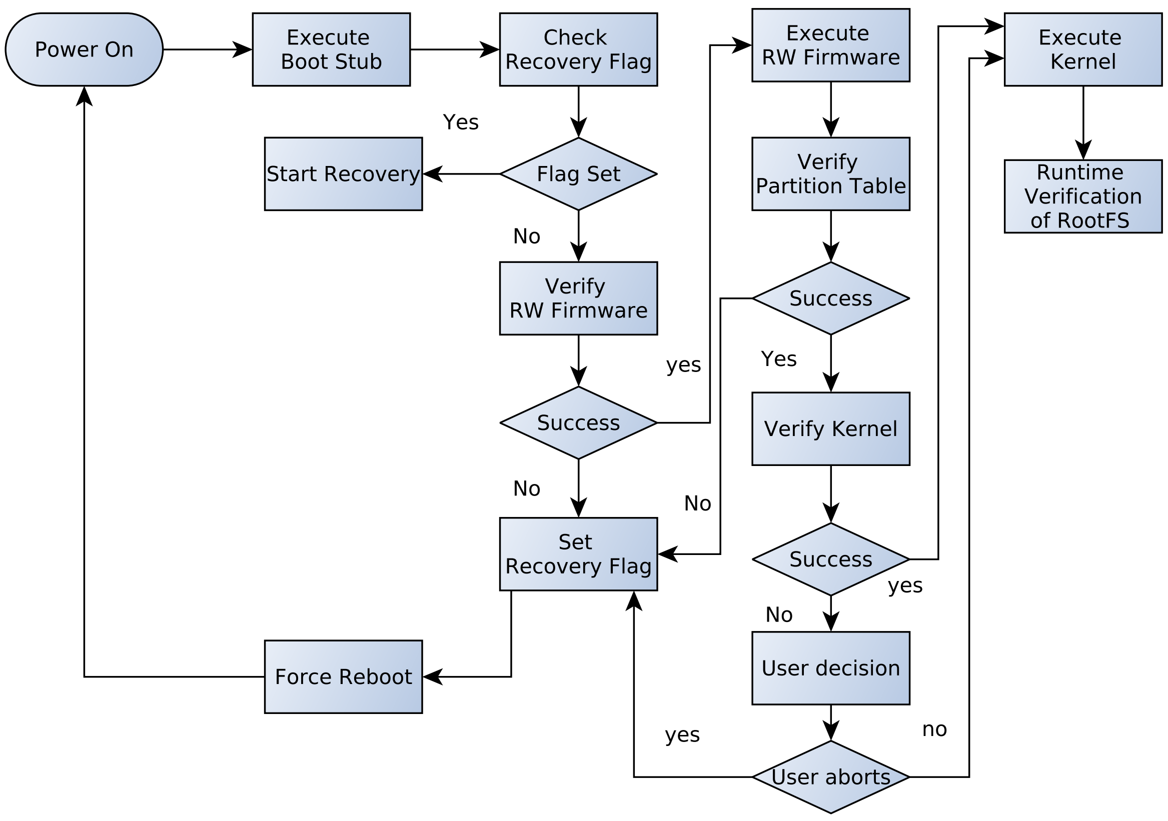


Figure 7.11: Verified boot [151].

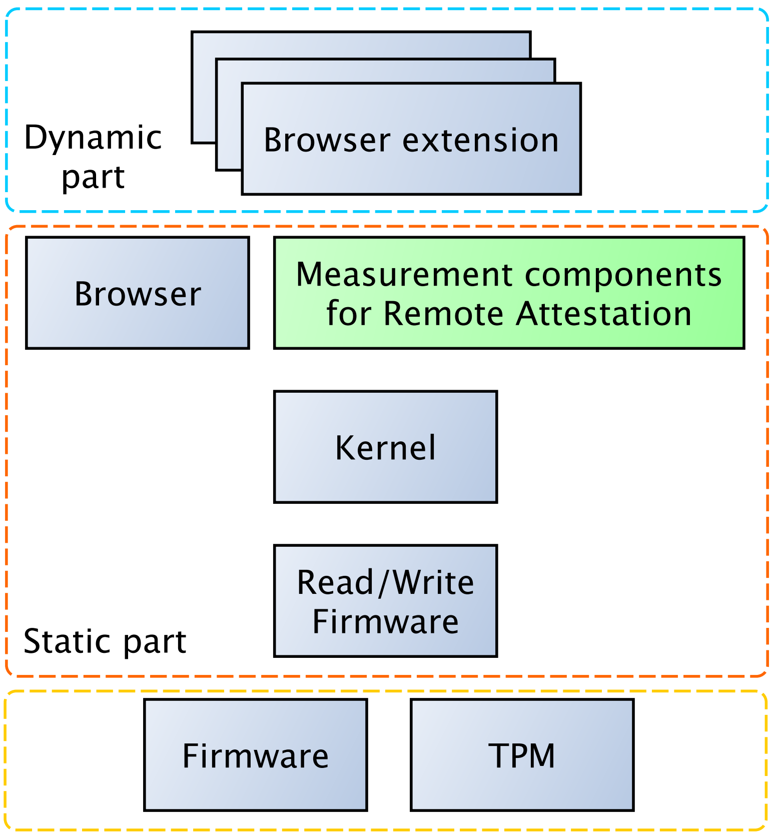


Figure 7.12: Architecture for RA on Chrome OS [151].

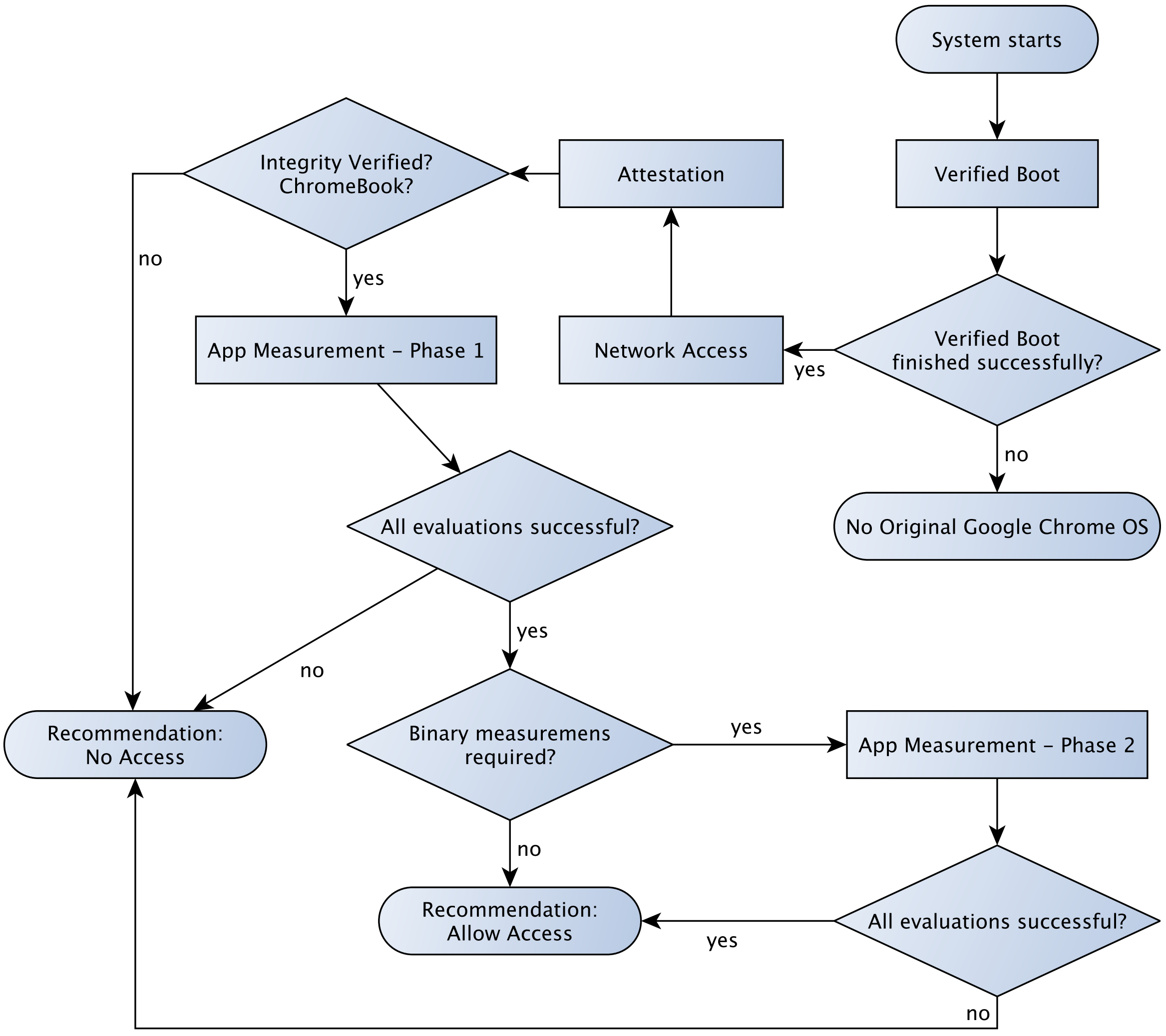

System starts
Verified Boot
Verified Boot finished successfully?
no
No Original Google Chrome OS
yes
Network Access
Attestation
Integrity Verified? ChromeBook?
yes
no
App Measurement – Phase 1
All evaluations successful?
yes
no
Recommendation: No Access
Binary measuremens required?
yes
App Measurement – Phase 2
no
Recommendation: Allow Access
All evaluations successful?
yes
no

Figure 7.13: Flow of operations [151].

7.9 MAP Server Performance Analysis

The following listing provides the source code for the `perf-complete-graph` test program as this was used in a changed form.

```
 /*
 * COPYRIGHT AND PERMISSION NOTICE
 *
 * Copyright (c) 2010-2011, Arne Welzel, <arne.welzel@googlemail.com>
 *
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software for any
    purpose
 * with or without fee is hereby granted, provided that the above copyright
 * notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED AS IS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY
    RIGHTS. IN
 * NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
 * DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
 * OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
    THE USE
 * OR OTHER DEALINGS IN THE SOFTWARE.
 *
 * Except as contained in this notice, the name of a copyright holder shall
    not
 * be used in advertising or otherwise to promote the sale, use or other
    dealings
 * in this Software without prior written authorization of the copyright
    holder.
 */

/*
 * Performance Measurement of a MAPS using a complete graph and some
 * subscriptions, changed to match the trust extended MAPS with doubled
    metdadata
 */
```

```
#include <libifmap2c/ssrc.h>
#include <libifmap2c/arc.h>
#include <libifmap2c/identifiers.h>
#include <libifmap2c/metadata.h>
#include <algorithm>
#include <cstdlib>
#include <iostream>
#include <list>
#include <string>
#include <sstream>
#include <cstdio>
#include "common.h"

using namespace std;
using namespace ifmap2c;

typedef pair<string, string> STRP;
typedef list<SearchResult *> SRLIST;
typedef list<ResultItem *> RILIST;
typedef list<XmlMarshalable *> XMLMLIST;

static unsigned int graphSize;
static int mdNode;
static int mdLink;

static void usage(const char *name)
{
  cerr << "Usage: " << name << " <nodes> <metadata per node> "
    "<metadata per link> " INDEPENDENT_USAGE_STRING << endl;
}

#define for_all_idents(value) for ((value) = 0; (unsigned int)(value) <
    graphSize; (value)++)

static void createSubscriptions(Identifier **idents, SSRC *ssrc)
{
  int i;
  SubscribeRequest *sr;
  list<SubscribeElement *> list;

  for_all_idents(i) {
    stringstream ss;
    ss << "sub_ident_" << i;
    list.push_back(Requests::createSubscribeUpdate(
            ss.str(),
            "meta:role",
            graphSize - 1,
            "meta:role",
            SEARCH_NO_MAX_RESULT_SIZE,
            idents[i]->clone()));
  }

  sr = Requests::createSubscribeReq(list);
  sr->addXmlNamespaceDefinition(TCG_META_NSPAIR);
  ssrc->subscribe(sr);
  delete sr;
}

static void publishMetaOnNodes(Identifier **idents, SSRC *ssrc)
{
  int i;
  PublishRequest *pr;

  if (mdNode <= 0)
    return;

  for_all_idents(i) {
    XMLMLIST list;
    for (int j = 0; j < mdNode; j++) {
      stringstream ss;
      ss << "role_" << j << "_on_identifier_" << i;
      list.push_back(Metadata::createRole(ss.str()));
    }
    pr = Requests::createPublishReq(
        Requests::createPublishUpdate(list, idents[i]->clone()));
    pr->addXmlNamespaceDefinition(TCG_META_NSPAIR);
    ssrc->publish(pr);
    delete pr;
  }
}

static void publishLink(Identifier *i1, Identifier *i2, SSRC *ssrc)
{
  XMLMLIST list;
  PublishRequest *pr;
  XmlMarshalable *md = Metadata::createRole("some role");

  for (int i = 0; i < mdLink; i++)
    list.push_back(md->clone());

  pr = Requests::createPublishReq(Requests::createPublishUpdate(
         list, i1->clone(), i2->clone()));

  pr->addXmlNamespaceDefinition(TCG_META_NSPAIR);
  ssrc->publish(pr);
  delete pr;
  delete md;

}

static int getCountOfMetadata(RILIST rilist)
{
  int ret = 0;
  RILIST::iterator start = rilist.begin();
  RILIST::iterator end = rilist.end();

  for ( ; start != end; start++)
    ret += (*start)->getMetadata().size();

  return ret;
}

static int getCountOfMetadata(SRLIST list)
{
  SRLIST::iterator start = list.begin();
  SRLIST::iterator end = list.end();
  int ret = 0;

  for ( ; start != end; start++)
    ret += getCountOfMetadata((*start)->getResultItems());

  return ret;
}

static int getCountOfMetadata(PollResult *pres)
{
  int ret = 0;
  if (pres->getErrorResults().size() > 0) {
    list<ErrorResult *>::const_iterator it = pres->getErrorResults().begin
        ();
    list<ErrorResult *>::const_iterator end = pres->getErrorResults().end()
        ;
    for (/* */; it != end; it++) {
      cerr << **it << endl;
    }
  }
  ret += getCountOfMetadata(pres->getSearchResults());
  ret += getCountOfMetadata(pres->getUpdateResults());
  ret += getCountOfMetadata(pres->getDeleteResults());
  ret += getCountOfMetadata(pres->getNotifyResults());

  return ret;
}

static void publishCompleteGraph(Identifier **idents, SSRC *ssrc, ARC *arc)
{
  PollResult *pres;
  int i, j, count, expected;
  for_all_idents(i) {
    for_all_idents(j) {
      if (j <= i)
        continue;
      cout << ".";
      cout.flush();

      publishLink(idents[i], idents[j], ssrc);
      pres = arc->poll();

      count = getCountOfMetadata(pres);

      if (i == 0)
        expected = 4 * j * (mdNode + mdLink);
      else
        expected = 2 * graphSize * mdLink;

      if (count != expected) {
        if(count/2 != expected) {
          cerr << "Unexpected metadata count ";
          cerr << "i=" << i << " j=" << j;
          cerr << " count=" << count;
          cerr << " expected=" << expected << endl;
        } else {
          cerr <<''Unexpected metadata count '';
          cerr <<''Use standard performance guest for standard MAPS only ;
          cerr <<''count= '' << count;
          cerr<<'' matches standard MAPS values but not trust value!'' << endl;
        }
        return;
      }


      delete pres;
    }
    cout << endl;
  }
}

static void checkFirstSearchResult(ARC *arc)
{
  PollResult *pres = arc->poll();
  int count;

  if (pres->getSearchResults().size() != graphSize && pres->getSearchResults().size()*2 != graphSize)
```

```
    cerr << "Unexpected SearchResults" << endl;

  count = getCountOfMetadata(pres);

  if (pres->getUpdateResults().size() > 0)
    cout << "Note: MAPS Returns updateResults in first pollResult" << endl;

  if (count != mdNode * (signed int)graphSize) {
    cerr << "Unexpected SearchResults Metadata ";
    cerr << "expected= " << mdNode * (signed int)graphSize;
    cerr << "got=" << count << endl;
  }

  delete pres;
}

int main(int argc, char *argv[])
{
  SSRC *ssrc = NULL;
  ARC *arc = NULL;
  char *url, *user, *pass, *capath;
  int i;
  Identifier **idents;

  checkAndLoadParameters(argc, argv, 3, usage, &url, &user, &pass,
      &capath);

  ssrc = SSRC::createSSRC(url, user, pass, capath);
  arc = ssrc->getARC();

  graphSize = atoi(argv[1]);
  mdNode = atoi(argv[2]);
  mdLink = atoi(argv[3]);

  idents = new Identifier* [graphSize];

  for_all_idents(i) {
    stringstream ss;
    ss << "user_" << i;
    idents[i] = Identifiers::createIdentity(username, ss.str(), user);
  }

  try {
    ssrc->newSession();
    createSubscriptions(idents, ssrc);
    publishMetaOnNodes(idents, ssrc);
    checkFirstSearchResult(arc);

    publishCompleteGraph(idents, ssrc, arc);
    //getchar();

    ssrc->endSession();
  } catch (XmlCommunicationError e) {
    cerr << e << endl;
  } catch (ErrorResult e) {
    cerr << e << endl;
  }

  for_all_idents(i)
    delete idents[i];

  delete idents;
  delete ssrc;
  delete arc;
}
```

Listing 7.2: Source of perf-complete-graph, based on [175].

List of Figures

Glossary

AES Advanced Encryption Standard.

API application programming interface.

AR Access Requestor.

BYOD Bring your own device.

CE Correlation Engine.

DDoS Distributed Denial of Service.

DMZ Demilitarized Zone.

DoS Denial of Service.

FC Feature Collector.

GPS Global Positioning System.

IDS Intrusion Detection System.

IEL Integrity Evaluation Layer.

IMC Integrity Measurement Collector.

IMEI International Mobile Station Equipment Identity.

IML Integrity Measurement Layer.

IMV Integrity Measurement Verifier.

IPS Intrusion Prevention System.

JDK Java Development Kit.

MAP Metadata Access Point.

MAPC MAP Client.

MAPS MAP Server.

NAC Network Access Control.

NAL Network Access Layer.

NAP Network Access Protection.

OS Operating System.

P1TT Phase 1 Trust Token.

P2TT Phase 2 Trust Token.

PDP Policy Decision Point.

PEP Policy Enforcement Point.

PTS Platform Trust Service.

RA Remote Attestation.

$RSPR$ Rated Security Property Record.

SL Security Level.

SP Security Property.

SPM Security Property Map.

SPR Security Property Record.

$SPRM$ Security Property Record Manager.

TCB Trusted Computing Base.

TCG Trusted Computing Group.

TL Trust Level.

TL_{ζ}^{P1} Phase 1 Trust Level.

TL_{ζ}^{P2} Phase 2 Trust Level.

TPM Trusted Platform Module.

TSS TCG Software Stack.

TT Trust Token.

TTM Trust Token Metadata.

VPN Virtual Private Network.

Bibliography

[1] P. Zheng and L. M. Ni, "Spotlight: The rise of the smart phone," *IEEE Distributed Systems Online*, vol. 7, no. 3, pp. 3–, Mar. 2006. [Retrieved: 18-May-2013] http://dx.doi.org/10.1109/MDSO.2006.22

[2] I. Bente, "Towards a network-based approach for smartphone security," Ph.D. dissertation, UniBW Muenchen, July 2013.

[3] M. Becher, "Security of smartphones at the dawn of their ubiquitousness," Ph.D. dissertation, 2009.

[4] Android Webpage. [Retrieved: 18-May-2013] http://www.android.com/

[5] International Data Corporation (IDC), "Android marks fourth anniversary since launch with 75.0% market share in third quarter," Nov. 2012. [Retrieved: 18-May-2013] http://www.idc.com/getdoc.jsp?containerId=prUS23771812

[6] Huffington Post, "Android Market Share Q3 2012: Google's Still Beating Apple, But Will The iPhone 5 Change That?" [Retrieved: 18-May-2013] http://www.huffingtonpost.com/2012/09/18/android-market-share-q3-2012-n-1893292.html

[7] CVE-2011-1823. [Retrieved: 18-May-2013] http://www.cvedetails.com/cve/CVE-2011-1823/

[8] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy, "Privilege escalation attacks on android," *Information Security*, pp. 346–360, 2011. [Retrieved: 18-May-2013] http://www.springerlink.com/index/D275570090NG72JT.pdf

[9] Fabian Scherschel, "Exynos 4 critical security hole affects many Galaxy devices," 2012. [Retrieved: 18-May-2013] http://www.h-online.com/open/news/item/Exynos-4-critical-security-hole-affects-many-Galaxy-devices-1770075.html

[10] R. Felder, C. Banse, C. Krauss, and V. Fusenig, "Android os security: Risks and limitations," May. [Retrieved: 18-May-2013] http://www.aisec.fraunhofer.de/content/dam/aisec/de/pdf/tech%20reports/AISEC-TR-2012-001-Android-OS-Security.pdf

[11] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode, "Rootkits on smart phones: attacks, implications and opportunities," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems &#38; Applications*, ser. HotMobile '10. New York, NY, USA: ACM, 2010, pp. 49–54. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1734583.1734596

[12] W. Shin, S. Kwak, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A small but non-negligible flaw in the android permission scheme," in *Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium on*, july 2010, pp. 107 –110.

[13] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, 2011, pp. 17–33. [Retrieved: 18-May-2013] http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Soundcomber:+A+Stealthy+and+Context-Aware+Sound+Trojan+for+Smartphones#0

[14] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds - MobiHeld '09.* New York, New York, USA: ACM Press, 2009, pp. 31–36. [Retrieved: 18-May-2013] http://portal.acm.org/citation.cfm?doid=1592606.1592614

[15] D. Damopoulos, G. Kambourakis, and S. Gritzalis, "isam: An iphone stealth airborne malware," *Future Challenges in Security and Privacy for Academia and Industry*, pp. 17–28, 2011.

[16] S. Esser, "Exploiting the iOS kernel," *Black Hat USA*, 2011.

[17] A. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices.* ACM, 2011, pp. 3–14.

[18] I. Bente, G. Dreo, B. Hellmann, J. Vieweg, and J. von Helden, "TCADS: Trustworthy, Context-related Anomaly Detection for Smartphones," *Presented at the 15th International Conference on Network-Based Information Systems (NBiS 2012)*, 2012, Best Paper Award.

[19] ——, "Trustworthy Anomaly Detection for Smartphones," *Poster presented at the 13th Workshop on Mobile Computing Systems and Applications (HotMobile 2012)*, 2012.

[20] The White House, "Bring Your Own Device." [Retrieved: 18-May-2013] http://www.whitehouse.gov/digitalgov/bring-your-own-device

[21] ESUKOM Konsortium, "ESUKOM Project Page," 2012. [Retrieved: 18-May-2013] http://www.esukom.de

[22] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.

[23] J. Richter, N. Kuntze, and C. Rudolph, "Security digital evidence," in *Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Fifth IEEE International Workshop on*, may 2010, pp. 119 –130.

[24] N. Kuntze and C. Rudolph, "Secure digital chains of evidence," in *Systematic Approaches to Digital Forensic Engineering (SADFE), 2011 IEEE Sixth International Workshop on*, may 2011, pp. 1 –8.

[25] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Annals of Telecommunications*, vol. 55, pp. 361–378, 2000, 10.1007/BF02994844. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/BF02994844

[26] ——, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/pii/S1389128698000176

[27] T. Verwoerd and R. Hunt, "Intrusion detection techniques and approaches," *Computer Communications*, vol. 25, no. 15, pp. 1356 – 1365, 2002. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/pii/S0140366402000373

[28] S. Axelsson, “Intrusion detection systems: A survey and taxonomy,” Tech. Rep., 2000.

[29] “Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory,” *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/382912.382923

[30] M. Roesch and S. Telecommunications, “Snort - lightweight intrusion detection for networks,” 1999, pp. 229–238.

[31] S. Kumar and E. H. Spafford, “A pattern matching model for misuse intrusion detection.”

[32] R. Erbacher, K. Walker, and D. Frincke, “Intrusion and misuse detection in large-scale systems,” *Computer Graphics and Applications, IEEE*, vol. 22, no. 1, pp. 38 –47, jan/feb 2002.

[33] A. Sundaram, “An introduction to intrusion detection,” *Crossroads*, vol. 2, no. 4, pp. 3–7, Apr. 1996. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/332159.332161

[34] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar, “A comparative study of anomaly detection schemes in network intrusion detection,” in *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

[35] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1541880.1541882

[36] W. Lee and D. Xiang, “Information-theoretic measures for anomaly detection,” in *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, 2001, pp. 130 –143.

[37] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, “A hierarchical anomaly network intrusion detection system using neural network classification,” in *CD-ROM Proceedings of 2001 WSES International Conference on: Neural Networks and Applications (NNA ’01*, 2001.

[38] P. Mell and M. McLarnon, "Mobile agent attack resistant distributed hierarchical intrusion detection systems," in *Purdue University*, 1999.

[39] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo, "Towards collaborative security and p2p intrusion detection," in *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, june 2005, pp. 333 – 339.

[40] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system," in *In Proceedings of Network and Distributed System Security Symposium (NDSS 04)*, 2004.

[41] R. Bidou, J. Bourgeois, and F. Spies, "Towards a global security architecture for intrusion detection and reaction management," in *Information Security Applications*, ser. Lecture Notes in Computer Science, K.-J. Chae and M. Yung, Eds. Springer Berlin / Heidelberg, 2004, vol. 2908, pp. 1769–1785. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-540-24591-9_9

[42] A. K. Ganame, J. Bourgeois, R. Bidou, and F. Spies, "A global security architecture for intrusion detection on computer networks," *Computers and Security*, vol. 27, no. 1–2, pp. 30 – 47, 2008. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/pii/S0167404808000047

[43] R. Ramachandran, S. Neelakantan, and A. Bidyarthy, "Behavior model for detecting data exfiltration in network environment," in *Internet Multimedia Systems Architecture and Application (IMSAA), 2011 IEEE 5th International Conference on*, dec. 2011, pp. 1 –5.

[44] R. Cullingford, "Correlation and collaboration in anomaly detection," in *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, march 2009, pp. 251 –254.

[45] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers and Security*, vol. 29, no. 1, pp. 124 – 140, 2010. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/pii/S016740480900073X

[46] C. Zhou, C. Leckie, S. Karunasekera, and T. Peng, "A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains," in *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, april 2008, pp. 321 –327.

[47] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, 2002, pp. 202 – 215.

[48] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," RFC 4765 (Experimental), Internet Engineering Task Force, March 2007. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc4765.txt

[49] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: a peer-to-peer approach to network intrusion detection and prevention," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, june 2003, pp. 226 – 231.

[50] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, may 1996, pp. 164 –173.

[51] A. Abdul-Rahman, "The pgp trust model," April 1997, eDI-Forum.

[52] A. D. Keromytis, J. Parekh, P. N. Gross, G. Kaiser, V. Misra, J. Nieh, D. Rubenstein, and S. Stolfo, "A holistic approach to service survivability," in *Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems: in association with 10th ACM Conference on Computer and Communications Security*, ser. SSRS '03. New York, NY, USA: ACM, 2003, pp. 11–22. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1036921.1036923

[53] Y. Bai and H. Kobayashi, "Intrusion detection systems: technology and development," in *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*, march 2003, pp. 710 – 715.

[54] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A trust-aware, p2p-based overlay for intrusion detection," in *Database and Expert Systems Applications, 2006. DEXA '06. 17th International Workshop on*, 0-0 2006, pp. 692 –697.

[55] C. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, june 2009, pp. 33 –40.

[56] C. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust management for host-based collaborative intrusion detection," in *Managing Large-Scale Service Deployment*, ser. Lecture Notes in Computer Science, F. Turck, W. Kellerer, and G. Kormentzas, Eds. Springer Berlin Heidelberg, 2008, vol. 5273, pp. 109–122. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-540-87353-2_9

[57] C. Fung, J. Zhang, I. Aib, and R. Boutaba, "Trust management and admission control for host-based collaborative intrusion detection," *Journal of Network and Systems Management*, vol. 19, pp. 257–277, 2011, 10.1007/s10922-010-9176-7. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/s10922-010-9176-7

[58] ——, "Dirichlet-based trust management for effective collaborative intrusion detection networks," *Network and Service Management, IEEE Transactions on*, vol. 8, no. 2, pp. 79 –91, june 2011.

[59] D. Quercia, S. Hailes, and L. Capra, "B-trust: Bayesian trust framework for pervasive computing," in *Trust Management*, ser. Lecture Notes in Computer Science, K. Stølen, W. Winsborough, F. Martinelli, and F. Massacci, Eds. Springer Berlin / Heidelberg, 2006, vol. 3986, pp. 298–312. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/11755593_22

[60] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 2003.

[61] S. T. Zargar, H. Takabi, and J. Joshi, "Dcdidp: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments." IEEE, 4 2012.

[62] S. S. Tripathi and S. Agrawal, "A survey on enhanced intrusion detection system in mobile ad hoc network," *International Journal of Advanced Research in Computer Engineering and Technology(IJARCET)*, vol. 1, no. 7, 2012. [Retrieved: 18-May-2013] http://ijarcet.org/index.php/ijarcet/article/view/367

[63] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 275–283. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/345910.345958

[64] A. Mishra, K. Nadkarni, and A. Patcha, "Intrusion detection in wireless ad hoc networks," *Wireless Communications, IEEE*, vol. 11, no. 1, pp. 48 – 60, feb 2004.

[65] O. Kachirski and R. Guha, "Effective intrusion detection using multiple sensors in wireless ad hoc networks," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, jan. 2003, p. 8 pp.

[66] ——, "Intrusion detection using mobile agents in wireless ad hoc networks," in *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*, 2002, pp. 153 – 158.

[67] D. B. Roy, R. Chaki, and N. Chaki, "A new cluster-based wormhole intrusion detection algorithm for mobile ad-hoc networks," *CoRR*, vol. abs/1004.0587, 2010.

[68] H. Deng, R. Xu, J. Li, F. Zhang, R. Levy, and W. Lee, "Agent-based cooperative anomaly detection for wireless ad hoc networks," in *Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on*, vol. 1, 0-0 2006, p. 8 pp.

[69] M. S. I. Mamun and A. F. M. S. Kabir, "Hierarchical design based intrusion detection system for wireless ad hoc network," *CoRR*, vol. abs/1208.3772, 2012.

[70] A. Morais and A. Cavalli, "A distributed intrusion detection scheme for wireless ad hoc networks," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 556–562. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/2245276.2245382

[71] N. Deb and N. Chaki, “Tids: Trust-based intrusion detection system for wireless ad-hoc networks,” in *Computer Information Systems and Industrial Management*, ser. Lecture Notes in Computer Science, A. Cortesi, N. Chaki, K. Saeed, and S. Wierzchon, Eds. Springer Berlin / Heidelberg, 2012, vol. 7564, pp. 80–91. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-33260-9_6

[72] K.-W. Yeom and J.-H. Park, “An immune system inspired approach of collaborative intrusion detection system using mobile agents in wireless ad hoc networks,” in *Computational Intelligence and Security*, ser. Lecture Notes in Computer Science, Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, Eds. Springer Berlin / Heidelberg, 2005, vol. 3802, pp. 204–211, 10.1007/11596981-31. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/11596981_31

[73] K. Aberer and Z. Despotovic, “Managing trust in a peer-2-peer information system,” in *Proceedings of the tenth international conference on Information and knowledge management*, ser. CIKM ’01. New York, NY, USA: ACM, 2001, pp. 310–317. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/502585.502638

[74] L. Mui, M. Mohtashemi, and A. Halberstadt, “A computational model of trust and reputation,” in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, jan. 2002, pp. 2431 – 2439.

[75] Z. Liu, A. W. Joy, and R. A. Thompson, “A dynamic trust model for mobile ad hoc networks,” *Future Trends of Distributed Computing Systems, IEEE International Workshop*, vol. 0, pp. 80–85, 2004.

[76] A. Pirzada, A. Datta, and C. McDonald, “Trust-based routing for ad-hoc wireless networks,” in *Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 1, nov. 2004, pp. 326 – 330 vol.1.

[77] F. Azzedin and M. Maheswaran, “Evolving and managing trust in grid computing systems,” in *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, vol. 3, 2002, pp. 1424 – 1429 vol.3.

[78] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, “The role of trust management in distributed systems security,” in *Secure Internet Programming*, ser. Lecture

Notes in Computer Science, J. Vitek and C. Jensen, Eds. Springer Berlin / Heidelberg, 1999, vol. 1603, pp. 185–210.

[79] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *In Proceedings of the 1996 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1996, pp. 164–173.

[80] I. Thomas, M. Menzel, and C. Meinel, "Using quantified trust levels to describe authentication requirements in federated identity management," in *Proceedings of the 2008 ACM workshop on Secure web services*, ser. SWS '08. New York, NY, USA: ACM, 2008, pp. 71–80. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1456492.1456504

[81] B. K. Bhargava and Y. Zhong, "Authorization based on evidence and trust," in *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, ser. DaWaK 2000. London, UK: Springer-Verlag, 2002, pp. 94–103. [Retrieved: 18-May-2013] http://portal.acm.org/citation.cfm?id=646111.679598

[82] E. Papalilo and B. Freisleben, "Towards a flexible trust model for grid environments," in *Grid Services Engineering and Management*, ser. Lecture Notes in Computer Science, M. Jeckle, R. Kowalczyk, and P. Braun, Eds. Springer Berlin / Heidelberg, 2004, vol. 3270, pp. 35–65.

[83] Y. Wang and J. Vassileva, "Bayesian network-based trust model," in *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, oct. 2003, pp. 372 – 378.

[84] Y. Sun, Z. Han, W. Yu, and K. Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –13.

[85] W. Yu, Y. Sun, and K. J. R. Liu, "Hadof: Defense against routing disruptions in mobile ad hoc networks," in *in IEEE INFOCOM*, 2005, pp. 1251–1261.

[86] F. Bao, I.-R. Chen, M. Chang, and J.-H. Cho, "Trust-based intrusion detection in wireless sensor networks," in *Communications (ICC), 2011 IEEE International Conference on*, june 2011, pp. 1 –6.

[87] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *Communications Surveys Tutorials, IEEE*, vol. 11, no. 2, pp. 52 –73, quarter 2009.

[88] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "Location verification and trust management for resilient geographic routing," *Journal of Parallel and Distributed Computing*, vol. 67, no. 2, pp. 215 – 228, 2007. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/pii/S0743731506001602

[89] F. Wang, C. Huang, J. Zhao, and C. Rong, "Idmtm: A novel intrusion detection mechanism based on trust model for ad hoc networks," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, march 2008, pp. 978 –984.

[90] B.-J. Chang and S.-L. Kuo, "Markov chain trust model for trust-value analysis and key management in distributed multicast manets," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 4, pp. 1846 –1863, may 2009.

[91] L. Ma, Y.-M. Zhang, and Q. Wei, "A trust evaluation method for dynamic distributed environment," in *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, vol. 4, july 2010, pp. 1935 –1940.

[92] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *Proceedings of the 2004 workshop on New security paradigms*, ser. NSPW '04. New York, NY, USA: ACM, 2004, pp. 67–77. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1065907.1066038

[93] TCG Trusted Platform Module Work Group, "Tpm main part 1 design principles," July 2007, specification Version 1.2 Level 2 Revision 103. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/resources/tpmspecificationversion12revision103part13

[94] ——, "TPM Main Part 2 TPM Structures," October 2006, specification Version 1.2 Level 2 Revision 103. [Retrieved: 18-

May-2013] http://www.trustedcomputinggroup.org/resources/tpm_specification_version_12_revision_103_part_1__3

[95] ——, “TPM Main Part 3 Commands,” October 2006, specification Version 1.2 Level 2 Revision 103. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/resources/tpm_specification_version_12_revision_103_part_1__3

[96] TCG Trusted Network Connect Work Group, “TNC Architecture for Interoperability, Version 1.5, Revision 3,” Trusted Computing Group, May 2012. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/resources/tnc_architecture_for_interoperability_version_13

[97] K. Dietrich and J. Winter, “Secure boot revisited,” in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, nov. 2008, pp. 2360 –2365.

[98] U. Kühn, M. Selhorst, and C. Stüble, “Realizing property-based attestation and sealing with commonly available hard- and software,” in *STC ’07: Proceedings of the 2007 ACM workshop on Scalable trusted computing.* New York, NY, USA: ACM, 2007, pp. 50–57.

[99] L. Chen, R. Landfermann, H. Löhr, M. Rohe, A.-R. Sadeghi, and C. Stüble, “A protocol for property-based attestation,” in *STC ’06: Proceedings of the first ACM workshop on Scalable trusted computing.* New York, NY, USA: ACM, 2006, pp. 7–16.

[100] R. Korthaus, A.-R. Sadeghi, C. Stüble, and J. Zhan, “A practical property-based bootstrap architecture,” in *STC ’09: Proceedings of the 2009 ACM workshop on Scalable trusted computing.* New York, NY, USA: ACM, 2009, pp. 29–38.

[101] V. Vijayakumar and R. WahidhaBanu, “Trust and reputation aware security for resource selection in grid computing,” in *Security Technology, 2008. SECTECH ’08. International Conference on*, dec. 2008, pp. 121 –124.

[102] V. Vijayakumar, R. S. Wahida Banu, and J. H. Abawajy, “An efficient approach based on trust and reputation for secured selection of grid resources,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 1, pp.

1–17, 2012. [Retrieved: 18-May-2013] http://www.tandfonline.com/doi/abs/10.1080/17445760.2011.575048

[103] B. Zhang, Y. Xiang, and Q. Xu, "Trust and reputation based model selection mechanism for decision-making," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, vol. 2, april 2010, pp. 14 –17.

[104] K.Selvi and D. W. Banu, "A hybrid model for load aware trust management in grid," *Journal of Computer Science*, vol. 7, no. 8, pp. 1237–1243, 2011.

[105] B. Gupta, H. Kaur, N. Namita, and P. Bedi, "Trust based access control for grid resources," in *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, june 2011, pp. 678 –682.

[106] TCG Infrastructure Work Group, "TCG Specification Architecture Overview," August 2007, specification Version 1.4. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Overview.pdf

[107] Trusted Computing Group, "TCG Website," 2012. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org

[108] F. Azzedin and M. Maheswaran, "Towards trust-aware resource management in grid computing systems," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, may 2002, p. 452.

[109] Trusted Computing Group, "TCG Glossary," 2012. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/developers/glossary

[110] B. Parno, J. M. McCune, and A. Perrig, "Roots of trust," in *Bootstrapping Trust in Modern Computers*, ser. SpringerBriefs in Computer Science. Springer New York, 2011, vol. 10, pp. 35–40. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-1-4614-1460-5_6

[111] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, pp. 91–98, May 2009. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1506409.1506429

[112] T. Vidas, "Volatile memory acquisition via warm boot memory survivability," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, jan. 2010, pp. 1 –6.

[113] I. Bente, J. Vieweg, and J. von Helden, "Towards trustworthy networks with open source software," *Horizons in Computer Science Research*, vol. 3, 2011.

[114] IEEE, "802.1X," 2004. [Retrieved: 18-May-2013] http://standards.ieee.org/getieee802/download/802.1X-2004.pdf

[115] B. Aboba, L. J. Blunk, J. R. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible authentication protocol (eap)," *RFC 3748*, June 2004.

[116] C. Rigney, S. Livingston, A. Merit, and W. Daydreamer, "Remote authentication dial in user service (radius)," *RFC 2865*, 2000.

[117] Ofir Arkin, "Bypassing Network Access Control Systems," September 2006. [Retrieved: 18-May-2013] http://www.blackhat.com/presentations/bh-dc-07/Arkin/Paper/bh-dc-07-Arkin-WP.pdf

[118] M. Thumann and D.-J. Roecher, "NAC@ATTACK - Hacking the Cisco NAC Framework," March 2007. [Retrieved: 18-May-2013] http://www.blackhat.com/presentations/bh-europe-07/Dror-Thumann/Whitepaper/bh-eu-07-dror-WP.pdf

[119] TCG Trusted Network Connect Work Group, "Platform Trust Services Interface Specification (IF-PTS)," November 2006, specification Version 1.0 Revision 1. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/temp/6427263A-1D09-3519-ADEE3EFF23C8F901/IWG_20IF-PTS_v1.pdf

[120] I. Bente, J. Vieweg, and J. von Helden, "Privacy enhanced trusted network connect," in *Trusted Systems*, ser. Lecture Notes in Computer Science, L. Chen and M. Yung, Eds. Springer Berlin / Heidelberg, 2010, vol. 6163, pp. 129–145. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-14597-1_8

[121] TCG Trusted Network Connect Work Group, "TNC IF-M: TLV Binding," January 2010, specification Version 1.0 Revision 37. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/495862FF-1D09-3519-AD8977DC98C1167C/TNC_IFM_TLVBinding_v1_0_r37a.pdf

[122] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM conference on Computer and communications security*, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 132–145. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1030083.1030103

[123] J. Camenisch, "Better privacy for trusted computing platforms," in *ESORICS '04: Proceedings of 9th European Symposium On Research in Computer Security.* Springer, 2004.

[124] J. M. McCune, A. Perrig, A. Seshadri, and L. van Doorn, "Turtles all the way down: research challenges in user-based attestation," in *Proceedings of the 2nd USENIX workshop on Hot topics in security*, ser. HOTSEC'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 6:1–6:5. [Retrieved: 18-May-2013] http://dl.acm.org/citation.cfm?id=1361419.1361425

[125] E. Cesena, G. Ramunno, R. Sassu, D. Vernizzi, and A. Lioy, "On scalability of remote attestation," in *Proceedings of the sixth ACM workshop on Scalable trusted computing*, ser. STC '11. New York, NY, USA: ACM, 2011, pp. 25–30. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/2046582.2046588

[126] I. Bente, J. Vieweg, J. von Helden, M. Jungbauer, and N. Pohlmann, "tNAC - Trusted Network Access Control," *Poster presented at the 19th Usenix Security Symposium (USENIX '10)*, 2010.

[127] B. Pfitzmann, J. Riordan, C. Stüble, M. Waidner, and A. Weber, "The perseus system architecture," 2001.

[128] A. Alkassar and C. Stüble, "Die Sicherheitsplattform Turaya," in *Trusted Computing*, N. Pohlmann and H. Reimer, Eds. Vieweg+Teubner, 2008, pp. 86–96. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-8348-9452-6_7

[129] S. Kent and K. Seo, “Security Architecture for the Internet Protocol,” RFC 4301 (Proposed Standard), Internet Engineering Task Force, December 2005. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc4301.txt

[130] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, “Internet Key Exchange Protocol Version 2 (IKEv2),” RFC 5996 (Proposed Standard), Internet Engineering Task Force, Sep. 2010, updated by RFC 5998. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc5996.txt

[131] M. Feilner, *OpenVPN: Building and Integrating Virtual Private Networks.* Packt Publishing, 2006.

[132] A.-R. Sadeghi and S. Schulz, “Extending ipsec for efficient remote attestation,” in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, R. Sion, R. Curtmola, S. Dietrich, A. Kiayias, J. Miret, K. Sako, and F. Sebé, Eds. Springer Berlin / Heidelberg, 2010, vol. 6054, pp. 150–165. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-14992-4_14

[133] S. Schulz and A.-R. Sadeghi, “Secure vpns for trusted computing environments,” in *Trusted Computing*, ser. Lecture Notes in Computer Science, L. Chen, C. Mitchell, and A. Martin, Eds. Springer Berlin / Heidelberg, 2009, vol. 5471, pp. 197–216. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-00587-9_13

[134] F. Baiardi and D. Sgandurra, “Attestation of integrity of overlay networks,” *Journal of Systems Architecture*, vol. In Press, Corrected Proof, pp. –, 2010. [Retrieved: 18-May-2013] http://www.sciencedirect.com/science/article/B6V1F-508PPYT-1/2/59cabe0d98e91e12c75b03d76b270d9f

[135] I. Bente, B. Hellmann, J. Vieweg, J. von Helden, and A. Welzel, “Interoperable remote attestation for vpn environments,” in *Proceedings of the Second international conference on Trusted Systems*, ser. INTRUST’10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 302–315. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-25283-9_20

[136] TCG Trusted Network Connect Work Group, “TNC IF-T: Binding to TLS,” May 2009, specification Version 1.0 Revision 16. [Re-

trieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/51F0757E-1D09-3519-AD63B6FD099658A6/TNC_IFT_TLS_v1_0_r16.pdf

[137] L. Chen, H. Löhr, M. Manulis, and A.-R. Sadeghi, "Property-based attestation without a trusted third party," in *Information Security*, ser. Lecture Notes in Computer Science, T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, Eds. Springer Berlin / Heidelberg, 2008, vol. 5222, pp. 31–46. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-540-85886-7_3

[138] A. Nagarajan, V. Varadharajan, M. Hitchens, and E. Gallery, "Property based attestation and trusted computing: Analysis and challenges," in *Network and System Security, 2009. NSS '09. Third International Conference on*, oct. 2009, pp. 278 –285.

[139] C. Kil, E. Sezer, A. Azab, P. Ning, and X. Zhang, "Remote attestation to dynamic system properties: Towards providing complete system integrity evidence," in *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, 29 2009-july 2 2009, pp. 115 –124.

[140] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "Swatt: software-based attestation for embedded devices," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, may 2004, pp. 272 – 282.

[141] S. Bugiel, L. V. Davi, and S. Schulz, "Scalable trust establishment with software reputation," in *Proceedings of the sixth ACM workshop on Scalable trusted computing*, ser. STC '11. New York, NY, USA: ACM, 2011, pp. 15–24. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/2046582.2046587

[142] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, oct. 2007, pp. 219 –230.

[143] X.-Y. Li, C.-X. Shen, and X.-D. Zuo, "An efficient attestation for trustworthiness of computing platform," in *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP '06. International Conference on*, dec. 2006, pp. 625 –630.

[144] W. Arbaugh, D. Farber, and J. Smith, “A secure and reliable bootstrap architecture,” in *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, may 1997, pp. 65 –71.

[145] M. Alam, X. Zhang, M. Nauman, T. Ali, and J.-P. Seifert, “Model-based behavioral attestation,” in *Proceedings of the 13th ACM symposium on Access control models and technologies*, ser. SACMAT ’08. New York, NY, USA: ACM, 2008, pp. 175–184. [Retrieved: 18-May-2013] http://doi.acm.org/10.1145/1377836.1377864

[146] V. Haldar, D. Chandra, and M. Franz, “Semantic remote attestation - a virtual machine directed approach to trusted computing,” in *USENIX Virtual Machine Research and Technology Symposium*, 2004, pp. 29–41.

[147] M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert, “Beyond kernel-level integrity measurement: Enabling remote attestation for the android platform,” in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, A. Acquisti, S. Smith, and A.-R. Sadeghi, Eds. Springer Berlin / Heidelberg, 2010, vol. 6101, pp. 1–15. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-13869-0_1

[148] “Integrity Measurement Architecture (IMA) Website.” [Retrieved: 18-May-2013] http://linux-ima.sourceforge.net/

[149] I. Bente, G. Dreo, B. Hellmann, S. Heuser, J. Vieweg, J. von Helden, and J. Westhuis, “Towards permission-based attestation for the android platform,” in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, J. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, Eds. Springer Berlin / Heidelberg, 2011, vol. 6740, pp. 108–115. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-642-21599-5_8

[150] Chromium OS, Design Documents. [Retrieved: 18-May-2013] http://www.chromium.org/chromium-os/chromiumos-design-docs/

[151] I. Bente, B. Hellmann, T. Rossow, J. Vieweg, and J. von Helden, “On remote attestation for google chrome os,” in *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, sept. 2012, pp. 376 –383.

[152] TCG Trusted Network Connect Work Group, "TNC IF-MAP Binding for SOAP, Version 2.1, Revision 15," Trusted Computing Group, May 2012.

[153] G. Mühl, "Large-scale content-based publish-subscribe systems," Ph.D. dissertation, TU Darmstadt, November 2002. [Retrieved: 18-May-2013] http://tuprints.ulb.tu-darmstadt.de/274/

[154] Trust@FHH Research Group, "TNC@FHH Project Page," 2012. [Retrieved: 18-May-2013] http://trust.inform.fh-hannover.de/joomla/index.php/projects/tncfhh

[155] TCG Trusted Network Connect Work Group, "TNC IF-MAP Metadata for Network Security, Version 1.1, Revision 8," Trusted Computing Group, May 2012.

[156] ——, "TNC IF-MAP Metadata for ICS Security, Version 1.0, Revision 39," Trusted Computing Group, October 2012.

[157] M. Boisot and A. Canals, "Data, information and knowledge: have we got it right?" *Journal of Evolutionary Economics*, vol. 14, no. 1, pp. 43–67, Jan. 2004. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/s00191-003-0181-9

[158] R. Gerhards, "The Syslog Protocol," RFC 5424 (Proposed Standard), Internet Engineering Task Force, March 2009. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc5424.txt

[159] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc5246.txt

[160] E. Rescorla, "HTTP Over TLS," RFC 2818 (Informational), Internet Engineering Task Force, May 2000. [Retrieved: 18-May-2013] http://www.ietf.org/rfc/rfc2818.txt

[161] H. F. Nielsen, N. Mendelsohn, J. J. Moreau, M. Gudgin, M. Hadley, A. Karmarkar, and Y. Lafon, "SOAP version 1.2 part 0: Primer," W3C, W3C Recommendation, Jun. 2007.

[162] H. F. Nielsen, N. Mendelsohn, J. J. Moreau, M. Gudgin, and M. Hadley, “SOAP version 1.2 part 1: Messaging framework,” W3C, W3C Recommendation, Jun. 2007.

[163] T. Bray, J. Paoli, E. Maler, F. Yergeau, and C. M. Sperberg-McQueen, “Extensible markup language (XML) 1.0 (fifth edition),” W3C, W3C Recommendation, Nov. 2008, http://www.w3.org/TR/2008/REC-xml-20081126/.

[164] C. M. Sperberg-McQueen, H. S. Thompson, M. Maloney, H. S. Thompson, D. Beech, N. Mendelsohn, and S. S. Gao, “W3C xml schema definition language (XSD) 1.1 part 1: Structures,” W3C, Last Call WD, Dec. 2009, http://www.w3.org/TR/2009/WD-xmlschema11-1-20091203/.

[165] D. Peterson, S. S. Gao, P. V. Biron, A. Malhotra, H. S. Thompson, A. Malhotra, and C. M. Sperberg-McQueen, “W3C xml schema definition language (XSD) 1.1 part 2: Datatypes,” W3C, Last Call WD, Dec. 2009, http://www.w3.org/TR/2009/WD-xmlschema11-2-20091203/.

[166] O. Diehl, “Vertrauenswürdigkeitsbewertungen von Metadaten in Netzwerken,” Bachelor’s Thesis, Hochschule Hannover, Mar. 2012.

[167] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[168] “Trust at FHH Research Group Site.” [Retrieved: 18-May-2013] http://trust.inform.fh-hannover.de/

[169] OpenVAS Project Team, “OpenVAS Project Page,” 2012. [Retrieved: 18-May-2013] http://www.openvas.org

[170] S. Oezkan, “CVE Details Page,” 2012. [Retrieved: 18-May-2013] http://cvedetails.com

[171] Internet Systems Consortium, “Isc dhcp project page,” 2012. [Retrieved: 18-May-2013] http://www.isc.org/software/dhcp

[172] Samsung Group, “Samsung Galaxy S III Webpage,” 2012. [Retrieved: 18-May-2013] http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/GT-I9300MBDBTU

[173] ——, "Samsung Galaxy Nexus Webpage," 2012. [Retrieved: 18-May-2013] http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/GT-I9250TSAXEU

[174] P. McHardy, J. Kadlecsik, P. N. Ayuso, E. Leblond, and F. Westphal, "netfilter/iptables Project Page," 2012. [Retrieved: 18-May-2013] http://www.netfilter.org

[175] A. Welzel and I. Bente, "A Note on IF-MAP Performance," HS Hannover, Trust@FHH, Technical Report, Nov. 2011.

[176] Arne Welzel, "libifmap2c Project Page," 2012. [Retrieved: 18-May-2013] http://code.google.com/p/libifmap2c/

[177] Google Inc., "Google Play Store," 2012. [Retrieved: 18-May-2013] https://play.google.com/

[178] LANCOM Systems GmbH, "LANCOM L-54g Wireless," 2012. [Retrieved: 18-May-2013] http://www.lancom-systems.de/en/l-54g-wireless-overview/

[179] D. Vassis, G. Kormentzas, A. Rouskas, and I. Maglogiannis, "The IEEE 802.11g standard for high data rate WLANs," vol. 19, pp. 21–26, May 2005. [Retrieved: 18-May-2013] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&#38;arnumber=1453395&#38;isnumber=31204

[180] R. Steuerwald, "Integration von OpenVAS in IF-MAP," Bachelor's Thesis, Hochschule Hannover, Jun. 2011.

[181] P. Khlebovich, "IP Webcam," 2013. [Retrieved: 18-May-2013] https://play.google.com/store/apps/details?id=com.pas.webcam

[182] R. Marx, N. Kuntze, C. Rudolph, I. Bente, and J. Vieweg, "Trusted Service Access with Dynamic Security Infrastructure Configuration," *Presented at the 18th Asia-Pacific Conference on Communications (APCC 2012)*, 2012.

[183] I. Bente, J. Vieweg, and J. Helden, "ESUKOM: Smartphone Security for Enterprise Networks," in *ISSE 2011 Securing Electronic Business Processes*, N. Pohlmann,

H. Reimer, and W. Schneider, Eds. Vieweg + Teubner Verlag — Springer Fachmedien Wiesbaden GmbH, 2012.

[184] N. Kuntze, C. Rudolph, I. Bente, J. Vieweg, and J. von Helden, "Interoperable device identification in Smart-Grid environments," in *Power and Energy Society General Meeting, 2011 IEEE*, july 2011, pp. 1 –7.

[185] I. Bente, J. Vieweg, and J. Helden, "Countering Phishing with TPM-bound Credentials," in *ISSE 2010 Securing Electronic Business Processes*, N. Pohlmann, H. Reimer, and W. Schneider, Eds. Vieweg+Teubner, 2011, pp. 236–246. [Retrieved: 18-May-2013] http://dx.doi.org/10.1007/978-3-8348-9788-6_23

[186] Intel Corporation, "Intel Trusted Execution Technology (Intel TXT) Measured Launched Environment Developer's Guide," December 2009. [Retrieved: 18-May-2013] http://download.intel.com/technology/security/downloads/315168.pdf

[187] Advanced Micro Devices, "Secure Virtual Machine Architecture Reference Manual," May 2005. [Retrieved: 18-May-2013] http://www.mimuw.edu.pl/~vincent/lecture6/sources/amd-pacifica-specification.pdf

[188] TCG Trusted Network Connect Work Group, "TNC IF-T: Protocol Bindings for Tunneled EAP Methods," May 2007, specification Version 1.1 Revision 10. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/8CC75909-1D09-3519-ADA6958AA29CF223/TNC_IFT_v1_1_r10.pdf

[189] ——, "TNC IF-PEP: Protocol Bindings for RADIUS," February 2007, specification Version 1.1 Revision 0.7. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/8CC5592B-1D09-3519-AD45F0F893766F6B/TNC_IF-PEP_v1.1_rev_0.7.pdf

[190] ——, "TNC IF-TNCCS: TLV Binding," January 2010, specification Version 2.0 Revision 16. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/495CA3DD-1D09-3519-AD0043966E821ECB/IF-TNCCS_TLVBinding_v2_0_r16a.pdf

[191] ——, "TNC IF-TNCCS: Protocol Bindings for SoH," May 2007, specification Version 1.0 Revision 0.8. [Re-

trieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/8D2DF7F3-1D09-3519-AD76CE4433FECE07/IF-TNCCS-SOH_v1.0_r8.pdf

[192] Microsoft Corporation, "Network Access Protection," 2012. [Retrieved: 18-May-2013] http://technet.microsoft.com/en-us/network/bb545879.aspx

[193] TCG Trusted Network Connect Work Group, "TNC IF-IMC," May 2006, specification Version 1.1 Revision 5. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/static_page_files/1D4F4303-1D09-3519-AD13BD81B3D741BB/TNC_IFIMC_v1_1_r5.pdf

[194] ——, "TNC IF-IMC," February 2007, specification Version 1.2 Revision 8. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/resource_files/8CB977E1-1D09-3519-AD48484530EF6639/TNC_IFIMC_v1_2_r8.pdf

[195] ——, "TNC IF-IMV," May 2005, specification Version 1.0 Revision 3. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/static_page_files/1D507DD9-1D09-3519-AD5C7FBC9B7BB368/TNC_IFIMV_v1_0_r3.pdf

[196] ——, "TNC IF-IMV," February 2007, specification Version 1.2 Revision 8. [Retrieved: 18-May-2013] http://www.trustedcomputinggroup.org/files/static_page_files/646808C3-1D09-3519-AD2E60765779A42A/TNC_IFIMV_v1_2_r8.pdf

[197] R. Sahita, U. R. Savagaonkar, P. Dewan, and D. Durham, "Mitigating the lying-endpoint problem in virtualized network access frameworks," in *DSOM'07: Proceedings of the Distributed systems: operations and management 18th IFIP/IEEE international conference on Managing virtualization of networks and services.* Berlin, Heidelberg: Springer-Verlag, 2007, pp. 135–146.