Bachelorarbeit

Generierung vertrauenswürdiger Daten auf mobilen Endgeräten auf Basis von Trusted Computing

zur Erlangung des akademischen Grades

Bachelor of Science

vorgelegt dem
Fachbereich Mathematik, Naturwissenschaften und Informatik der TH Mittelhessen

Dominik Sebastian Weiß
im Juni 2012

Referent:	Prof. Dr. Michael Jäger
Korreferent:	Dipl. Inform. Nicolai Kuntze

Inhaltsverzeichnis

Vorwort

Diese Arbeit ist im Rahmen eines Entwicklungsprojekts von mehreren Studenten im Wintersemester 2011/12 erstellt worden. Jeder Student hatte bei diesem Projekt einen eigenen Arbeitsbereich. Meiner war die Erstellung und Übertragung von Daten. Diese Daten sollen auf einer vertrauenswürdigen Plattform generiert werden. Um einzelne Teile des Projekts nicht mehrfach neu zu beschreiben, wurden Informationen an die anderen Teampartner weitergegeben. An dem Projekt haben die Herren Dieter Kramer und Mark Schlüter mitgearbeitet. Die Themen ihrer Arbeiten lauten:
"Realisierung einer mobilen vertrauenswürdigen Geschäftsplattform auf Basis von Trusted Computing zur gesicherten Datenerfassung" [Sch12] von Herrn Schlüter und
"Konzeption und Entwicklung einer regelbasierten Verarbeitung von Events auf der Basis von Trusted Computing" [Kra12] von Herrn Kramer.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und ohne unerlaubte Hilfe angefertigt und andere als die in der Bachelorarbeit angegebene Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Gießen, Juni 2012

(Dominik Weiß)

Teil I

Einführung

Kapitel 1

Einleitung

Es gibt eine Vielzahl von Anwendungsgebieten in denen es wichtig ist, Vertrauen in die Richtigkeit digitaler Daten zu haben. Ein gutes Beispiel hierfür sind Fotos, die als Beweismittel verwendet werden. Es ist nicht möglich das Negativ eines analog aufgenommenen Fotos nachträglich zu verändern. Im Gegensatz dazu kann man ein digitales Bild sehr einfach mit einer Bildbearbeitungssoftware nachträglich bearbeiten. Um dies zu erkennen, kann man ein Bild nach der Erstellung digital signieren. Aber was ist wenn die Hardware mit der das Foto aufgenommen wird, sich bereits in einem nicht vertrauenswürdigen Zustand befindet. Daten können so bereits bei der Erstellung schon kompromittiert werden und eine anschließende Signierung kann diese Manipulation nicht aufdecken. Für eine lückenlose Beweisführung sollte bereits vor der Erzeugung von Daten Vertrauenswürdigkeit gewährleistet sein. Dazu muss der Aspekt der Sicherheit bereits in der Hardware verankert werden. Zudem sollte der komplette Weg von der Generierung bis zur Speicherung nachvollziehbar sein. Dafür reicht eine digitale Signatur alleine nicht aus.

Während der Arbeiten am Projekt TruLoc haben wir uns in einer Arbeitsgruppe Gedanken über verschiedene Szenarien gemacht. Anwendungsgebiete wären zum Beispiel komplexe Lieferketten-Systeme. Hier könnte man zum einen zurückgelegte Routen vertrauenswürdig speichern und mit zusätzlichen Sensoren die Temperatur der transportierten Waren überwachen. Weitere denkbare Szenarien wären eine automatisierte Produktionsstätte, oder auch der Transport von medizinischen Gütern. Diese Szenarien werden in Kapitel 3.2 (S.10) genauer erläutert. Es gibt verschiedene Faktoren die berücksichtigt werden müssen, um alle an das Projekt gestellte Anforderungen zu erfüllen. Diese bestehen aus Sicherheitsanforderungen und verschiedenen juristischen Aspekten.

1.1 Problemstellung und Zielsetzungen

Aufgabenstellung ist es Positionsdaten und/oder Routen vertrauenswürdig zu erstellen, sie zu archivieren und zu visualisieren. Gleichzeitig soll es möglich sein, die Daten zu einem späteren Zeitpunkt zu verifizieren. Die so gewonnenen Daten sollen in einer sicheren Umgebung erzeugt werden. Gerichte haben es momentan schwer digitale Daten als Beweismittel anzuerkennen, da eine Manipulation nicht immer aufgedeckt werden kann. Dieses Entwicklungsprojekt stellt den technischen Aspekt für eine belegbare Datengenerierung dar. Der juristische Aspekt wird nicht betrachtet. Dies ist Aufgabe der Juristen, da sie den nötigten Überblick über die Gesetzeslage besitzen.

Als Datengenerator wird ein Gerät der Android-Plattform zum Einsatz kommen. Während der Arbeiten an dem Projekt wird dazu neben dem Emulator ein Android Smartphone mit angepasstem Betriebssystem genutzt. Die Verifikation der Daten

kann auf einem Tablet-PC, auf welchem ebenfalls Android installiert ist, vorgenommen werden. Durch das größere Display ist es leichter sich einen Überblick über eine Route zu verschaffen. Das Betriebssystem des Verifikators muss allerdings nicht angepasst werden, da hier keine besonderen Sicherheitsmechanismen von Nöten sind. Somit können die Daten auf jedem zur Verfügung stehenden Gerät überprüft werden, auf dem die Applikation installiert ist. Ziel ist es eine lauffähige Version des Projekts TruLoc zu erstellen. Dazu werden verschiedene Ansätze für die Datenerstellung, ihre Übertragung und Speicherung erörtert und miteinander verglichen. Bei der Übertragung der Daten wird darauf geachtet, möglichst wenig Bandbreite zu verwenden. Dies wird durch die Skalierung der versendeten Daten erreicht. Je nach Größe der Arbeitsumgebung, in der die Software später eingesetzt werden soll, kann die Minimierung der zu übertragenden Daten einen wichtigen Faktor darstellen. Die Zeit, welche für die Verarbeitung von anfallenden Daten aufgewendet werden muss, stellt einen weiteren Punkt dar.

1.2 Anforderungen

Es gibt verschiedene Anforderungen die bei der Umsetzung zu erfüllen sind. Zum einen gibt es den Aspekt der Sicherheit. Zum anderen muss auch darauf geachtet werden, dass die gewonnenen Daten juristisch einwandfrei erstellt werden. Welche Punkte dabei genau in Betracht kommen, wird in den nächsten Abschnitten erklärt.

1.2.1 Sicherheitsanforderungen

Solange es Hard- und Software gibt, werden Angriffe auf diese durchgeführt, um Daten zu manipulieren oder sie sich unrechtmäßig anzueignen. In einer gesicherten Umgebung, wie beispielsweise einem Firmengebäude, kann kontrolliert werden, wer welchen Zugriff auf Anwendungen und Geräte hat. In Firmennetzwerken gibt es Firewalls gegen Angriffe von außen und Zugriffsbeschränkungen können mit Benutzerrechten und Passwörtern durchgesetzt werden. Des weiteren ist es möglich, nicht autorisierten Personen den Zugang zu gesicherten Bereichen zu verwehren. Dies lässt sich bei mobilen Geräten, wie zum Beispiel Handys oder Tablet-PCs schlecht bis gar nicht realisieren. Solche mobile Geräte befinden sich oft in einer Umgebung, in der eine Manipulation nicht ausgeschlossen werden kann. Bei einer kurzen Unaufmerksamkeit, etwa in einem Café wenn das Gerät auf dem Tisch liegt, kann es passieren, dass jemand Veränderungen vornimmt, oder das Gerät sogar entwendet. An diesem Punkt setzt dieses Projekt an. Mit unserer Software kann ein Diebstahl oder eine Manipulation nicht verhindert werden. Aber sie dient dazu, dass Veränderungen an einem Gerät aufgedeckt werden können.

1.2.2 Juristische Aspekte

Die erzeugten Daten sollen derart abgesichert werden, dass sie vor Gericht als Beweismittel verwendet werden können. Dazu müssen bestimmte Vorkehrungen getroffen werden. So kann beispielsweise nicht einfach der Zeitstempel des GPS Signals verwendet werden. Dieser wird vor Gericht nicht als Beweismittel zugelassen. GPS ist ein kontinuierlich empfangenes Signal. Um einen vor Gericht nutzbaren Zeitstempel zu erhalten, muss dieser von der Anwendung angefordert werden. Somit wird sichergestellt, dass nicht ein beliebiger Zeitstempel genutzt wird. Ebenfalls muss die Umgebung in der die Daten erstellt werden ein Mindestmaß an Vertrauenswürdigkeit erfüllen. Mit Umgebung ist nicht der Ort gemeint an dem die Daten erstellt werden, sondern die verwendete Hard- und Software.

Kapitel 2

Technische Grundlagen

Im diesem Kapitel werden die Technischen Grundlagen vorgestellt. Sie wurden zuvor in einem Projekt an der TH Mittelhessen erarbeitet. Es lief unter dem Titel TruLoc. In diesem Projekt wurden vertrauenswürdige Bilder unter Zuhilfenahme eines Android Handys erstellt. Dabei kamen vergleichbare Hardwarekomponenten und Sicherheitsanforderungen zum Einsatz.

2.1 Trusted Computing

Als Trusted Computing (TC) bezeichnet man verschiedene Konzepte und Spezifikationen, welche dazu dienen, nicht vertrauenswürdige Hard- und Software sicherer zu machen. Diese Spezifikationen werden von der Trusted Computing Group entwickelt und beworben.

2.1.1 Trusted Computing Group

Diese Gruppe, kurz TCG genannt, ist eine nicht profitorientierte Standardisierungs-Organisation. Sie besteht aus einem Zusammenschluss vieler namhafter Firmen wie z.B. AMD, IBM, HP, Intel uvm. Sie hat sich die Entwicklung eines offenen Standards für Trusted Computing Plattformen als Ziel gesetzt. Die TCG hat 2003 die Standardisierungsarbeit der Trusted Computing Platform Alliance (TCPA) übernommen. Auf der offiziellen Website definiert sich die Gruppe selbst so:

> "The Trusted Computing Group (TCG) is a not-for-profit organization formed to develop, define and promote open, vendor-neutral, industry standards for trusted computing building blocks and software interfaces across multiple platforms." [1]

Innerhalb der TCG gibt es eine Vielzahl von Arbeitsgruppen zu unterschiedlichen Themengebieten, welche die jeweiligen Spezifikationen erarbeiten. Zu den Themengebieten gehören u.a. das Trusted Platform Module sowie die TNC Architektur.

2.1.2 Trusted Platform Module

Das Trusted Platform Module (TPM) dient dazu, aus einer nicht vertrauenswürdigen Plattform, wie einem Smartphone, eine vertrauenswürdige zu machen. Somit ist es ein Kernelement der verwendeten Sicherheitsarchitektur.

[1] Quelle: http://www.trustedcomputinggroup.org

Der vollkommen passive Chip enthält verschiedene kryptographische Funktionen und Methoden zur Sicherung von internen und externen Daten. Außerdem enthält er geschützte Speicherbereiche, in denen z.B. Schlüsselpaare abgelegt werden können. Eines dieser Schlüsselpaare ist der "Endorsement Key" (EK). Hierbei handelt es sich um ein einmaliges Schlüsselpaar, welches jeder TPM enthält. Der EK ist ein 2048 Bit RSA Schlüssel und dient zur Signierung von Daten. Er ist fest in den Chip integriert und kann nachträglich nicht mehr geändert werden. Der private Teil des Schlüsselpaars wird den TPM-Chip niemals verlassen. Jeder Chip kann über diesen Key eindeutig identifiziert werden. Daten können mit dem öffentlichen Teil des Schlüsselpaars signiert werden. Da der private Teil des Schlüsselpaars nicht von außen zugänglich ist, können mit dem EK verschlüsselte oder signierte Daten auch nur durch den TPM wieder entschlüsselt oder verifiziert werden. Aus Gründen der Privatsphäre werden im Normalfall Daten nicht mit dem EK signiert, da alle auf diese Art signierten Daten zu dem verwendeten TPM zurückverfolgt werden können. Stattdessen werden mit dem EK neue Schlüssel generiert, sogenannte Attestation Identity Keys (AIK). Mit diesen können dann die Signierungen vorgenommen werden. Sollte mit einem TPM-Chip gesicherte Hardware in einer automatisierten Produktion eingesetzt werden, kann es allerdings auch von Vorteil sein, direkt den EK zu verwenden. Hierbei muss nicht auf die Privatsphäre geachtet werden und ein manipulierter oder defekter Netzwerkknoten kann somit leichter ausfindig gemacht werden.

Der TPM-Chip wird bereits in einer Vielzahl von Rechnern verbaut. So muss z.B. ein Rechner um das "Vista Ready" Logo zu bekommen einen TPM-Chip enthalten. Auch wenn der Chip bereits verbaut ist, wird er in den meisten Fällen nicht von der Software angesteuert und ist in dem Moment nur ein Stromverbraucher. Dies wollen wir mit unserer Software ändern.

2.1.3 TNC-Architektur

Trusted Network Connect nennt sich eine 2004 ins Leben gerufene Arbeitsgruppe der TCG. Diese Subgruppe hat eine Architektur entwickelt, welche zur Überprüfung der Integrität von Systemen, die auf ein Netzwerk zugreifen wollen, dient. Die TNC-Architektur ermöglicht es Regeln und Sicherheitsrichtlinien bezüglich der Vertrauenswürdigkeit von Endgeräten durchzusetzen. Des weiteren können Informationen über System- und Endgeräte- Zustand mit einem gesicherten Kommunikationsprotokoll übertragen werden. TNC baut auf vorhandenen Technologien auf und kann somit in bestehende Systeme integriert werden. Um eine weitere Sicherheitskomponente zu integrieren, ist es möglich auf ein TPM zurückzugreifen. Die folgende Abbildung (Abb. 2.1) zeigt das Gesamtkonzept der TCG-Architektur.

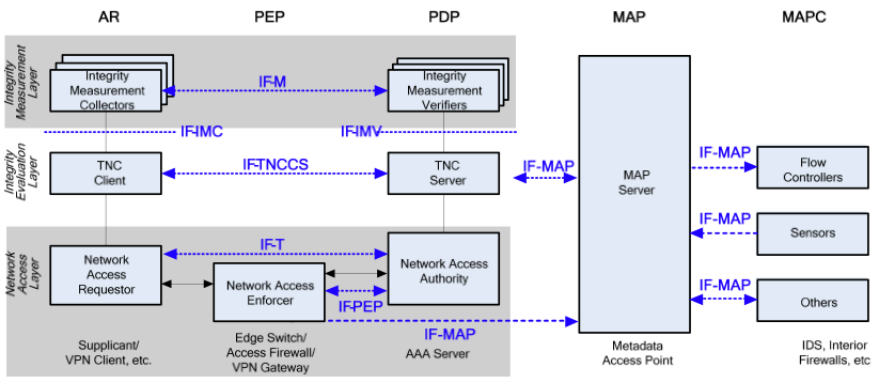

Abb. 2.1: TNC Architektur, ©Trusted Computing Group

2.2 Infrastruktur

Die für das Projekt benötigte Infrastruktur ist der des abgeschlossenen Projekts TruCam relativ ähnlich, somit können die meisten Komponenten als Grundlagen für das neue Projekt übernommen werden. Genauer gesagt werden die Privacy Certification Authority (PCA), Timing Authority (TA), das Archiv und der MAP-Server nicht neu implementiert.

TruCam war ebenfalls ein Entwicklungsprojekt in dem ein erster Prototyp der Infrastruktur entwickelt wurde. Einige der erstellten Softwarekomponenten sind noch nicht vollständig funktionsfähig. Diese Komponenten werden im Laufe der Entwicklungsarbeiten für TruCam überarbeitet und eventuelle Fehler behoben. Des weiteren stellt TruLoc einen Test für die Infrastruktur dar.

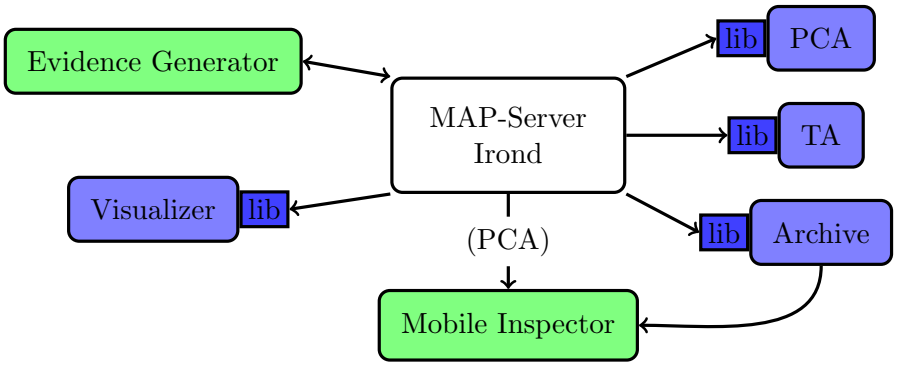

Abb. 2.2: Übersicht der Infrastruktur

Die grün dargestellten Komponenten sind Android Applikationen, wogegen die blauen Komponenten Java Programme sind.

2.2.1 MAP-Server

Der Metadata Access Point, kurz MAP-Server, welcher hier verwendet wird, ist eine open source Entwicklung der Fachhochschule Hannover aus dem IROND-Projekt. Es ist eine Implementierung des Interface zwischen dem Metadata Access Point (MAP) und anderen Komponenten der TNC-Architektur. Alle Komponenten kommunizieren miteinander unter Verwendung des MAP-Servers. Die Kommunikation erfolgt nach dem Publish-Subscribe Prinzip.

MAP-Clients veröffentlichen (publishen) ihre Informationen über den MAP-Server. Andere Clients können für sie relevante Informationen abonnieren und abrufen (subscriben). Die einzige Ausnahme bildet hier der Mobile Inspector (MI), da er über Webservices an das Archiv angebunden ist. Gegenüber dem Original Server wurde im Rahmen des Projekts ein Regelwerk und ein Policy-Framework an den Server angebunden. Hiermit können verschiedene Regeln bezüglich Zugriff, Lese- und Schreibrechte durchgesetzt werden.

2.2.2 Privacy Certification Authority

Die Privacy Certification Authority (PCA) dient als zentrale Zertifizierungsstelle. Mit der PCA werden verschiedene vom TPM erzeugte AIKs überprüft und signiert. Somit wird sichergestellt, dass die Schlüsselpaare von einer vertrauenswürdigen Plattform stammen. Die PCA ist ein ständig laufender Service, welcher an den MAP-Server angebunden ist. Jede Anfrage einen AIK zu bestätigen wird von ihr bearbeitet und falls

der AIK korrekt ist, mit einer signierten Version des AIK beantwortet. Somit kann sichergestellt werden, dass auf den einzelnen Geräten ein laufender TPM-Chip vorhanden ist.

Gleichzeitig wird so gewährleistet, dass der EK eines TPM nicht verwendet werden muss. Dadurch ist die Problematik der Nachverfolgung eines bestimmten TPM-Chip nicht mehr gegeben und die Privatsphäre des Benutzers wird gewahrt.

2.2.3 Timing Authority

Die Timing Authority (TA) wird benötigt, um dem vom TPM-Chip generierten sogenannten Tickstamp eine reelle Zeit zuzuordnen. Der Tickstamp ist eine kontinuierlich hoch zählende Integer Variable im TPM. Eine Synchronisierung läuft wie folgt ab: Der EG nutzt zur eigenen Synchronisierung die im Mobilgerät enthaltene Uhr und fordert vom TPM den aktuellen Tickstamp an. Diese Daten werden mit einem von der PCA verifizierten AIK signiert und an die TA gesendet. Die TA sendet als Antwort ein signiertes Paket bestehend aus Tickstamp und tatsächlicher Zeit. Die Kommunikation zwischen den Komponenten läuft nicht direkt, sondern über den MAP-Server. Durch die Verwendung der TA wird sichergestellt, dass der verwendete Zeitstempel vor Gericht Bestand hat.

2.2.4 Evidence Generator

Der Evidence Generator (EG) ist eine Android Applikation, welche auf einem angepassten Android System betrieben wird. Das Betriebssystem ist zusätzlich mit Trusted Computing Funktionalität ausgestattet. So wird z.B. jedes Programm beim Starten gemessen und das Ergebnis im Stored Measurement Log festgehalten. Zusätzlich wird ein TPM-Chip verwendet. Die TPM Funktionalität muss allerdings noch in Form eines Emulators nachgebildet werden, da sich Endgeräte mit TPM-Chip noch in der Entwicklung befinden.

Mit der Applikation werden die Wegdaten erzeugt. Die Daten werden signiert und in einem einstellbaren Intervall an den MAP-Server gesendet. Sollte keine Verbindung zu einem Datennetz bestehen, werden die Datenpakete in einer lokalen Datenbank zwischengespeichert. Dadurch wird sichergestellt, dass keine Daten verloren gehen und die Beweiskette nicht unterbrochen wird. Sobald wieder Zugriff auf ein Datennetz möglich ist, sendet der EG den Inhalt der Datenbank an den Server.

2.2.5 Archiv

Im Archiv werden die vom MAP-Server empfangenen Daten gespeichert. Das Archiv ist eine rein passive Komponente, d.h. die an das Archiv gesendeten Daten werden vor dem Speichern nicht validiert oder anderweitig verarbeitet. Mit dem Regelwerk können unterschiedliche Regeln für das Abrufen von Datensätzen festgelegt werden. Die Daten können über den MAP-Server, oder über RESTful[2] Webservices abgerufen werden. Unter RESTful Webservices versteht man die Idee, dass sich hinter einer Internetadresse (URL) immer der gleiche, reproduzierbare Seiteninhalt verbirgt.

[2]REST bezeichnet ein Programmierparadima für Webanwendungen

2.2.6 Mobile Inspector

Der Mobile Inspector dient zur Verifizierung der im Archiv liegenden Datensätze. Er ist eine Android Applikation ohne TPM Funktionalität. Diese wird nicht benötigt, da der MI keine Funktionalität zum Verändern von Daten enthält. Zunächst wird eine Übersicht der im Archiv vorhandenen Beweissätze angezeigt. Nach Auswahl eines Beweissatzes werden über Webservices die zu überprüfenden Datenpakete vom Archiv abgerufen. Diese werden auf dem Gerät auf einer Karte dargestellt. Während der Darstellung werden die Daten im Hintergrund verifiziert. Alle Wegpunkte werden mit Punkten auf der Karte angezeigt. Wenn ein Wegpunkt erfolgreich verifiziert wurde, wird er grün dargestellt. Bei gescheiterter Verifikation wird er rot angezeigt.

2.2.7 Visualisierer

Der Visualisierer dient als Monitor-Applikation zum Anzeigen des inneren Status vom irond Server. Er erhält vom irond Server die gesamten Metadaten. Dazu wird eine spezielle “dump” Anfrage an den Server gesendet. Die erhaltenen Daten werden von der Anwendung grafisch aufgearbeitet und so angezeigt, dass man die einzelnen Beweisketten voneinander unterscheiden kann. Der Visualisierer wird hauptsächlich zur Analyse und Fehlerbehebung genutzt.

2.2.8 Shared Library

In der IFMapLib genannten Bibliothek ist die IF-MAP Kommunikation implementiert worden. Dies hat den Vorteil, dass die Kommunikation nur einmal implementiert werden muss und alle Komponenten darauf zugreifen können. Von der IFMapLib gibt es eine Version für Android Programme und eine für Java Programme. Somit ist es allen verwendeten Komponenten möglich auf die Bibliothek zurückzugreifen.

Teil II

Analyse des Projekts

Kapitel 3

Konzept

In diesem Kapitel möchte ich Folgerungen aus der Analyse-Phase des Projekts veranschaulichen. Wir haben uns in dieser Phase zunächst mit den Vorgaben für das Projekt und bereits vorhandenen Erkenntnissen beschäftigt. Es wurden Informationen gesammelt und auf ihre Verwertbarkeit untersucht. Die Anforderungen wurden unter Rücksprache mit den verantwortlichen Betreuern angepasst. Die einzelnen Bereiche des Projekts wurden aufgeteilt, sodass jeder Beteiligte genügend Material für seine Arbeit hat. Gleichzeitig wurde darauf geachtet, dass die Themengebiete der verschiedenen Bereiche voneinander abgrenzbar sind.

3.1 Anforderungen an das Gesamtkonzept

Das Konzept sieht vor, dass Positionsdaten mit einem Android Gerät erstellt werden. Dazu wird die Android Plattform durch Hard- und Software Komponenten erweitert. Sie wird dabei so verändert, dass sie als vertrauenswürdig gilt. Dies geschieht durch den Einsatz von TC Technologie. Die Authentizität der Daten wird durch mehrere Verfahren gesichert. Ein Verfahren davon ist eine Hardware Komponente zur Absicherung und Signierung. Ein TPM-Chip. Mit diesen Anforderungen wird eine Manipulation des Geräts nicht ausgeschlossen, aber sie kann erkannt werden.

3.2 Szenarien

In dieser Phase der Arbeiten wurden verschiedene Szenarien in Betracht gezogen.

- Komplexe Lieferketten-Systeme
  Eine Spedition möchte die Position ihrer Fahrzeuge belegbar erfassen. In Verbindung mit Sensoren ist neben der Positionsbestimmung die Erfassung weiterer Daten möglich. Bei der Verwendung von Temperatursensoren kann überwacht werden, ob die Kühlkette während des Transports eingehalten wird. In der Datenstruktur ist eine HashMap für die Sensordaten vorgesehen, somit kann eine beliebig Anzahl verschiedener Sensoren genutzt werden.

- Automatisierte Produktionsstätten
  Das nächste Szenario kommt aus der Industrie. In einer automatisierten Produktion werden viele Netzwerkknoten eingesetzt. Diese müssen vor Angriffen über das Netzwerk geschützt werden. Angriffe auf die Hardware sind hier meist nicht ohne größeres Aufsehen zu bewerkstelligen. Aber Software lässt sich oft schnell

und recht unauffällig austauschen. In manchen Fällen lässt sich die Veränderung im Nachhinein nicht mehr nachweisen.

- Wegführung auf dem Campus zu Büros und Hörsälen
  Es ist möglich mit Hilfe von W-LAN Ortung die Position zu bestimmen. Dazu kann man dann noch die Türschilder mit RFID-Chips versehen auf denen Informationen über die Raumbelegung hinterlegt sind. Diese Chips können über Near Field Communication (NFC) ausgelesen werden.

- Transport von medizinischen Gütern
  Der Transport von Blutkonserven, Spenderorganen oder Medikamenten innerhalb, oder zu einem anderen Krankenhaus ist ein weiteres denkbares Szenario. Ein solcher Transport unterliegt strengen Auflagen in Bezug auf Transportbehälter und Temperaturüberwachung. Blutkonserven haben ohne Kühlung eine sehr begrenzte Haltbarkeit haben. Wenn die einzelnen Konserven mit einem RFID-Chip ausgestattet wären, könnte man nachverfolgen, wann der Behälter aus der Kühlung entnommen wurde. Somit wäre eine Überprüfung möglich, ob die Konserve wieder eingelagert werden darf.

Da wir uns mit der Android-Plattform beschäftigen, wird das Szenario der Lieferkette umgesetzt. Es wäre auch möglich, eine große automatisierte Produktionsstätte in Betracht zu ziehen. Da sich eine solche nicht ohne hohen Aufwand nachstellen lässt, würde dieses Szenario den Rahmen unserer Möglichkeiten sprengen. Gleichzeitig gibt es an der Hochschule ein anderes Projekt, welches sich mit dem Thema der Wegfindung auf einem Campus befasst. Diese Projekt läuft unter dem Namen MoCaInfo[1].

[1] Quelle: http://homepages.thm.de/~hg10187/mocainfo-prj-skizze.pdf

Kapitel 4

Feinkonzept

Nachdem das Szenario, welches umgesetzt werden soll gefunden ist, können wir uns nun mit den Einzelheiten des Projekts beschäftigen. Es muss festgelegt werden, welche Entwicklungsumgebung genutzt werden soll. Außerdem müssen die Versionen der verwendeten Software abgestimmt werden. Jede Art von Daten, welche erstellt werden, müssen in einer festgelegten Struktur gespeichert werden. Alle Beteiligten müssen sich an die Struktur halten, damit die einzelnen Komponenten untereinander kommunizieren können.

4.1 Datenstruktur

Um Daten übertragen zu können, muss im Vorfeld eine Datenstruktur festgelegt werden. Die einzelnen Variablen wurden in der Planungsphase vor der Implementierung festgelegt. Die Datenstruktur aus dem Projekt TruCam diente uns als Vorlage. Diese wurde von uns überarbeitet und auf die Eigenschaften des Projekts angepasst. Die in der Datenstruktur von TruCam enthaltenen Variablen für Bilddaten und Kommentare werden nicht weiter benötigt. Im Gegenzug wird Platz für Sensordaten und Daten zur Optimierung der Bandbreite benötigt.

Innerhalb dieser Struktur gibt es verschiedene Nutz- und Administrationsdaten. Da nicht immer alle Daten benötigt werden, wurde die Datenstruktur in 3 Bereiche aufgeteilt.

Msg-Handler	Publisher-ID
Metadaten	Meta: { ... }
Attestierungsblock	Attestation: { ... } TA: { ... }
Signaturblock	Signature: { ... }

Tab. 4.1: Bereiche der Datenstruktur

Eine Übersicht aller verwendeten Variablen befindet sich im Anhang (Anhang B.1, S.48). Der Bereich “Metadaten” enthält Nutzdaten, die in jedem Datenpaket enthalten sein müssen. Diese Nutzdaten bestehen aus den GPS Koordinaten, falls vorhanden, den Sensorwerten und verschiedenen Daten zur Verkettung der Datenpakete. Zusätzlich ist

in jedem Paket die Publisher-ID vorhanden. Sie wird vom MAP-Server aus Daten, welche im Zertifikat hinterlegt sind generiert. An dieser erkennt der MAP-Server welcher Evidence Generator die Daten gesendet hat und kann die einzelnen Datenpakete zu einem Gesamtbeweis zusammenfassen. Die Bereiche “Attestierungsblock” und “Signaturblock” hingegen werden nur bei signierten Paketen mit versendet. Normalerweise wird im Signatur-Quote eine Zufallszahl als sogenannte “Nonce” mitgesendet. Diese Nonce wird in unserem Projekt durch die Variable “Geheimnis” repräsentiert und befindet sich im Attestierungsblock der Datenstruktur. Die Nonce wird dazu verwendet, die einzelnen signierten Datenpakete durch diesen Freshness-Wert zusätzlich miteinander zu verknüpfen. Jedes mal wenn ein Datenpaket signiert wird, wird zusätzlich eine Attestierung ausgelöst. Die Attestierungsdaten werden mit den in den Metadaten enthaltenen Informationen signiert. Es ist wichtig bei allen ausgeführten Aktionen auf die Reihenfolge zu achten (Ordering Problem). So sollte immer zuerst das Quote erzeugt und danach mit einem Zeitstempel versehen werden. Dadurch kann man gewährleisten, dass keine unbekannte Zeitspanne zwischen Quote und Erzeugung des Zeitstempels liegt. Des weiteren ist somit gesichert, dass der Quote in einer vertrauenswürdigen Umgebung erstellt worden ist.

4.2 Datengenerator

Mit dem Datengenerator (Evidence Generator) werden im Projekt TruLoc Positionsdaten ermittelt. Alle Messwerte werden in die dafür vorgesehenen Felder der Datenstruktur gespeichert. Um sicherzustellen, dass es sich bei dem EG um eine vertrauenswürdig Plattform handelt, wird ein Messsystem namens IMA (Integrity Measurement Architecture) verwendet. Mit diesem System werden alle auf dem Gerät verwendeten Programme und Bibliotheken gemessen und ein Hashwert gebildet, welcher in der SML gespeichert wird.

Es wird eine Möglichkeit benötigt kontinuierlich die Positionsdaten vom GPS Baustein des verwendeten Geräts zu ermitteln. Hierfür wird ein im Hintergrund laufender Service verwendet. Im Gegensatz zu dem bei TruCam verwendeten EG, der eine Verbindung zum MAP-Server bei jeder Übertragung neu aufgebaut hat, ist es je nach Skalierungsfaktor sinnvoll eine dauerhafte Verbindung zum Server zu erstellen. Dafür wird ein zweiter Service implementiert.

Um einen Datenverlust bei Abbruch der Netzverbindung zu verhindern, benötigt man noch eine lokale Datenbank als Zwischenspeicher.

Es gibt verschiedene Möglichkeiten die Daten auf dem Gerät abzulegen. Zum einen den internen Speicher, zum anderen besteht die Möglichkeit eine SD-Karte zu verwenden. Beide Möglichkeiten haben ihre Vor- und Nachteile. Eine SD-Karte hat den Vorteil, dass sie mehr Speicherplatz als der interne Speicher bieten kann. Jedoch hat sie den Nachteil, dass sie leicht aus dem Gerät entfernt und somit gestohlen oder kopiert werden kann. Gleichzeitig ist es sinnvoll erstellte Daten so kurz wie möglich auf dem Gerät zu belassen. Auf einem Server in gesicherter Umgebung sind die Daten besser aufgehoben, als auf einem mobilen Gerät. Zudem ist die zentrale Speicherung effizienter, als die Datensätze auf vielen verschiedenen mobilen Geräten zu haben.

4.3 Benutzeroberfläche

Die Oberfläche des Evidence Generators ist relativ schlicht und funktional gehalten. Sie ist in drei Teilbereiche untergliedert. Der mittlere und größte Bereich besteht aus einer Karte, auf der die zuletzt gesendete Position angezeigt wird. Dabei wird auf

das Kartenmaterial von Google Maps© zurückgegriffen, da diese Daten unter Android standardmäßig zur Verfügung stehen.

Im oberen Bereich werden die Einstellungen für die Übertragungsabstände und die Skalierung angezeigt. Hier ist ebenfalls ein Schalter vorgesehen, mit dem die aktuelle Beweisführung beendet und eine neue gestartet werden kann. Da es sich um einen On/Off Schalter handelt lässt sich sofort erkennen, ob eine Beweisführung aktiv ist.

Der untere Bereich ist ein Scrollbereich, in dem die ausgeführten Operationen aufgelistet werden. Dies stellt ein Log der durchgeführten Aktionen dar, in dem der Benutzer nachvollziehen kann, welche Aktionen der EG durchführt. Gleichzeitig werden hier auch eventuell auftretende Fehlermeldungen dokumentiert.

Die verschiedenen Programmeinstellungen können über das Applikations-Menü vorgenommen werden. In den Einstellungen kann die Adresse des verwendeten MAP-Servers eingetragen oder verändert werden. Des weiteren werden hier die benötigten Zugangsdaten für den Server eingetragen. Außerdem können hier die Einstellungen für den Skalierungsfaktor und das Zeitintervall für die zu versendenden Pakete eingestellt werden.

4.4 Datenerfassung

Die benötigten Positionsdaten werden durch das im Handy integrierte GPS erzeugt und per IF-MAP Protokoll an den irond Server übertragen. Dort können sich weitere Komponenten registrieren, welche Daten empfangen wollen. Jedes gesendete Datenpaket ist ein Teilstück eines Gesamtbeweises. Es muss darauf geachtet werden, dass kein Datenpaket verloren geht. Dies könnte auf Grund einer nicht vorhandenen Netzverbindung geschehen. Falls keine Netzverbindung besteht, werden die Daten in der lokalen Datenbank zwischengespeichert und wenn die Verbindung wieder hergestellt ist an den MAP-Server gesendet. Während der Routenverfolgung kann es dazu kommen, dass kein GPS Signal empfangen werden kann. Dies geschieht meist in Tunnels oder anderen betonierten Bauwerken wie Tiefgaragen. In diesen Fällen werden die für GPS Daten vorgesehenen Variablen mit dem Wert “NULL” gefüllt. Somit ist sichergestellt, dass zumindest die Sensordaten weiterhin erfasst und übertragen werden.

4.4.1 Verfahren

Zum Versenden von Daten geht der Evidence Generator wie folgt vor:

1. Booten
2. Synchronisation der Uhrzeit
3. Bindung des Tickstamp an die Uhrzeit unter Verwendung der TA
4. Erstellung eines Schlüsselpaars
5. (optional) Verifizierung des AIK mit der PCA
6. Erzeugung und Austausch der initialen “Geheimnis”-Nonce
7. Referenz Metrik für lokale Attestierung
8. lokale Attestierung

Dieser Ablauf muss zur Sicherung der Vertrauenswürdigkeit durchgeführt werden. Es wird ein TPM-Quote erzeugt. Dieser wird benötigt, um den Systemzustand während der Laufzeit abfragen zu können. Der Zustand ist Bestandteil der Signierungs- und Attestierungsinformationen. Die Attestierungsinformationen werden mit dem Zeitstempel, bestehend aus aktuellem Tickstamp und synchronisierter Tickstamp-Uhrzeit inklusive Zertifikat, sowie dem Freshnesswert erweitert.

4.4.2 Beweiskette

Um die Beweiskette zu sichern, müssen verschiedene Faktoren berücksichtigt werden. Zum einen muss die Datenerzeugung auf einer vertrauenswürdigen Plattform geschehen. Dazu ist der Einsatz einens TPM-Chips unumgänglich. Mit diesem Chip werden die einzelnen Abschnitte des Bootvorgangs gemessen. Es werden Hashwerte der einzelnen Komponenten erstellt und in den PCR Registern des TPM-Chip hinterlegt. Sobald der Kernel des Betriebssystems gestartet ist, werden die weiteren Messungen vom IMA System übernommen. Diese Werte werden dann im Stored Mesurement Log (SML) gespeichert. Als Hashverfahren kommt der SHA-1 Algorithmus zum Einsatz. Dieser gilt allerdings als gebrochen und wird somit zukünftig nicht mehr als sicher eingestuft sein. In zukünftigen Spezifikationen der TCG wird eine Erweiterung der möglichen Hash-Algorithmen nötig sein.

Um später Vergleichswerte für diese Log Datei zu haben, muss bei der Initialisierung eines neuen EGs eine Messung erfolgen, bei dem sich das Gerät in einem vertrauenswürdigen Zustand befindet. Die so gewonnenen Daten müssen für den späteren Vergleich gespeichert und im Archiv hinterlegt werden. Somit ist ein Grundstein für vertrauenswürdige Daten gelegt (Root of Trust). Die Beweiskette selbst wird durch mehrere Verfahren abgesichert.

- Zum einen durch die Signierung der einzelnen Datenpakete.
  So kann später nachgewiesen werden, ob es sich noch um die ursprünglich erstellten Daten handelt. Eine Manipulation kann zwar nicht verhindert, aber aufgedeckt werden. Im Rahmen der Minimierung des verwendeten Datenvolumens wird nicht mehr jedes Datenpaket signiert, sondern mehrere Pakete zusammengefasst. Die Anzahl der Pakete wird über einen im EG einstellbaren Scale-Wert festgelegt.

- Um die Absicherung der Zwischenpakete zu gewährleisten, wird ein weiteres Verfahren verwendet. Die Datenpakete werden durch eine Hashliste miteinander verknüpft. Durch diese Verknüpfung kann bei der Verifikation festgestellt werden, ob alle Datenpakete der Streckenführung vorhanden sind. Hierbei gibt es wiederum zwei Varianten:
  Zum einen kann ein fortlaufender Hashwert, sogenannte konkatenierte-Hashes gebildet werden. Hierbei werden die einzelnen Pakete miteinander verknüpft. Dabei wird der Hashwert des aktuellen Pakets mit dem der bisher versendeten verknüpft. Somit wird immer nur ein Wertepaar übertragen. Dies besteht aus den Paketnummern und dem Hashwert.

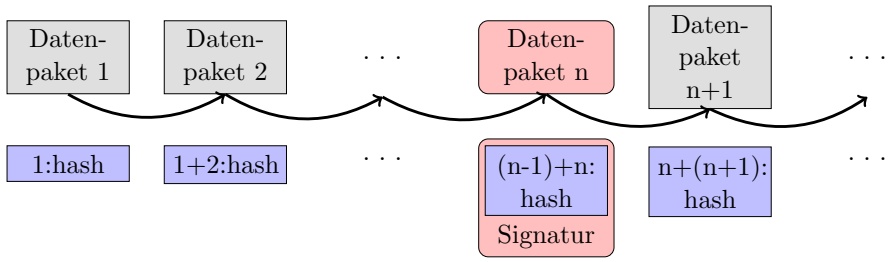

Abb. 4.1: Verkettung: konkatenierte Hashes, Scale-Value: n

Ein so erstellter Hashwert kann auch über ein Paket mit Signatur hinweg weitergeführt werden. Dies hat den Vorteil, dass alle Pakete einer Beweisführung miteinander verknüpft sind. Die Verifizierung erfolgt, indem man die Verkettung zurückrechnet. Das bedeutet, die einzelnen Paket-Hashwerte werden vom Gesamthash abgezogen, ausgehend vom zuletzt übertragenen Paket. Theoretisch

wird durch dieses Verfahren die größtmögliche Ersparnis erreicht. In der Praxis muss allerdings der Verlust von Datenpaketen in Betracht gezogen werden. Sollte dies geschehen, kann die Beweiskette nicht zurückgerechnet werden. Der Hashwert des verlorenen Pakets kann nicht nachträglich ermittelt werden. Um dieses Problem zu umgehen, ist es notwendig dem eigentlichen Hash eine Anzahl von Vorgänger-Hashwerten mitzugeben. Die genaue Anzahl muss noch statistisch ermittelt werden.

Es kann aber auch eine Hashliste aufgebaut werden. Bei diesem Verfahren wird eine Tabelle angelegt und zu jedem gesendeten Paket ein Eintrag bestehend aus Paketnummer und Hashwert gespeichert. Dies hat den Vorteil, dass jedes Paket einzeln überprüft werden kann und die Rückrechnung auch noch möglich ist, falls zwischendurch ein Datenpaket fehlen sollte. Der Scale Wert legt fest, für jedes wievielte Paket eine Signatur erstellt wird. Zusätzlich ist es nach einer Signierung sinnvoll eine neue Tabelle zu beginnen. Dies verhindert, dass in der Tabelle zu viele Einträge vorhanden sind. Dabei könnte man die Hashliste der bisherigen "n"-Pakete erneut hashen und mit diesem Wert die neue Tabelle beginnen. So wird eine Verbindung der einzelnen Listen gewährleistet.

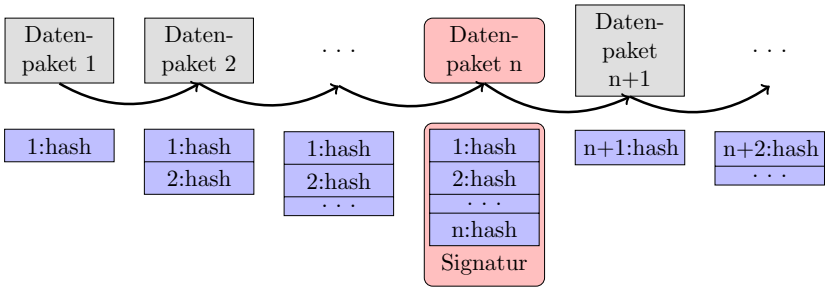

Abb. 4.2: Verkettung: Hashliste, Scale-Value: n

- Zusätzlich wird das TPM-Quote mitgesendet.
  Er enthält u.a. das Measurement Log anhand dessen eine vertrauenswürdige Umgebung bestätigt werden kann. Das Measurement Log sagt allerdings nur aus, in welchem Zustand sich das System zum Zeitpunkt des booten befand. Es können später Applikationen gestartet oder installiert worden sein, welche das System kompromittieren.

- Um sicherzustellen, dass kein falsches Datenpaket eingeschleust werden kann, werden jedes mal zwei Werte zur Erkennung mitgesendet. Der erste Wert, "Sec_new" genannt, ist ein vom EG generierter Zufallswert, der mit einem Datenpaket mitgesendet wird. Das darauf folgende Paket enthält immer den Zufallswert des vorhergehenden, gespeichert in der Variable "Sec_old" und einen neuen Wert in "Sec_new". So kann sichergestellt werden, dass die einzelnen Datenpakete in der richtigen Reihenfolge verkettet werden.

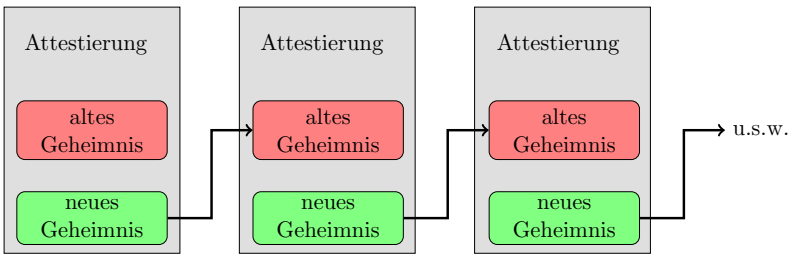

Abb. 4.3: Verkettung der Geheimnisse

Begonnen wird diese Verkettung mit einem initialen Geheimnis, dies muss auch dem Verifizierer bekannt sein. Dadurch wird verhindert, dass eine komplette Beweiskette auf einem gefälschten Geheimnis aufgebaut wird. Die folgende Aufzählung und Grafik zeigt eine Übersicht der Verkettung von Beweisdaten:

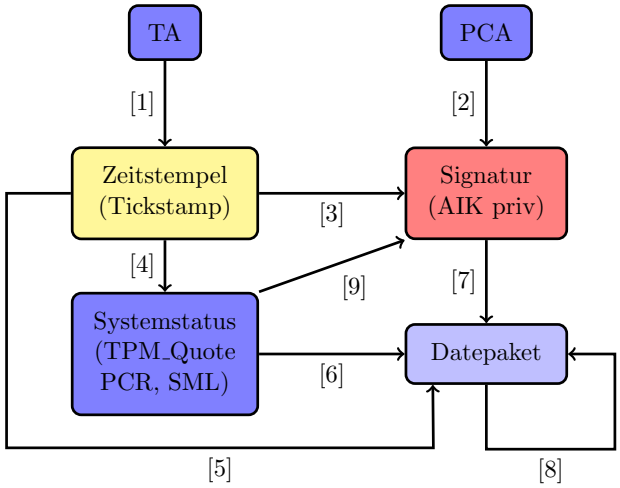

Abb. 4.4: Übersicht der Beweisverkettung

1. Die TA liefert den synchronisierten Zeitstempel
2. Die AIK Schlüsselpaare werden von der PCA zertifiziert
3. Die Signatur wird an den Zeitstempel gebunden
4. Freshness Wert für die Erzeugung der Attestierungsinformationen
5. Datenpakete werden an den Zeitstempel gebunden
6. Datenpakete werden an die Attestierung gebunden
7. Datenpakete werden signiert
8. Datenpakete werden untereinander über eine Hashliste verbunden
9. Die Signatur bzw. ein AIK wird an den Systemstatus gebunden (Quote)

4.4.3 Signierung

Die zu sendenden Datenpakete werden unter Zuhilfenahme eines AIK des TPM-Chip signiert und somit die Authentizität der Daten belegt.

- Als erstes wird ein LocationPackage angelegt. Dies wird mit den gesamten Daten gefüllt
- Ein TPM-Kontext Objekt anlegen, um auf den Chip zugreifen zu können
- Einen RSA Key vom KeyManager besorgen, der AIK
- Zertifikate vom CertManager holen
- Die Zertifikate im Beweisblock des Pakets speichern
- SHA-1 Hash vom Beweisblock des Pakets berechnen

- Den Hash mit der TPM Funktion “Quote” signieren
- Den Quote und die zugehörige PCR-Variable im Signaturblock des Pakets speichern
- SHA-1 Hash des Quote erzeugen
- Der erzeugte Hash wird mit der TPM Funktion “Tickstamp” signiert
- Der Tickstamp wird mit dem zugehörigen CurrentTicks Wert im Signaturblock gespeichert

Eine genaue Beschreibung des Signierungsprozesses ist im Kapitel 8, S.29 zu finden.

4.5 Skalierung

Das Versenden der Daten beansprucht eine gewisse Bandbreite. Beim Einsatz von wenigen Evidence Generatoren ist dieses kein hoher Kostenfaktor. Aber wenn das System im Umfeld eines größeren Unternehmens eingesetzt werden soll, wie z.B. bei einer Spedition, muss man den Faktor Datentransfer genauer betrachten. Bei einer Übertragung werden verhältnismäßig wenig Nutzdaten übermittelt. Dem gegenüber steht ein recht hoher Anteil an administrativen Daten. Es werden zwar insgesamt nur kleine Datenpakete übertragen, aber wenn das System mit einigen 100 oder mehr EGs genutzt wird, können schnell hohe Unterhaltskosten für den Betreiber entstehen. Insbesondere wenn zur Übertragung eine mobile Datenverbindung benötigt wird. Falls das System in der industriellen Produktion in einem LAN eingesetzt wird, können die vielen gleichzeitigen Anfragen z.B. an die Timing Authority zu einer recht hohen Netzlast führen. Um solche Engpässe zu umgehen, muss man sich mit dem Thema Skalierung auseinander setzen. Welche Daten müssen in jedem Datenpaket vorhanden sein? Kann man die Administrationsdaten auf bestimmte Datenpakete beschränken?

4.5.1 Vorgehensweise zur Bandbreitenoptimierung

Ohne Skalierung würden bei allen Datenpaketen immer alle administrativen Daten mitgesendet. Nun gilt es zu überlegen, ob diese Daten in allen Paketen benötigt werden. Bei der Analyse sind wir zu dem Entschluss gekommen, dass aufgrund der Verknüpfung der einzelnen Datensätze nicht immer alle administrativen Daten mitgesendet werden müssen. Aus diesem Grund ist es möglich, in der EG Applikation einen Skalierungsfaktor einzustellen. Mit diesem Wert wird festgelegt, jedes wievielte Paket mit allen Daten ausgestattet ist und signiert wird. Falls die Hashtabelle verwendet wird, ist zu überlegen, ob man nach einem signierten Paket mit einer neuen Tabelle beginnt, Sollte man die Tabelle vom Anfang einer Wegführung bis zum Ende durchgehend nutzten, wird die Tabelle je nach Strecke sehr lang. Somit sollte man nach einer Signierung einen Hash über die gesamte Tabelle erzeugen und damit eine neue Tabelle beginnen.

4.6 Verifikation

Für die Verifikation ist der Mobile Inspector zuständig. Er holt sich die dafür nötigen Daten aus dem Archiv. Wie schon zuvor im Projekt TruCam, wird der MI über RESTful-Webservices mit dem Archiv kommunizieren, da die Funktionalität des MAP-Servers nicht benötigt wird. Diese Daten werden in einer Live-View angezeigt und dann auf ihre Echtheit überprüft. Die Überprüfung geschieht im Hintergrund. Nach Abschluss

wird der auf der Karte angezeigte Punkt je nach Status eingefärbt. Sollten alle Werte für gut befunden werden, wird der Punkt grün angezeigt, ansonsten rot.

4.6.1 Signierungs-Prüfung

Als erstes wird die Vertrauenswürdigkeit des Evidence Generators überprüft, mit welchem die Beweisdaten erzeugt wurden. Dabei wird überprüft, welche Applikationen auf dem Gerät installiert oder aktiv sind und ob diese als vertrauenswürdig eingestuft werden können. Dies stellt den Grundstein für die Beweisketten dar. Danach werden die signierten Pakete auf ihre Korrektheit überprüft. Es wird wie folgt vorgegangen:
Zunächst muss man aus dem Paket den öffentlichen Schlüssel des EGs extrahieren. Mit diesem Schlüssel können dann die Signaturen überprüft werden. Dies ist der erste Schritt um den Gesamtbeweis zu verifizieren. Als nächstes muss überprüft werden, ob die einzelnen Datenpakete korrekt miteinander verkettet sind. Dazu mehr im nächsten Abschnitt.

4.6.2 Beweisketten-Prüfung

Die Hashliste stellt die Verknüpfung der einzelnen Datenpakete dar. Um sicherzustellen, dass keines der Datenpakete verändert wurde, müssen alle nacheinander überprüft werden. Hierbei ist je nach Hashverfahren unterschiedlich vorzugehen. Handelt es sich um eine Hashtabelle mit Name - Wert Paaren, können die Pakete einzeln überprüft werden. Jedoch kann man zuvor die signierten Teilbereiche verifizieren. Des weiteren ist es sinnvoll, nach jedem signierten Datenpaket einen Hashwert der bisherigen Liste zu erstellen und damit eine neue Hashtabelle zu beginnen. Geschieht dies nicht, kann es bei einer langen Routenverfolgung zu einer sehr umfangreichen Hashtabelle führen. Falls es sich bei der Verkettung um einen fortlaufenden Wert handelt, müssen alle Pakete nacheinander überprüft werden. Sollte bei dieser Verkettung zwischendurch ein Paket verloren gegangen sein, ist eine Überprüfung der darauf folgenden Pakete nicht mehr möglich.

Teil III

Realisierung eines Prototypen

Kapitel 5

Datenerfassung

Nachdem ich mich im letzten Kapitel mit der Planungsphase beschäftigt habe, geht es nun um die Realisierung. Das Projekt ist in verschiedene Bereiche aufgeteilt. Ich habe mich zunächst um die Anpassung des Betriebssystems gekümmert und die benötigten Images für das Samsung Galaxy S II erstellt. Danach wurde von mir der Datengenerator gebaut. Weitere Bereiche sind das Archiv, die Verifikation und die Skalierung. Diese Bereiche wurden von anderen Mitgliedern der Arbeitsgruppe realisiert.

5.1 Problemstellung

Die Daten werden auf einem Android Gerät ermittelt. Hierbei gilt es festzulegen, in welchem Abstand die Erfassung geschehen soll und welche Daten genau übertragen werden müssen. Es gibt zwei Werte an denen eine Veränderung festgehalten werden kann, die Entfernung zur letzten Position, oder eine vergangene Zeitspanne seit der letzten Erhebung. Bei dem gewählten Szenario muss im Archiv eindeutig feststellbar sein, von welchem Gerät die Daten stammen und zu welcher Route sie gehören.

5.2 Lösungsansatz

In jedem Datenpaket wird eine eindeutige Publisher-ID mitgesendet. Vergeben wird sie durch den MAP-Server und setzt sich aus Komponenten der verwendeten Zertifikate zusammen. Mit dieser ID kann jedes Datenpaket einem Evidence Generator zugewiesen werden. Um feststellen zu können wie viele Pakete zu einer Streckenführung gehören, enthält jeder Datensatz eine fortlaufende Paketnummer. Zur Feststellung mit welchem Datenpaket eine Strecke beginnt und endet, wird in jedem Paketen ein Flag gesetzt. Dieses Flag kann drei Werte annehmen, Start-, Stop- oder Zwischenpaket.

Kapitel 6

Betriebssystemanpassung

Um das Betriebssystem gegen unbefugte Veränderungen zu schützen, müssen verschiedene Veränderungen vorgenommen werden. Als Basis für diese Veränderungen dient uns das Vogue Projekt.[1] Dieses Projekt beschäftigt sich mit dem Thema "Vertrauenswürdiger mobiler Zugriff auf Unternehmensnetze". Für die Realisierung wird eine ebenso vertrauenswürdige Plattform benötigt, wie bei unserer Datenerfassung.

6.1 Problemstellung

Im Moment gibt es auf dem Markt kein Handy mit integriertem TPM-Chip. Somit muss die Funktionalität mit anderen Mitteln hergestellt werden. Aus diesem Grund wird das Betriebssystem so angepasst, dass es mit einem TPM-Emulator die Funktionalität nachbildet. Während der Entwicklung bietet ein Emulator den Vorteil, dass er Fehlermeldungen aufzeigt. Ein später genutzter echter TPM-Chip bietet diese Möglichkeit nicht.

6.2 Lösungsansatz und Umsetzung

Es existiert bereits eine Version des Android Betriebssystems mit angepassten Sicherheitsfunktionen. Diese wurde unter anderem von Stephan Heuser entwickelt, er ist Mitarbeiter des Fraunhofer SIT Instituts. Die Master Thesis von Herrn Johannes Westhuis [Wes10] beschäftigt sich ebenfalls mit dem Thema. Die von mir verwendeten Patches für das Betriebssystem und den Kernel wurden im Rahmen seiner Master Thesis erstellt. Als Basis für das angepasste Betriebssystem werden die Quellen des CyanogenMod[2] 7 genutzt. Die Version des Fraunhofer Instituts ist für Android 2.2 (Froyo) ausgelegt und konnte uns somit zunächst nur als Vorlage dienen. Das Projekt TruLoc wird aufgrund der vorgegebenen Anforderungen und der uns zur Verfügung stehenden Hardware für Android 2.3.x (Gingerbread) entwickelt. Zunächst musste die Entwicklungsumgebung um verschiedene Bibliotheken und Compiler/Linker erweitert werden, sodass es möglich wurde das Betriebssystem zu compilieren. Danach wurde aus den uns zur Verfügung gestellten Quelltexten ein generisches Image des Betriebssystems zusammengestellt und compiliert. Diese konnte sofort mit einem Android Emulator gestartet werden. Dann konnte die TPM Funktionalität getestet werden und es konnte Kontakt zum TPM-Emulator hergestellt werden. In den uns vorliegenden Sourcen sind Vendor-spezifische Daten für das Google Nexus One (HTC Passion) vorhanden. Diese Daten sind im offiziellen Source-Tree von Android enthalten. Es lassen sich damit

[1] Vogue Projekt: http://www.vogue-projekt.de/
[2] CyanogenMod: http://www.cyanogenmod.com/

die spezifischen Images für das Handy erstellen. Leider konnten wir diese Images nicht testen, da gerätespezifische Images nicht auf dem Android Emulator gestartet werden können. Uns stand auch kein Handy zur Verfügung um die Software darauf zu installieren. Somit war der nächste Schritt Images für uns zur Verfügung stehende Hardware zu erstellen. Für die von uns verwendete Hardware ist standardmäßig Android 2.3.3 vorgesehen. Zu der neueren Version des Betriebssystems gehört auch immer ein neuer Systemkern, ein neuer Kernel. Bevor ich versuchte die vorhandenen Android Patches auf die Sourcen von Android 2.3.3 anzuwenden, verwendete ich die Kernel Patches um einen aktuellen Kernel mit TPM Funktionalität zu versehen. Die Patches sind für den Kernel mit der Version 2.6.29.1 angefertigt worden.

Kernel	Android	
2.6.29	1.6	(Donut)
2.6.32	2.2.x	(Froyo)
2.6.35	2.3.x	(Gingerbread)
3.0.8	4.x	(Ice Cream Sandwich)

Tab. 6.1: Übersicht Android- und Kernel Versionen

Wie man der Tabelle entnehmen kann, gehört der Kernel zu einer recht alten Android Version, nämlich zur Version 1.6 (Donut). Somit stellt der Kernel schon einen Backport[3] dar. In den von Fraunhofer zur Verfügung gestellten Sourcen ist ein 2.6.32er Kernel enthalten, auf den bereits die Patches angewendet wurden. Für das Galaxy S II wird ein 2.6.35er Kernel verwendet. Diesen versuchen wir nun für unsere Zwecke anzupassen. Zwischen den beiden Kernel Versionen wurde vieles an der bereits im Kernel integrierten Integrity Measurement Architecture (IMA) verändert. Somit ist es zu aufwendig die uns vorliegenden Kernel-Patches auf die neue Version anzupassen. Daher behalten wir für unseren Kernel die ältere Version von IMA bei und integrieren diese in den aktuellen Kernel. Abgesehen von IMA wird an den vorhandenen Sourcen nicht viel verändert, sondern neue Module und Funktionen, wie der TPM Emulator, hinzugefügt. Um den Kernel auf eine aktuelle Version zu bringen, wurden die Patches von uns auch auf einen 3.1.5er Kernel angewendet. Dies war nach kleineren Anpassungsarbeiten möglich. Der geringe Aufwand lässt sich darauf zurückführen, dass sich an den Kernel-Sourcen in dem Bereich, den wir für unsere Zwecke anpassen, von offizieller Seite her nicht mehr viel verändert hat. Der 3.1.5er Kernel lies sich compilieren, sodass es auch auf einem aktuellen Kernel möglich ist, die von uns benötigten Änderungen zu realisieren.

Nachdem ein aktueller Kernel zur Verfügung steht, musste noch das Android System selbst angepasst werden. Somit wurden als nächstes die Vendor-spezifischen Daten für das Samsung Galaxy S II organisiert. In den Dateien des CyanogenMod ist ein Skript enthalten, mit dem sich die Daten aus einem Handy auslesen lassen. Um unter Linux auf die Daten des Handys zugreifen zu können, muss dies erst dem System bekannt gemacht werden. Hierzu muss mittels “lsusb” die Hersteller-ID und die Geräte-IDs ausgelesen werden und in einer Regel Datei hinterlegt werden. Die Geräte-IDs unterscheiden sich je nach Modus in dem sich das Handy befindet. So gibt es eine ID für den Normal, den Debug oder Recovery Modus. Diese Datei wird unter “/lib/udev/rules.d/51-android.rules” abgelegt. Der Inhalt der Datei sieht wie folgt aus:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="04e8", ATTRS{idProduct
    }=="6860", MODE="0666", OWNER="<username>" #Normal Galaxy
     S II
SUBSYSTEMS=="usb", ATTRS{idVendor}=="04e8", ATTRS{idProduct
    }=="685e", MODE="0666", OWNER="<username>" #Debug &
    Recovery Galaxy S II
```

[3]Als Backport bezeichnet man eine Modifikation einer Software auf eine ältere Version

Nach diesem Schritt ist es dem Computer möglich über ADB mit dem Handy zu kommunizieren. Es kann vorkommen, dass das System keinen Zugriff auf das angeschlossene Gerät bekommt. Dies kann über den Befehl “adb devices” überprüft werden, mit dem angeschlossene Geräte aufgelistet werden. Falls der Zugriff verweigert wird, ist es nötig als Root zu arbeiten und den ADB Server neu zu starten. Wenn man mit verschiedenen Geräten arbeitet, müssen die Einträge der Datei gesondert für jedes Gerät angelegt werden, da jedes Gerät über eigene IDs verfügt. Das beim CyanogenMod mitgelieferte Skript kann erst verwendet werden, wenn bereits eine Version der Modifikation auf dem Handy installiert ist. Ansonsten sind mehrere Dateien und Verzeichnisse nicht auf dem Gerät vorhanden. Auf der Suche nach einer alternativen Möglichkeit die Daten zu bekommen, fand ich in einem Forum einen Link zu einem Repository.[4] In diesem Repository liegen neben den Vendor-spezifischen Daten für eine Vielzahl von Geräte auch APK Dateien von Google Applikationen. Anhand dieser Informationen ist es möglich ohne Umwege die Vendor-spezifischen Daten herunterzuladen. Bevor diese Daten zum Einsatz kommen, habe ich zunächst die Android-Patches auf die Android 2.3.3 Sourcen angewendet. Bei den Patches handelt es sich um:

- Anpassungen an der DalvicVM
- Erweiterungen des init Prozess
- Erweiterungen des Install Prozess
- es wird eine Metric in das System eingebaut
- ein TPM Treiber wird integriert
- Trousers - eine Trusted Computing Software Stack Library
- libiconv - Daten zur Internationalisierung
- langinfo - Funktion zur sprachlichen Localisierung
- TPM Tools - Anwendungen für den Zugriff auf den TPM-Chip
- Anpassungen and GMP Library

Die Anwendung der Patches war bis auf wenige “rejects”, welche schnell behoben werden konnten, möglich. Während dem Compilieren tauchten dann weitere Fehler auf. So musste in den neu hinzugefügten Modulen noch Tags gesetzt werden, damit das Modul bei einem Compiliervorgang berücksichtigt wird. Nachdem alle Module mit Tags versehen waren, ist leider ein weiterer Fehler aufgetaucht, welcher sich nicht ohne weiteres beheben ließ. Dieser Fehler passierte vermutlich aufgrund dessen, dass sich zwischen Android Version 2.2 und Version 2.3 die verwendete Java Version von 5 auf 6 geändert hat. Um nicht zu viel Zeit für das Anpassen des Android Systems aufzuwenden, wurde an dieser Stelle beschlossen, die Arbeiten daran zu beenden. Somit wurde wieder auf das bereits angepasste generische Image von Android 2.2 zurückgegriffen. Ein generisches Image wurde verwendet, da die Vendor-spezifischen Daten ebenfalls für Android 2.3.x gedacht sind.

[4] https://github.com/koush/proprietary_vendor_samsung

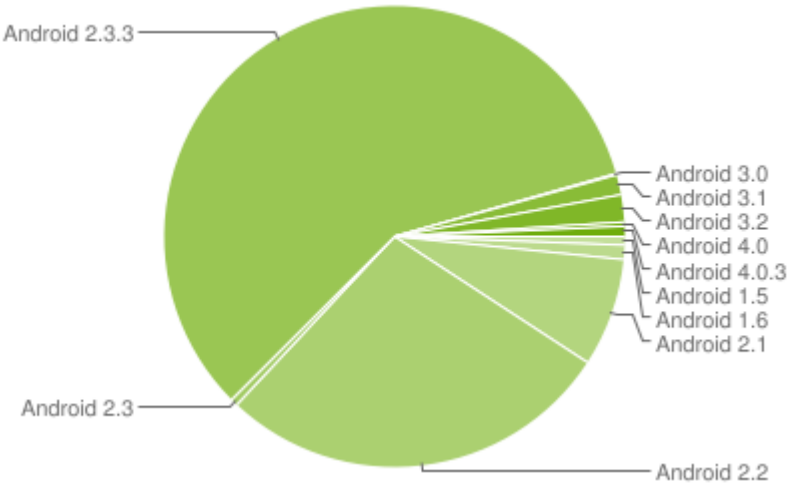

Abb. 6.1: Android Versionsverteilung

Plattform	**Codename**	**API**	**Anteil**
Android 1.5	Cupcake	3	0,6%
Android 1.6	Donut	4	1.0%
Android 2.1	Eclair	7	7,6%
Android 2.2	Froyo	8	27,8%
Android 2.3 - Android 2.3.2	Gingerbread	9	0,5%
Android 2.3.3 - Android 2.3.7		10	58,1%
Android 3.0	Honeycomb	11	0,1%
Android 3.1		12	1,4%
Android 3.2		13	1,9%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0,3%
Android 4.0.3		15	0,7%

Tab. 6.2: Android Versionsverteilung

Laut einer aktuellen Erhebung von Google wird Android 2.2 noch auf über 25% der sich im Umlauf befindlichen Geräten genutzt. Die Erhebung fand in einem Zeitraum von 14 Tagen zwischen dem 18. Januar und 1. Februar 2012 statt.[5] Aufgrund dieser Tatsache ist es gut vertretbar, dass wir auf den Android 2.2 Images weiter arbeiten.

[5] http://developer.android.com/resources/dashboard/platform-versions.html

Kapitel 7

Skalierbarkeit

Die Skalierbarkeit ist eines der Hauptthemen der Abschlussarbeit von Herrn Mark Schlüter. Da die Datengenerierung und die Skalierung sehr eng miteinander verknüpft sind, haben wir an dieser Stelle gemeinsam gearbeitet. Der Begriff Skalierbarkeit beschreibt in der Informatik normalerweise die Verwendbarkeit eines Algorithmus bezüglich der Menge an zu verarbeitenden Daten. Für das Projekt wird der Begriff in Bezug auf die Größe der zu übertragenden Datenpakete und Rechenzeit verwendet.

Herr Schlüter hat sich zunächst mit der Theorie des Themas Skalierung beschäftigt, um sich das für die Umsetzung nötige Wissen anzueignen. Die Umsetzung wurde anhand verschiedener Testapplikationen vorbereitet. Diese aus den Testapplikationen stammenden Funktionen wurden von uns in den bereits erstellten Evidence Generator integriert.

7.1 Problemstellung

In einem einstellbaren Abstand wird ein Datenpaket mit Positions- und Sensor Daten versendet. Zusätzlich ist ein nicht unerheblicher Teil an administrativen Daten in den Paketen enthalten. Die Größe der Nutzdaten ist in diesem Fall im Vergleich zu den administrativen Daten recht gering. Um nicht jedes mal den Overhead an Zusatzinformationen mitsenden zu müssen, ist eine Skalierung dieser Daten sinnvoll. Somit kann das Datenvolumen, welches übertragen wird, optimiert werden. Ein weiterer Faktor ist die Zeit, welche aufgewendet werden muss. Das Erstellen einer Signatur ist eine aufwendige Prozedur, welche natürlich eine gewisse Zeit in Anspruch nimmt.

7.2 Lösungsansatz

Es werden nicht alle Pakete mit einer Signierung versehen. So muss nicht jedes Mal die Zeit aufgewendet werden und dies spart einen Großteil der administrativen Daten ein. Um trotzdem die Vollständigkeit der Beweiskette nicht zu verletzten, werden die Pakete zwischen den signierten in einer "Hashtable" abgebildet. Mit einer solchen Hashtabelle kann sichergestellt werden, dass alle Datensätze die zu einer Beweiskette gehören, im Archiv abgelegt werden. In der Tabelle werden nicht die Nutzdaten direkt gespeichert, sondern wie der Name schon sagt, ein Hashwert der jeweiligen Daten. Durch die Werte aus der Tabelle kann der Mobile Inspector die Richtigkeit der einzelnen Datenpakete überprüfen.

Für die Hashtabelle gibt es verschiedene Lösungen. Zum einen ist es möglich eine Hashliste zu erstellen, in der Name-Wert Paare gespeichert werden. Zum anderen kann auch jeweils der aktuelle Hashwert mit dem neuen verknüpft werden. Dieses Verfahren nennt

man Konkatenation von Hashes. Jedes der Verfahren hat seine Vor- und Nachteile. Bei Verwendung der Hashliste können auch bei Verlust eines Datenpakets die restlichen Pakete weiterhin auf ihre Richtigkeit hin überprüft werden. Um dies zu erreichen müssen allerdings mehr Daten übermittelt werden, als bei bei dem zweiten Verfahren. Dieses Verfahren ist das sparsamste in Bezug auf die Datenmenge. Hierbei ist es allerdings bei Verlust eines Datenpakets nicht mehr möglich die darauf folgenden noch zu überprüfen.

Kapitel 8

Signierung

Um sicher zu stellen. dass die vorliegenden Daten auch die sind, welche vom EG generiert wurden, kann man mit der Signatur die Daten überprüfen. Eine nachträgliche Veränderung kann durch die Signatur nicht verhindert werden. Es ist nur möglich bei der Verifikation festzustellen, ob eine Veränderung geschehen ist. In dem für die Signierung verwendeten Paket befinden sich drei Klassen

- Prozessor
- KeyManager
- CertManager

Der Prozessor ist für die Signierung der Beweisdaten zuständig. Unter Android müssen Berechnungen, die eine längere Laufzeit haben in einem eigenen Thread laufen. Aus diesem Grund wird für den Prozessor das Runnable Interface implementiert. Um einen Beweis zu signieren, wird eine neue Instanz der Klasse aufgerufen. Der Konstruktor benötigt hierfür das Beweispaket, den Android Kontext, einen Handler und eine Callback Runnable. Der Kontext wird benötigt, um signierte Beweispakete an den MAP-Server zu senden. Danach wird von dem mitgegebenen Handler die Callback Runnable ausgelöst. Der KeyManager wird genutzt um AIKs zu generieren. Die Generierung eines Schlüsselpaars ist ebenfalls eine aufwendige Prozedur mit einem gewissen Zeitaufwand.

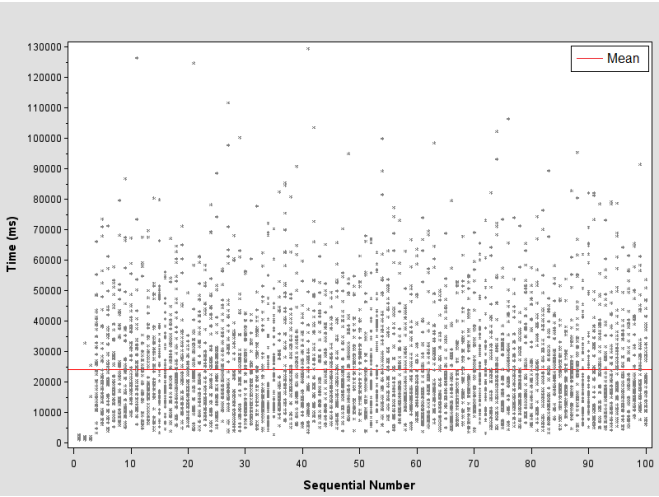

Abb. 8.1: AIK Experiment

Wie man der Grafik entnehmen kann, ist es sinnvoll sich bei der Verwendung von AIKs

zu beschränken. In dem vom Fraunhofer Institut durchgeführten Experiment wurden in 50 Schleifendurchläufen, ohne Pause, jeweils n=100 AIKs erzeugt. Wie man an der Verteilung erkennen kann, sollte man sich bei der Verwendung von AIKs auf möglichst nicht mehr als drei beschränken. Diese können mit einer annehmbaren Zeit von ca. 2,3 Sekunden erstellt werden. Sollten mehr AIKs benötigt werden, dauert es im Mittel 24 Sekunden um das Schlüsselpaar zu erzeugen. Es kann allerdings bis zu 130 Sekunden dauern. In unserer Anwendung beschränken wir uns auf ein einziges Schlüsselpaar. Im Anhang (Anhang B.2, S.49) ist die Grafik noch einmal in höherer Auflösung vorhanden. Der CertManager dient dazu die von der CA und TA erstellten Zertifikate zu verwalten. Die Kommunikation der Authorities wird über den MAP-Server realisiert. Alle Zertifikate werden als Singletons gespeichert. Ein Singleton ist global verfügbar, damit können alle Klassen einer Anwendung auf die Daten zugreifen. Gleichzeitig wird so gewährleistet, dass die Zertifikate den gleichen Lebenszyklus haben, wie der AIK zu dem sie gehören.

Kapitel 9

Benutzeroberfläche

Die Benutzeroberfläche ist ein wichtiges Kriterium für eine gut verwendbare Applikation. In diesem Kapitel möchte ich meine Gedanken vermitteln, wie ich mir eine solche Applikation vorstelle und umgesetzt habe. Ich beziehe mich dabei nur auf den Evidence Generator, da dieser von mir entwickelt wurde.

9.1 Problemstellung

Wie sollte eine ergonomische Oberfläche der Applikation aussehen?
Welche Daten sollen dem Benutzer auf der Hauptseite angezeigt werden?
Welche Daten müssen zusätzlich abrufbar sein?
Was muss einstellbar sein?

9.2 Lösungsansatz

Um eine gute Bedienbarkeit des Programms zu gewährleisten ist es wichtig, dass die Oberfläche nicht zu viele Informationen gleichzeitig darstellt. Aus diesem Grund werden nur die wichtigsten Informationen immer angezeigt. Um dem Benutzer nicht direkt die Wegführung anzuzeigen, wird zunächst eine "Startseite" eingeblendet.

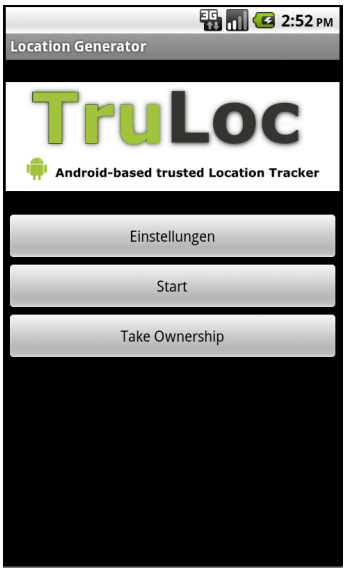

Abb. 9.1: Startseite

Standardmäßig ist jeder TPM-Chip deaktiviert. Um zu verhindern das der Chip über einen Remote Zugriff aktiviert wird muss das Kommando “Take Ownership” ausgeführt werden. Dies geschieht mit dem Button “Take Ownership”. Nachdem die Funktion einmal erfolgreich ausgeführt wurde, wird der Button deaktiviert und wird nicht mehr benutzbar sein. Dies ist eine Sicherheitsmaßnahme und soll verhindern, dass die Bindung des TPM-Chip zu einem späteren Zeitpunkt verändert werden kann. Hinter dem Button “Einstellungen” verbirgt sich ein Menü in dem die Zugangsdaten für den MAP-Server eingestellt werden können. Außerdem kann hier der Skalierungsfaktor und das Intervall für den Übertragungsabstand eingestellt werden.

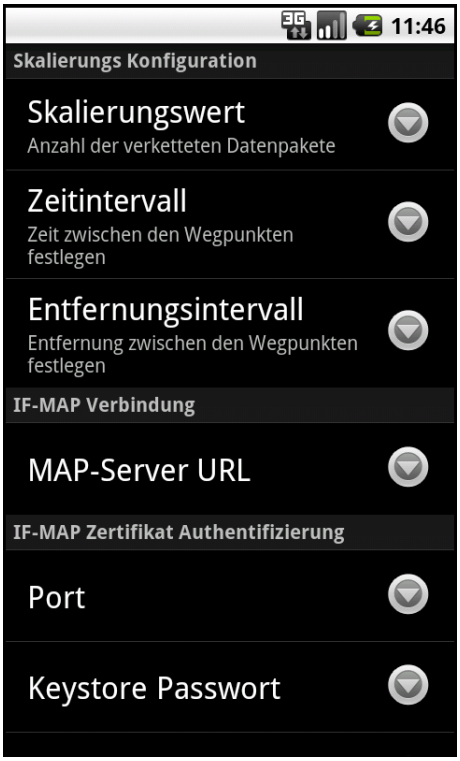

Abb. 9.2: Einstellungsmenü

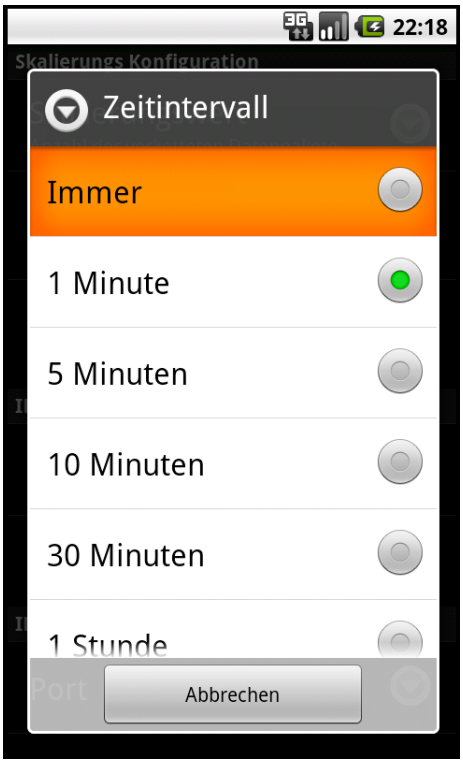

Abb. 9.3: Auswahl Zeitintervall

Realisiert wird das ganze über das von Android zur Verfügung gestellte Framework “PreferenceActivity”. Dies beinhaltet Methoden zur Generierung der Oberfläche für Einstellungen und sorgt automatisch dafür, dass vorgenommene Änderungen abgespeichert werden. Es ist möglich zur Laufzeit das Intervall, in dem die Daten übertragen werden, zu verändern. Standardmäßig wird jedes 100te Paket signiert. Dieser Wert ist frei änderbar. Für den zeitlichen Abstand zwischen den einzelnen Paketen hab ich mich für eine Auswahl vorgegebener Werte entschieden, um es dem Benutzer einfacher zu machen. Die Einstellungen “immer” sollte nur gewählt werden, wenn das Gerät an ein Netzteil angeschlossen ist, da das dauerhafte Abfragen von GPS Daten zu einem hohen Verbrauch des Accus führt. Für die Erfassung der Wegdaten gibt es neben dem Zeitintervall noch die Möglichkeit, eine Entfernung vorzugeben, nach welcher ein neues Datenpaket generiert wird. Der Wert sollte je nach Länge der Wegführung gewählt werden. Bei einer kurzen Strecke kann der Abstand recht eng gewählt werden, um eine möglichst gute Abdeckung der Strecke zu gewährleisten. Auf langen Wegführungen kann ein zu klein gewähltes Intervall zu einer verwirrend großen Anzahl von Wegpunkten führen. Die Generierung wird ausgelöst, je nach dem welche Option zuerst erfüllt wird, Weg oder Zeit. Neben dem Eintrag auf der “Startseite” können die Optionen auch jederzeit über den Menü-Button am Gerät aufgerufen werden. Über den Button “Start” gelangt man in die eigentliche Anwendung. Diese ist in den nächsten Screenshots zu sehen.

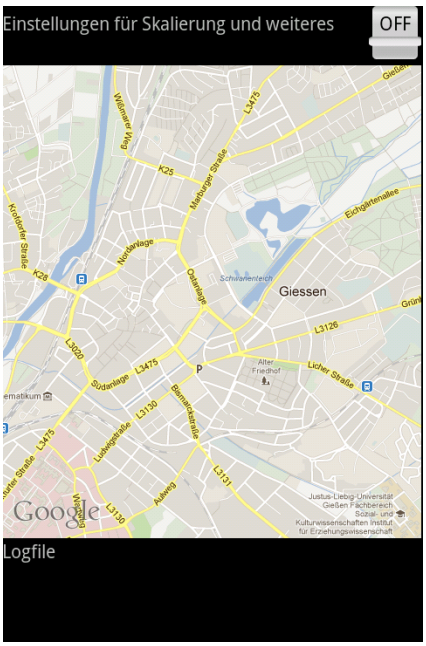

Abb. 9.4: mit Google Maps

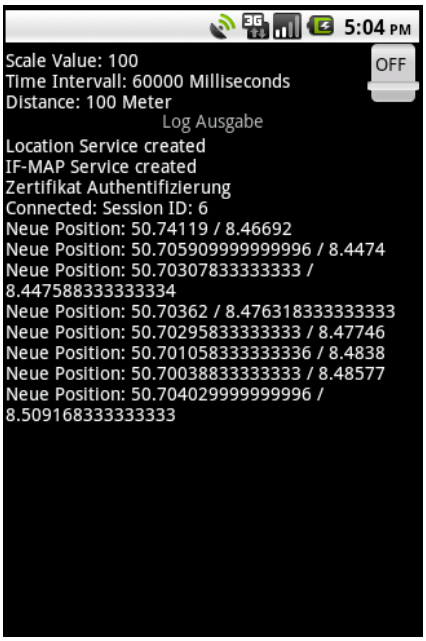

Abb. 9.5: ohne Google Maps

Wie in den Abbildungen zu sehen ist, werden im oberen Bereich die aktuellen Einstellungen für Übertragungsabstand und Skalierung angezeigt. Außerdem ist rechts ein Schalter zum Beenden oder Starten einer Beweisführung. In der Mitte wird die Karte von Google Maps angezeigt. Auf dieser ist immer die zuletzt gesendete Position zu sehen. Die Anzeige der Karte ist nicht auf jedem Gerät möglich. Sollte es nicht möglich sein, wird stattdessen der Bereich komplett mit dem Logfile gefüllt. Der Dritte untere Bereich, stellt ein Log der ausgeführten Aktionen dar. In diesem Bereich werden eventuell auftretende Fehler angezeigt. Ansonsten wird hier festgehalten, wann und welche Daten übertragen werden.

Um die Google Karte in der Mitte der Activity anzeigen zu können, muss die von mir compilierte Version des Android Betriebssystems erneut überarbeitet werden. In den Sourcen welche standardmäßig zur Verfügung stehen, sind die nötigen Dateien für den Zugriff auf Google Maps nicht vorhanden. Im Internet existieren verschieden Anleitungen, wie die Integration vorgenommen werden kann. Eine Möglichkeit ist es die benötigten Daten direkt in das Image mit hinein zu compilieren. Da die Daten für Google Apps proprietär sind, müssen sie aus einem Handy ausgelesen werden. Als Vorlage wurde ein Samsung Galaxy Mini verwendet, da auf diesem Android 2.2.1 installiert ist. Neben der eigentlichen Anwendung werden zusätzlich noch ein Framework und ein XML Schema zur Rechtevergabe benötigt.

```
vendor/google/app/Maps.apk
vendor/google/etc/permissions/com.google.android.maps.xml
vendor/google/framework/com.google.android.maps.jar
```

Um dem Compiler mitzuteilen, dass diese Daten von Google mit in das Image eingebaut werden sollen, müssen zusätzlich noch einige Buildscripts angepasst werden. So muss die im Verzeichnis /android/build/target/product liegende Datei AndroidProducts.MK um eine Zeile erweitert werden.

```
$(LOCAL_DIR)/generic_google.mk
```

Die Datei generic_google.mk ist eine Kopie der standardmäßig vorhandenen Datei generic.mk mit Erweiterungen für die Google Applikationen.

```
PRODUCT_PROPERTY_OVERRIDES := \
    ro.com.google.locationfeatures=1
PRODUCT_PACKAGES := \
    Maps \
    com.google.android.maps
PRODUCT_COPY_FILES := \
    vendor/google/app/Maps.apk:system/app/Maps.apk \
    vendor/google/etc/permissions/com.google.android.maps.
       xml:system/etc/permissions/com.google.android.maps.
       xml \
    vendor/google/framework/com.google.android.maps.jar:
       system/framework/com.google.android.maps.jar \
```

Während dem Compilieren erkannte ich, dass diese Methode nicht für ein generisches Image anwendbar ist. Bei der Suche nach weiteren Möglichkeiten stellte ich fest, dass immer ein reales Gerät benötigt wird, da es keine Option gibt einen Emulator im Recovery-Modus zu starten. Dieser wird allerdings benötigt um die Applikationen von Google nachträglich zu installieren. Somit habe ich die App dahingehend angepasst, dass eine Verwendung mit und ohne Google Maps möglich ist.

Um eine Activity mit einer Karte zu realisieren, muss die Oberfläche aus verschiedenen Layouts zusammengesetzt werden. Die Verbindung zu dem IF-MAP Server wird über einen im Hintergrund laufenden Service aufgebaut und besteht so lange, wie die Applikation läuft. Ein zweiter Service ist für das Erfassen der GPS Koordinaten zuständig. Wenn man die Erfassung direkt in der Activity laufen lässt, führt dies dazu, dass die Applikation nach dem Empfang von Koordinaten blockiert wird, bis neue Daten eintreffen. In diesem Zeitraum ist es nicht möglich andere Aktivitäten auszuführen. Die Services werden mit dem Aufruf der Hauptactivity gestartet. Gleichzeitig wird die Generierung des AIK ausgelöst und ein signierter Timestamp von der TA angefordert. Es kann bis zu zwei Minuten dauern, bis dieser von der TA geliefert wird. Wenn eine App für einen zu langen Zeitraum nicht reagiert, wird vom System ein Timeout ausgelöst. Um dies zu verhindern wird in einem Dialog eine Ladeanimation angezeigt.

Für die Kommunikation zwischen den Komponenten verwende ich Broadcasts. Diese können an beliebiger Stelle innerhalb einer Anwendung versendet und in allen Activitys oder Services empfangen werden. Um zu bestimmen für wen die Nachricht gedacht ist, wird ein sogenannter Intent festgelegt. Jeder Reciever der auf eine Nachricht wartet, empfängt alle Broadcasts die versendet werden. Über den Intent wird dann gefiltert, welche für den jeweiligen Reciever bestimmt sind. Eine Übersicht der verwendeten Klassen befindet sich im Anhang B.2 auf Seite 51.

Kapitel 10

Datenverarbeitung

Um die gewonnenen GPS Koordinaten und die damit verbundenen administrativen Daten speichern zu können, wurde von uns ein Location Package implementiert. Dieses ist wie die Datenstruktur selbst in die Bereiche GPS, Meta, Attestation und Signature und TA unterteilt.

Nach der Übergabe einer neuen Position vom Location Service an die Hauptactivity, wird ein Location Paket mit den Daten gefüllt. In das Paket werden von mir neben den Koordinaten Publisher ID, Sequenznummer, Start/Stop Flag, Sensorwerte, und Scale Value gesetzt. Wenn diese Werte gespeichert sind, wird die von Herrn Schlüter implementierte Klasse Scale aufgerufen. Diese versorgt das Paket mit den Daten der TA. Weiterhin übernimmt sie die Verkettung und Signierung der Daten. Wenn das Paket vollständig ist, wird es aus der Activity an den IF-MAP Service übergeben.

Der Service überprüft, ob die Verbindung zum MAP-Server noch besteht und versendet die Daten. Sollte die Überprüfung der Verbindung fehlschlagen, wird das Paket solange im Cache zwischengespeichert bis eine neue Verbindung aufgebaut werden kann.

Während der Tests ist uns aufgefallen, dass teilweise GPS Koordinaten doppelt vom Location Service geliefert werden. Um diese nicht unnötig weiter zu verarbeiten, habe ich eine Funktion zur Überprüfung der GPS Daten implementiert. In dieser werden die neuen Daten mit den zuletzt signierten verglichen. Es wird überprüft, ob die in den Preferences eingestellten Werte für Mindestabstand und Zeitspanne eingehalten werden. Sollte dies nicht der Fall sein, wird das Datenpaket verworfen. Ansonsten werden die Koordinaten weiterverarbeitet und als zuletzt gesendete Position gespeichert.

Kapitel 11

Archivierung

Alle mit einem Evidence Generator erstellten Daten müssen dauerhaft und sicher über einen langen Zeitraum gespeichert werden. Für das Archiv wird keine spezielle Absicherung vorgesehen, da davon auszugehen ist, dass die für das Archiv verwendete Infrastruktur an einem als vertrauenswürdig anzusehendem Ort betrieben wird. Es werden alle gesendeten Daten in einer Datenbank gespeichert. An das Archiv ist ein Policy-Framework angegliedert, mit dem es möglich ist, ein Regelwerk für verschiedene Szenarien zu definieren. Die Kommunikation läuft, wie bei den anderen Komponenten auch, über den MAP-Server. Eine Vorverifizierung ist ein Fall der mit dem Regelwerk realisiert werden kann. Hierbei kann es sich zum Beispiel um eine Plausibilitätsüberprüfung oder eine SML-Überprüfung handeln.

Diese Überprüfung ist recht aufwendig, da sich die Reihenfolge der in der SML vorhandenen Einträge bei jedem Start des genutzten Geräts ändert. Es kann nicht festgelegt werden in welcher Reihenfolge die einzelnen Anwendungen auf einem Gerät gestartet werden. Bei der Überprüfung muss jede Zeile der SML aus dem zu verifizierenden Datensatz mit einer Referenz SML aus dem Archiv verglichen werden. Ein Ausschnitt aus einer Beispiel SML befindet sich im Anhang B.3 S.50

Die Archivierung ist neben dem Policy-Framework Themenbereich der Thesis von Herrn Dieter Kramer [Kra12].

Kapitel 12

Verifikation

Der ganze Aufwand Daten zu signieren lohnt sich nur, wenn die Sicherheitsfunktionen irgendwann auch wieder verifiziert werden. Die Überprüfung wird dabei in folgender Reihenfolge vorgenommen:

- Zunächst werden die Signaturen von TPM-Quote und TPM-Tickstamp überprüft.
- Danach die TA Uhrzeit.
  Dazu wird der Tickstamp gesichtet und die zugehörige Time Variable der vorliegenden Daten. Aus diesen Werten wird dann die echte Uhrzeit berechnet.
- PCR /SML
- Sollten diese Werte alle ohne negativen Befund sein, wird die Verkettung der einzelnen Datenpakete anhand der Freshnesswerte überprüft.
- Je nach verwendetem Verfahren wird die Verkettung der Rolling-Hashes oder der Hashliste überprüft.
- Wenn die Verkettung vollständig ist, wird als letzter Schritt die Überprüfung der Datenpakete vorgenommen. Hierbei werden die einzelnen Pakete auf Modifikationen hin untersucht. Dies geschieht durch eine Signatur-Gegenprüfung. Von jedem Paket wird eine neue Signatur erstellt und mit der vorhandenen verglichen.

Der Vergleich einer SML ist recht aufwendig. In ihr gibt es für jedes gestartete Programm einen Eintrag. Da die Reihenfolge in der die einzelnen Programme gestartet werden nicht beeinflusst werden kann, unterscheidet sich jede SML von der vorherigen. Um einen Vergleich machen zu können, muss jede Zeile der SML mit einer Referenz verglichen werden. Dies stellt je nach Größe der SML einen hohen Aufwand dar. Dieser ist so hoch, dass es sich nicht rechnet diesen Vergleich auf einem mobilen Gerät zu machen. Stattdessen wird dies bereits vorab auch dem MAP-Server durch das Regelwerk vorgenommen. Der Server auf dem diese Anwendung läuft ist wesentlich leistungsfähiger als ein mobiles Gerät und kann die Überprüfung unabhängig vom Benutzer durchführen. Die weiteren Schritte der Verifikation lassen sich problemlos auf dem mobilen Gerät realisieren.

Das Themengebiet Verifikation ist Bestandteil der Thesis von Herrn Mark Schlüter [Sch12].

Kapitel 13

Visualisierung

Bei den gesammelten Daten von verschiedene Evidence Generatoren sollte es auch möglich sein, sich einen Überblick über die vorhandenen Daten zu verschaffen. Die Darstellung der gesamten Beweisdaten, welche sich im Archiv befinden, soll in einer Art Beweiswolke geschehen. Die Wolke besteht aus Graphen, die die einzelnen Beweispunkte über Verbindungen in Relation setzen. Die folgende Grafik stellt ein Beispiel für einen Beweisgraph in der Wolke dar. Anhand der Einfärbung der einzelnen Punkte kann man den Stand der Verifikation erkennen. Eine weitere Möglichkeit der Visualisierung ist der Location Inspector. Neben der Möglichkeit die im Archiv gespeicherten Daten zu verifizieren, können diese auch auf einer Karte angezeigt werden. Nachdem man eine Beweisführung ausgewählt hat, wird diese auf der Karte dargestellt. Je nach Stand der Verifikation werden die einzelnen Wegpunkte dann eingefärbt. Grün, falls die Verifikation ohne Beanstandungen durchgeführt wurde, Rot bei einem Fehler.

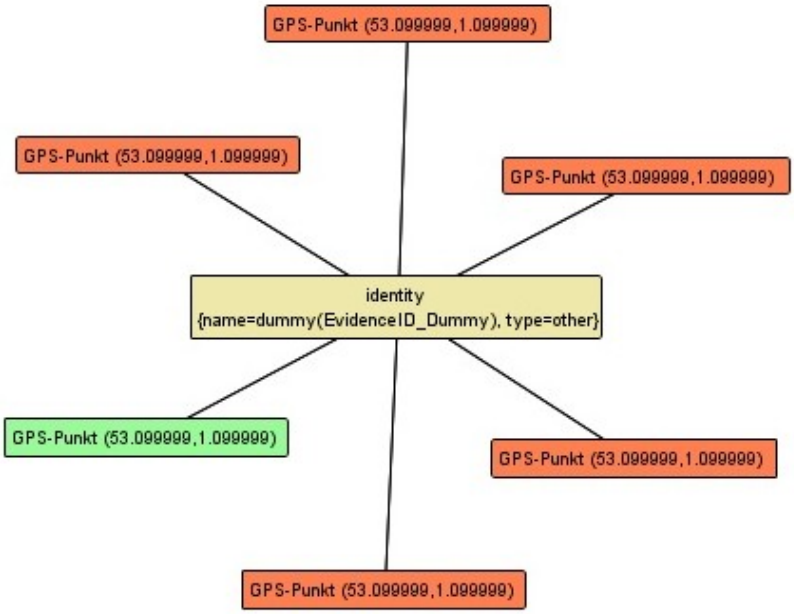

Abb. 13.1: Beweiswolke (Grafik von Herrn Kramer zur Verfügung gestellt)

Teil IV

Teamwork

Kapitel 14

Gruppenarbeit

In diesem Kapitel möchte ich einige persönliche Worte zum Thema Gruppenarbeit verfassen. Es wurden mehrere Arbeiten anhand des Projekts TruLoc geschrieben. Meine Bachelor Arbeit, sowie zwei weitere Master Arbeiten. Das Projekt wurde von Anfang an zusammen geplant. Alle Beteiligten haben verschiedene Ideen eingebracht, welche in der Gruppe diskutiert wurden. In den Diskussionen wurde immer ein Konsens gefunden und gemeinsam entschieden, welche Ideen umgesetzt werden können. Bei einer solchen Entscheidung muss immer abgewägt werden, ob das Vorhaben in einem angemessenen Zeitraum umgesetzt werden kann. Aber auch nach einer solchen Entscheidung kann es vorkommen, dass im Verlauf der Arbeiten festgestellt wird, dass ein Arbeitsabschnitt den geschätzten Zeitrahmen sprengt. Dies ist bei uns eingetreten. Die Anpassung des Betriebssystems auf eine aktuelle Version war zu zeitintensiv. In einem solchen Fall muss dann entschieden werden, ob der Mehraufwand vertretbar ist, oder ob man die Arbeiten an der Stelle abbrechen sollte.

Ein Projekt dieser Größenordnung kann in einem annehmbaren Zeitraum nicht von einer Person alleine bewältigt werden. Aus diesem Grund ist es notwendig, dass alle beteiligten Personen zusammenarbeiten. Ebenfalls ist es unwahrscheinlich, dass jemand sich in allen Bereichen, die für die Umsetzung eins Projekts notwendig sind, perfekt auskennt. Während der Entwicklung sollten Fortschritte immer in der Gruppe besprochen und getestet werden. Eine sehr gute Möglichkeit in der Gruppe mit den Arbeiten weiter zukommen, ist Kommunikation und konstruktive Kritik. Eine gute Zusammenarbeit ist aber nicht nur von internen Faktoren abhängig, sondern auch von externen. So kann es vorgekommen, dass man längere Zeit auf Informationen oder Daten warten muss, die von außerhalb der Gruppe angefordert werden.

Einen weiteren Faktor stellt das Internet dar. Es kann passieren, dass ein Webserver, zum Beispiel durch einen technischen Defekt, über einen längeren Zeitraum nicht erreichbar ist. Solche Vorkommnisse sollten immer bei der Zeitplanung berücksichtigt werden und der Plan nicht zu eng gesteckt werden.

Bei der Beteiligung mehrerer Personen verläuft die Zusammenarbeit nicht immer reibungslos. So ist es möglich, dass man von Personen über einen gewissen Zeitraum keine aktuellen Informationen erhält. Dies kann dazu führen, dass sich die verschiedenen Arbeitsabschnitte zu weit von einander entfernen und somit zusätzliche Zeit für Anpassungen aufgewendet werden muss.

Ein weiteres Problem war die Informationspolitik von Google. Zwar gibt es eine ausführliche Beschreibung der API, aber es war schwierig Informationen zu finden, wie man die Google Apps für Android in ein selbst compiliertes System integriert.

Teil V

Fazit

Kapitel 15

Fazit und Ausblick

Im Ausblick werden Änderungen oder Verbesserungen des Projektes behandelt, welche in der aktuelle Implementierung aus zeitlichen Gründen oder wegen erheblichem Aufwand nicht berücksichtigt werden konnten.

Während unserer Arbeiten am Projekt wurden wir auf eine Meldung von Heise Security aufmerksam [Hei10], Diese befasst sich damit, dass es einem Hacker gelungen ist, den Krypotschlüssel aus einem TPM-Chip auszulesen. Auf Grund der Tatsache, dass der Chip selbst dabei zerstört wird und das Verfahren sehr aufwendig und teuer ist, bin ich der Meinung, dass dies kein all zu großes Sicherheitsrisiko darstellt.

15.1 Erneuerung des TPM-Quote

Es ist wünschenswert für jede Beweiskette einen aktuellen TPM-Quote zu besitzen. Dieser Quote enthält unter anderem das Mesurement Log. In diesem Log-File sind die vom TPM gemessenen Werte der einzelnen Softwarekomponenten gespeichert. Um dies zu erreichen, muss das Gerät, auf welchem der Evidence Generator läuft vor jeder Beweiskette neu gestartet werden.

15.2 Initialgeheimnis auf Gerät übertragen

Um auszuschließen, dass das Initialgeheimnis kompromittiert wird, wäre es möglich dieses auf einer vertrauenswürden Art in einer gesicherten Umgebung zu generieren. Dieses generierte Geheimnis könnte mit dem öffentlichen Teil des Endorsement Key verschlüsselt werden und dann per SD-Karte auf das Handy übertragen werden. Auf dem Gerät kann das Geheimnis wieder entschlüsselt werden. Dies wäre eine sichere Art der Übertragung, da der private Teil des EK nur dem TPM-Chip bekannt ist. Eine Kompromittierung wäre somit nicht möglich.

15.3 Absicherung des Systemstart

Eine Möglichkeit das System abzusichern ist es, den Start per Sealing an den vom TPM gemessenen Zustand des Systems zu koppeln. Dazu werden die Messwerte, welche durch den TPM beim Systemstart erstellt werden, mit einem Satz von Werten verglichen, welche im gesicherten Speicherbereichs des TPM-Chip abgelegt sind.

15.4 Zertifikate

Der verwendete Keystore, der das von uns benutzte Zertifikat enthält, ist in der aktuellen Version des Programms nicht austauschbar. Die Datei wird mit der Applikation auf dem Gerät installiert. Im Moment ist keine Möglichkeit vorgesehen, eine solche Datei zu importieren, oder nachträglich auszutauschen.

15.5 TrouSerS, JTSS Wrapper / JTSS

In den von uns verwendeten Ressourcen wird im Betriebssystem der TrouSerS Software Stack als Verbindung zum TPM-Chip verwendet. Dieser Software Stack ist in der Programmiersprache C implementiert. Um aus einer Android Applikation darauf zugreifen zu können, benötigt man zusätzlich den JTSS Wrapper. Wie der Name schon vermuten lässt, stellt ein Wrapper eine Umhüllung für die Programmiersprache Java dar. Mit dieser Hülle werden die Strukturen von TrouSerS umhüllt, sodass ein Zugriff ermöglicht wird. Die für unser Projekt verwendete Version wurde an der Uni Graz[1] in Jahr 2009 entwickelt.

Mittlerweile wurde eine neuere Version von JTSS entwickelt. Sie stammt aus dem Jahr 2011. Zu den Neuerungen zählt unter anderem, dass der TrouSerS Stack nicht mehr benötigt wird. Dieser Stack ist nun auch in Java implementiert. Das JTSS Projekt ersetzt somit den Wrapper und TrouSerS. Diese neuere Version wird von uns nicht verwendet, da der Umbau zu aufwendig gewesen wäre.

[1] http://trustedjava.sourceforge.net/

Anhang

Anhang A

Kurzfassung

In diesem Projekt geht es darum die Hardware und Software einer mobilen Plattform so zu gestalten, dass sie als Basis für vertrauenswürdige Daten genutzt werden kann.

Aufgabenstellung war es, Positionsdaten zu generieren, sie zu archivieren und zu visualisieren. Gleichzeitig soll es zu einem späteren Zeitpunkt möglich sein, die Authentizität der Daten zu verifizieren.

Seit es Hard- und Software gibt, gibt es auch Angriffe auf selbige. Dabei wird versucht sich Daten unrechtmäßig anzueignen, oder sie zu manipulieren. Bei einem mobilen Gerät kann nicht immer sichergestellt werden, dass es sich in einer kontrollierbaren gesicherten Umgebung befindet. In einer Alltagssituation, in der das Gerät kurzzeitig nicht beaufsichtigt wird, kann eine nicht autorisierte Person eine Manipulation vornehmen oder das Gerät entwenden. Die Tatsache, dass ein Gerät entwendet wird, oder Daten verändert werden, kann auch dieses Projekt nicht verhindern, aber eine vorgenommene Manipulation kann aufgedeckt werden. Um dies zu bewerkstelligen muss die verwendete Hard- und Software angepasst werden.

Bevor das Projekt umgesetzt wurde, gab es zunächst eine Planungsphase. In dieser wurden verschiedene Szenarien erörtert für die unser Projekt von Interesse sein könnte. Diese wurden auch im Hinblick auf ihre Umsetzbarkeit hinsichtlich unser Möglichkeiten untersucht. In Betracht kamen eine Reihe von Szenarien aus den Bereichen Medizin, Transport und Produktion. Am Ende entschieden wir uns für die Umsetzung einer Kontrolle von komplexen Lieferketten-Systemen. Dabei wird die zurückgelegte Route eines Transports protokolliert. Zusätzlich kann bei einem Kühltransport die Temperatur der Ware kontrolliert werden.

Um unser Vorhaben umzusetzen, werden verschiedene Ansätze der Trusted Computing Group kurz TCG verwendet. Die Hardware wird durch ein sogenanntes Trusted Platform Module, im weiteren Verlauf TPM-Chip genannt, abgesichert. Dieses Modul verfügt über verschiedene Funktionen zur Signierung und Verschlüsselung. Zur Realisierung wird eine komplexe Infrastruktur benötigt, mit der die gewonnenen Daten übertragen, archiviert und verifiziert werden können. So ist zum Beispiel eine unabhängige Autorität nötig, mit der ein vertrauenswürdiger Zeitstempel erstellt werden kann, die so genannte Timing Authority. Damit kann gewährleistet werden, dass diese Daten leichter vor Gericht Bestand haben. Gerichte haben es im Moment nicht einfach digitale Daten als Beweismittel zuzulassen, da eine nachträgliche Veränderung oft nicht ausgeschlossen werden kann. Eine Manipulation hinterlässt meist keine Spuren.

Als Grundlage stand uns die Infrastruktur eines vorhergehenden Projekts zur Verfügung. Hierbei handelt es sich um das Entwicklungsprojekt TruCam. In diesem Projekt wurde ein erster Prototyp der Infrastruktur entwickelt. Einige der Softwarekomponenten waren nicht voll funktionsfähig und wurden im Laufe unserer Arbeiten

verbessert.

Um mit dem TPM-Chip kommunizieren zu können, muss das Betriebssystem angepasst und um einige Funktionen erweitert werden. Zur Zeit der Entwicklung stand uns keine Hardware mit eingebautem TPM-Chip zur Verfügung. Aus diesem Grund wurde während der Entwicklung ein Emulator verwendet. Zusätzlich ist es mit einem Emulator einfacher Fehlermeldungen auszulesen. Ein Hardware TPM-Chip hat im Gegensatz zu einem auf Software basierenden keine Möglichkeit Fehlermeldungen anzuzeigen. Eine genaue Beschreibung der Arbeiten am Betriebssystem finden Sie im Kapitel 6 auf Seite 23.

Die Arbeiten am Betriebssystem stellen nur einen Teil meines Aufgabenreichs dar. Der größere Teil ist die Entwicklung einer Anwendung für die Android Plattform. Mit dieser Applikation werden die gewünschten Wegdaten erzeugt, signiert und an den MAP-Server übertragen.

Zunächst wurde in der Planungsphase vom gesamten Team eine Datenstruktur (Kapitel 4.1 Seite 12) entwickelt, welche dann auch von allen Beteiligten genutzt wurde. In dieser Datenstruktur ist neben den GPS Koordinaten auch Platz vorgesehen, für die Speicherung von Sensordaten. Neben diesen Nutzdaten beinhaltet die Datenstruktur aber auch einen nicht unerheblichen Teil an administrativen Daten. Um nicht jedes mal alle Daten senden zu müssen, werden die einzelnen Datenpakete miteinander verknüpft. Durch die Verknüpfung ist es nicht nötig in jedem Paket alle Daten zur Signierung mitzusenden. Dadurch wird ein Großteil der ansonsten benötigten Bandbreite eingespart.

Neben dem technischen Aspekt wurde auch eine ergonomisch nutzbare Applikation von mir entwickelt. Dabei habe ich darauf geachtet, dass der Nutzer alle wichtigen Daten leicht erkennen kann, aber gleichzeitig nicht mit einer zu großen Anzahl an Daten überflutet wird. Zunächst wird der Nutzer durch eine schlicht gehaltene Startseite empfangen, auf der es die Möglichkeit gibt in ein Optionsmenü zu wechseln oder eine Streckenführung zu starten. Nach dem Start einer Streckenführung wird dem Benutzer eine neue Oberfläche angezeigt. Auf dieser werden im oberen Bereich die verwendeten Einstellungen angezeigt. Den größten Teil beansprucht die Einblendung der aktuellen Position auf einer Karte von Google Maps. Im unteren Bereich wird ein Logfile dargestellt. In diesem wird der Benutzer über durchgeführte Arbeitsschritte informiert. Gleichzeitig werden hier etwaige Fehlermeldungen aufgezeigt. Abbildungen dieser Oberfläche befinden sich neben der Beschreibung in Kapitel 9 auf Seite 31.

Insgesamt zeigt das Projekt eine Möglichkeit digitale Daten auf einer als vertrauenswürdig geltenden Plattform zu generieren. Für die Absicherung der Plattform kommen verschiedene Ansätze und Maßnahmen der Trusted Computing Group zum Einsatz.

Anhang B

Tabellen und Grafiken

B.1 Datenstruktur

	Publisher ID	String	discovered-ID aus Spec.
	Time	dateTime	discoveredTime aus Spec.
Metadaten			
Meta: {			
	GPS_Latitude	Double	GPS Breitengrad
	GPS_Longitude	Double	GPS Längengrad
	GPS_Accuracy	Double	GPS Genauigkeit
	GPS_Altitude	Double	GPS Höhenangabe
	GPS_Time	Double	GPS Zeitstempel
	Sensor_List	Hashliste	Liste für Sensorwerte
	Evidence_ID	String	Pub-ID + Datum/Uhrzeit
	Sequence_No	Long	Paket Counter/Nummer
	Start_Stop	Integer	Start=0, Stop=1, Rest=NULL
	Scale_Value	Integer	Wert der Skalierung
	Hash (Liste)	Liste	Verkettung der Zwischenpakete
	Hash (Wert)	String	Je nach Liste oder Wert
}			
Atestierung			
Attestation: {			
	SML	String	“pcr hash path pcr hash path ...”
	Secret	String	160 Bit Wert aus TPM.getRandom()
}			
TA: {			
	Tickstamp	String	vom TPM erstellter Tickstamp der verifiziert werden soll
	Signed_Tickstamp	String	von TA signierter Tickstamp
	Current_Ticks	BigInteger	
	Time	Long	Echtzeit
	Tick_Nonce	BASE64 String	Nonce gegen Replay-Att.
}			
Signatur			
Signature: {			
	Quote_Data	BASE64 String	TcBlobData - Q-Data
	Quote_ValiData	BASE64 String	TcBlobData - Q-ValidationData
	Pub_Key	BASE64 String	TcTpmPubkey
	Q_Tickstamp_Vali	BASE64 String	Signierter Current_Tickstamp, Tickstamp-ValidationData
	Q_Tickstamp_Data	BASE64 String	Tickstamp-Data
	Q_Current_Ticks	BigInteger	aktueller Zählerwert des TPM
	Q_Tick_Nonce	String	Nonce gegen Replay-Att. enthält unser Quote
	TickRate	Integer	Zähler, 1 entspricht 1ms
}			

Tab. B.1: Datenstruktur

B.2 Messung von AIKs

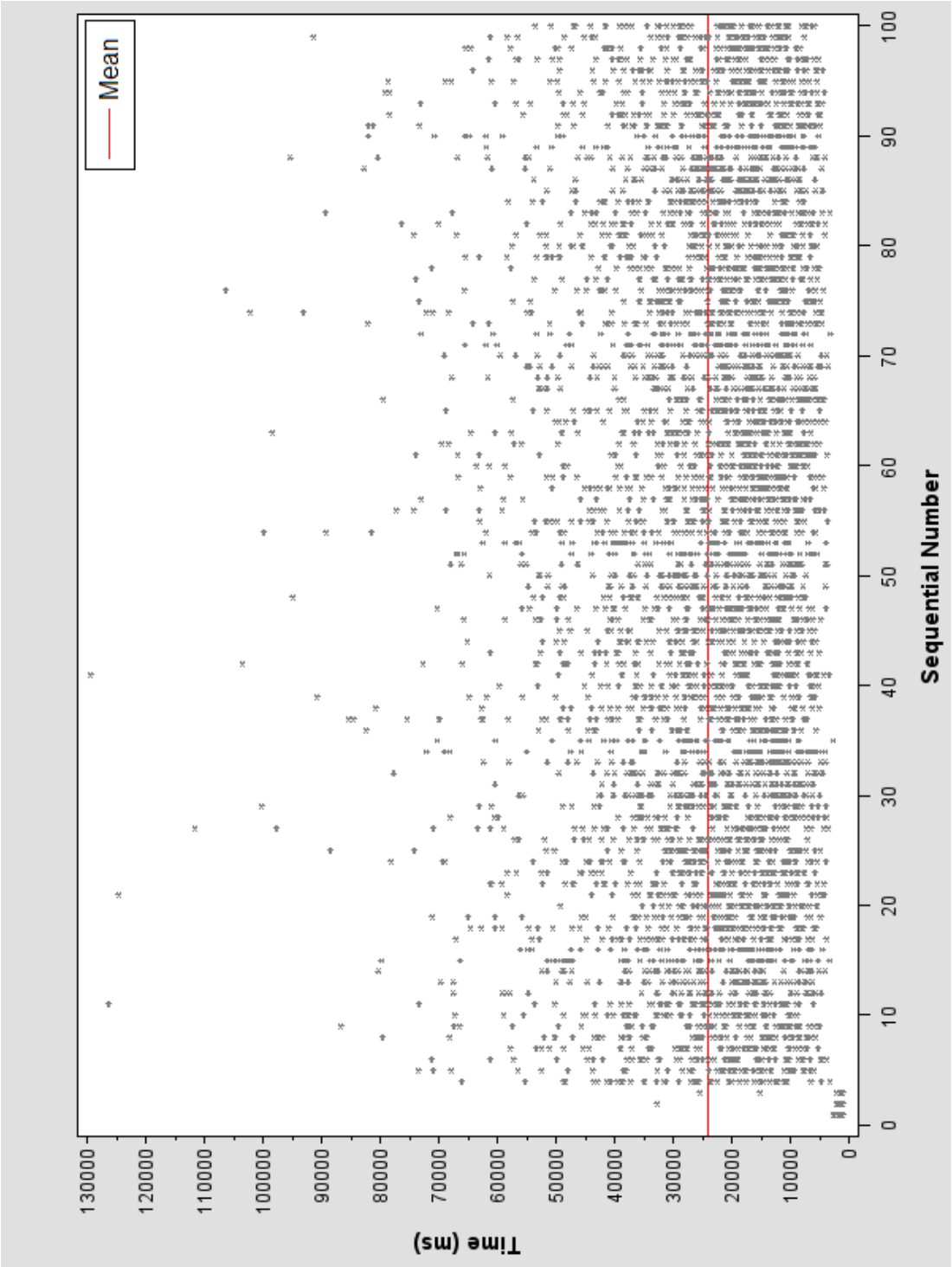

Abb. B.1: Messung von AIKs

B.3 Beispiel SML

Eine komplette SML ist noch weit aus umfangreicher als der hier abgebildete Ausschnitt. Er genügt allerdings um sich den Aufbau einer solchen Datei vorstellen zu können. Für jede auf dem Gerät laufende Software wird in der SML eine Zeile mit dem Messwert hinterlegt.

```
10 d435f0cc4244a7ae338633db6fcc0d62eff26916 /init
10 45ca463de9369cc9a80b0efe8dbb56c4ee610837 /system/bin/
   toolbox
10 d93ccdb4215dfffd788b181437825e085fede516 /system/bin/
   linker
10 71925792a0d7e6d6996182aa5966da1aba3c8970 /system/lib/
   liblog.so
10 c28916493bffe9333883026f28a1f9701aae1359 /system/bin/
   servicemanager
10 b5e6798d7c687c70a094d5662944d998604cd79e /system/bin/vold
10 d3991a43e0e70327ddd39dd3021d8a1eb70baf22 /system/bin/
   debuggerd
10 d0c3eabf3f928c60c3918f8c0b943d81f9384a50 /system/bin/rild
10 9f86aa15100d0f24a51c4af4d052cbc43cbc0f43 /system/bin/
   app_process
10 dc4ebfb93f6e24e35b2c603b3aa01be027ce3fd5 /system/bin/
   mediaserver
10 15f821e21fb141d6a4f56e06c2327241268612ff /system/bin/dbus
   -daemon
10 dd4379e716a1a960b662b6644ac8cf5735e0dd47 /system/bin/
   installd
10 0928414af9ade3bde73bcdaec7bb30a35d5eb6cf /system/bin/
   keystore
10 3360aa9904264a9f4530b1cda80e55311b0e958d /system/bin/sh
10 f09b6e0ec2c79ac0a1ac2f8f2af89a17e8772f45 /system/lib/libc
   .so
10 e2f43be1e7c983e451f2d6699cd3b2c47528f0f9 /system/bin/
   qemud
10 e6cb1600f1d3931dc87d16a74780ba650bf1e4d5 /sbin/adbd
10 030588456392caf07b9e8ad7c59596c9dd35199b /system/bin/
   logcat
10 74ea20e2a5ffab928e3b95a8193fa4defa30b838 /system/lib/
   libstdc++.so
10 8e94002a5c483778561c81fb75b8805d24a8af74 /system/lib/
   libexpat.so
10 10d298d28e6861df8a651be6760935f6b1a3a794 /system/lib/
   libaudioflinger.so
10 a4bc08ce0db62728b37e84aaac1f485c7c33ceff /system/bin/tcsd
10 678ed59919cade831e6341a5b0c222bfc9714cbc /system/lib/libm
   .so
10 4eae4915420c657428c39ca3eedd1fb24b3dccff /system/lib/
   libcutils.so
10 842287c2972976c77d353b51427363d8d0853532 /system/bin/tpmd
10 4df20c5ccf9a08f2c64df00b6f5cdef20d179b66 /system/lib/
   libdbus.so
10 9f92739b0dced10157d30b3a24ab2a466a967fb9 /system/lib/
   libutils.so
10 95960567ac9efaa42c80e6281fc609c2b988288f /system/lib/
   libril.so
10 ddab15b925aff2fb8a3289296a9ecfa05ade35aa /system/lib/libz
   .so
10 c7b9802ede919495113d4eb805e5b41621861daa /system/lib/
   libbinder.so
...
```

B.4 Klassendiagramme

B.4.1 Evidence Generator

Dies ist eine Übersicht der neu implementierten Klassen und Funktionen für den Evidence Generator. Es wurden noch weitere Quellen für die vollständige Implementierung benötigt. Allerdings wäre es zu umfangreich alle Klassen an dieser Stelle zu präsentieren.

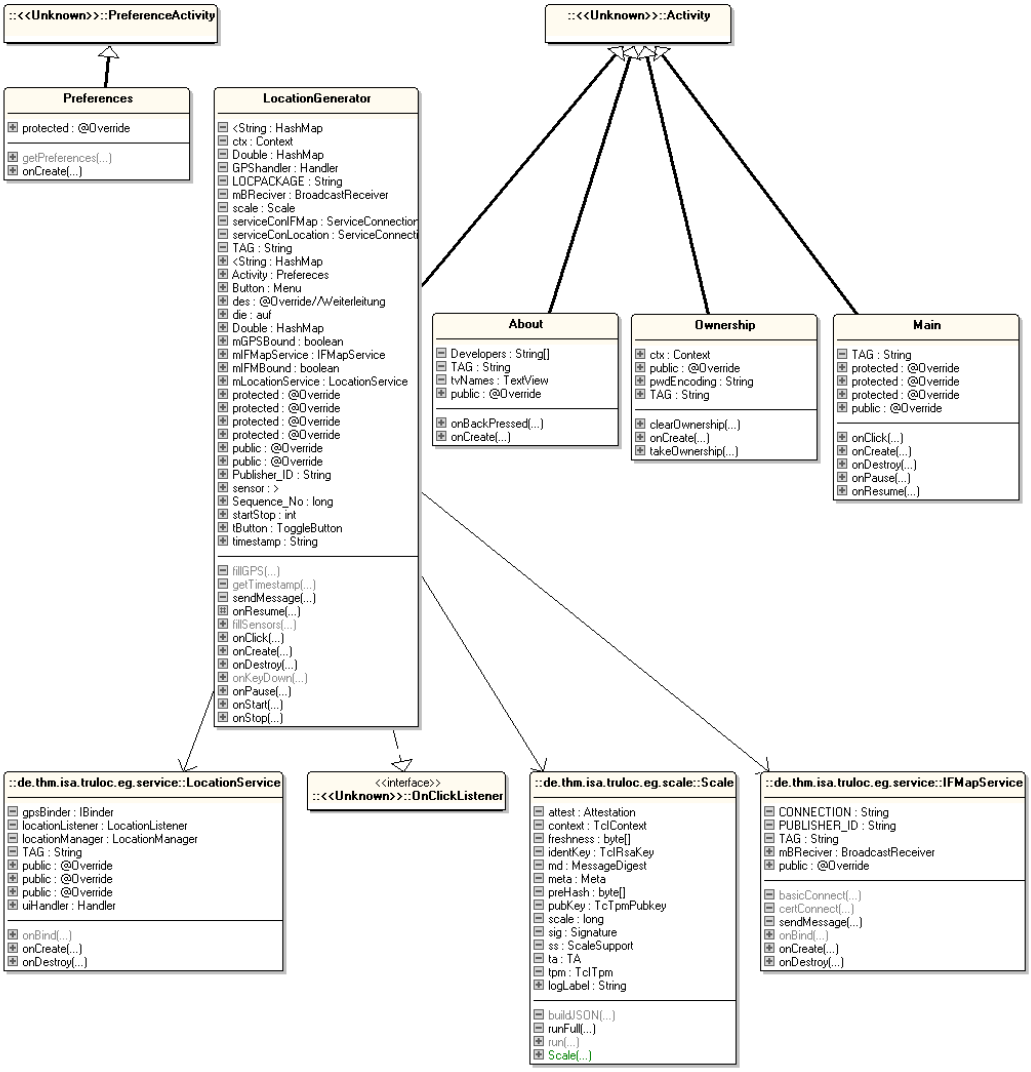

Abb. B.2: Klassendiagramm Evidence Generator

B.4.2 Evidence Package

Dies ist die Implementierung der Datenstruktur. Man kann gut erkennen aus welchen Teilen sich das Location Package zusammensetzt. Das Location Paket selbst beinhaltet die Klassen Meta, Attestation, Signature und TA. Im der Meta Klasse wird zusätzlich noch ein Objekt aus der Klasse GPS verwendet.

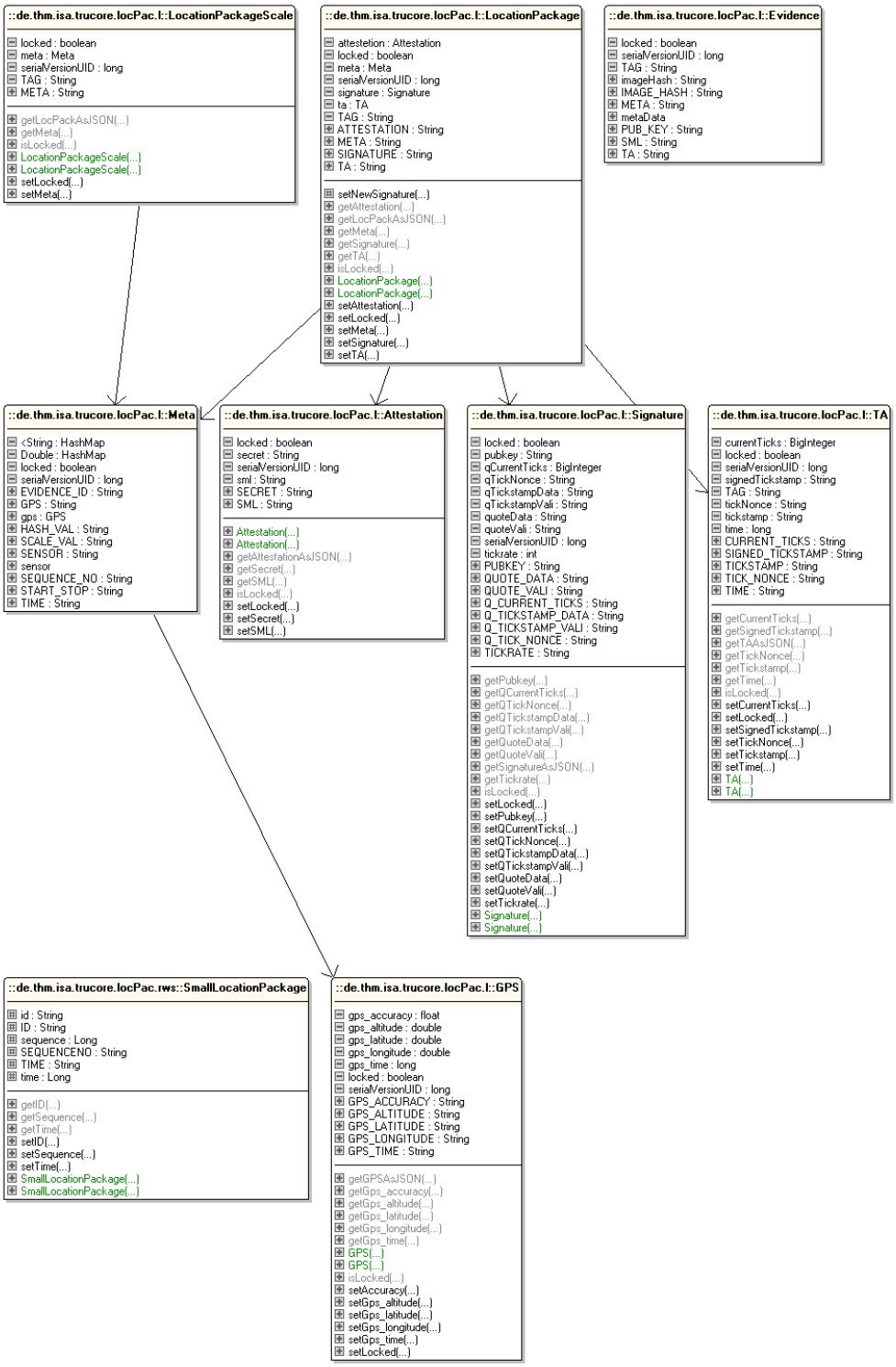

Abb. B.3: Klassendiagramm Evidence Package

Anhang C

Demonstrationsumgebung

Dieses Kapitel ist eine Installationsbeschreibung. Sie dient dem Leser dazu die Software zu testen, ohne sich vorher mit der Umgebung in der sie gestartet werden soll vertraut machen zu müssen.

C.1 Software

Für das Projekt werden verschiedene Software Komponenten benötigt. Die Komponenten werden auf der CD, welche dieser Arbeit beiliegt, mitgeliefert. Alle Programme sind für Windows und Linux in 32 und 64 Bit Versionen vorhanden.
Die benötigten Komponenten sind:

- Sun Java5 JDK
- Adroid SDK, inklusive Emulator
- Entwicklungsumgebung Eclipse
- TPM-Emulator
- MAP-Server

C.2 Installation

Ich werde die Installation anhand von Ubuntu Linux erläutern, da ich selbst auf diesem System arbeite. Die Installation auf anderen Systemen funktioniert vergleichbar. Die Einstellungen der einzelnen Programme werden nach dem gleichen Schema vorgenommen.

Das Projekt wurde unter Java Version 5 entwickelt. Daher muss zunächst das Sun Java 5 JDK installiert werden. Mit einer anderen Java Version können nicht vorhersehbare Fehler auftreten.

Die Installation kann über die auf der CD mitgelieferten Archiv oder Binary Dateien geschehen. Unter Linux ist es auch möglich das JDK über einen Paketmanager Ihrer Wahl zu installieren. Dazu ist es allerdings nötig, dem System die dazu notwendige Paketquelle bekannt zu machen. Die Konsolenbefehle dafür lauten:

```
sudo add-apt-repository ppa:ferramroberto/java
sudo apt-get update
```

Danach kann dann mit dem weiteren Befehl das JDK installiert werden.

```
sudo apt-get install sun-java5-jdk
```

Standardmäßig wird bei Ubuntu das Open-JDK vorinstalliert. Da einige Funktionen im Open JDK anders realisiert sind, sollte dies für die Demonstrationsumgebung deinstalliert und durch das Sun Java5 JDK ersetzt werden.

Das Android SDK wird von Hand in ein beliebiges Verzeichnis entpackt und kann direkt verwendet werden.

Als Entwicklungsumgebung kommt Eclipse zum Einsatz. Im Paketmanager ist zwar eine Version vorhanden, da sie aber relativ alt ist, bietet es sich an, eine aktuellere von Hand aufzuspielen. Wie schon zuvor beim Android SDK wird nun auch wieder ein Archiv in ein beliebiges Verzeichnis entpackt und kann sofort verwendet werden.

C.3 Einstellungen

Nach dem Installieren der einzelnen Komponenten müssen noch verschiedene Einstellungen vorgenommen werden.

C.3.1 Eclipse

Der Entwicklungsumgebung muss noch der Ort bekannt gemacht werden, an dem der SDK installiert ist. Dazu muss im Menu "Window"der Unterpunkt "Preferences"geöffnet werden. Dort im Abschnitt "Android" muss die Position des SDK eingegeben werden.

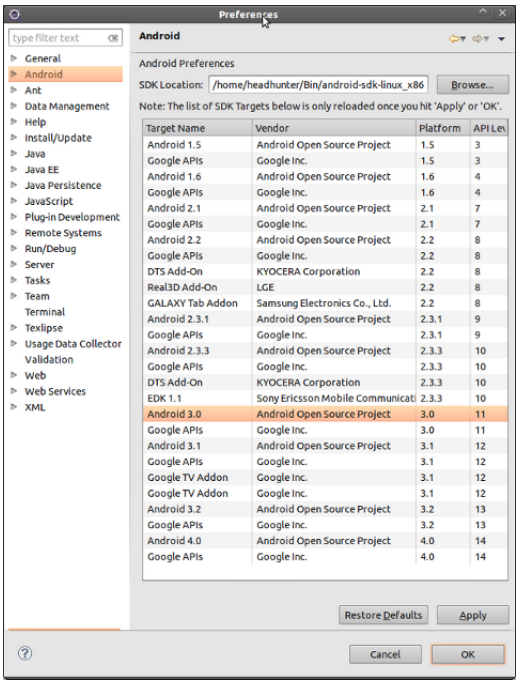

Abb. C.1: Screenshot Eclipse Window Menu

C.3.2 Android SDK

Wenn Android Applikationen gestartet werden sollen, muss ein entsprechendes virtuelles Device angelegt werden. Dies wird im Android Virtual Device Manager (AVD) angelegt. Er ist auch im Menu "Window" zu finden.

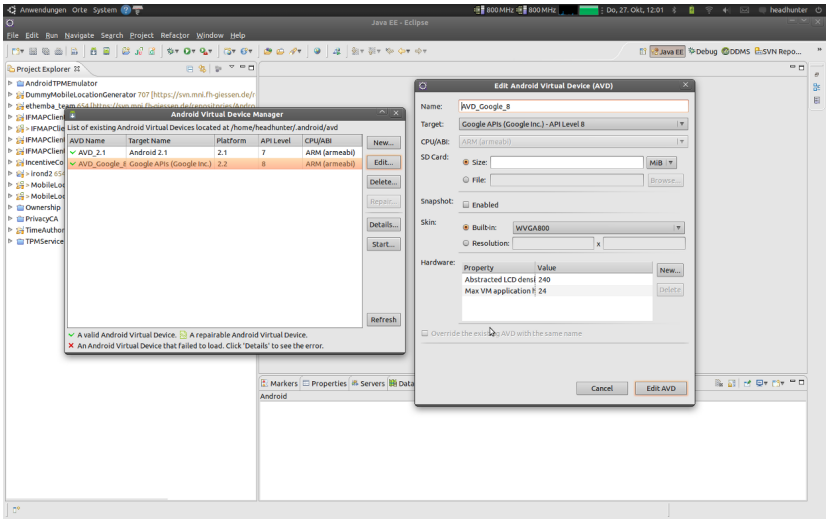

Abb. C.2: Screenshot AVD Manager

Als Target muss dabei "Google APIs (Google Inc.) - API Level 10" angegeben werden. Sollte diese API nicht zur Auswahl stehen, kann sie über den Android SDK Manager nachinstalliert werden. Sie ist unter der Rubrik Android 2.3.3 (API 10) zu finden. Die so erstellten virtuellen Devices verfügen leider nicht über TPM Funktionalität. Es ist zwar möglich ein solches Device zu erstellen, dass mit einem TPM arbeitet, aber bei unseren Arbeiten hat sich herausgestellt, dass ein per Konsole gestarteter Emulator weniger Probleme mit sich bringt.

C.4 Programme starten

Das Projekt umfasst verschiedene Programme. Den Evidence Location Generator, den Location Inspector, den MAP-Server und das Archiv. Zunächst muss der MAP-Server und das Archiv gestartet werden. Eine Beschreibung hierzu wird von mir nicht geliefert, da ich auf einen bereits konfigurierten und laufenden Server zurückgreife. Danach kann man die einzelnen Apps als Android Applikation starten. Hierzu wird ein Android Gerät oder Emulator benötigt. Ein fertig compilierter Emulator ist auf der beiliegenden CD/DVD enthalten. Dem Verzeichnis des Images ist auch ein Startscript für den Emulator beigefügt. Es gibt ein Version für Linux und eine für Windows. Wenn der Emulator gestartet ist, kann man über das Programm ADB oder über die Entwicklungsumgebung Eclipse die Apps auf dem Emulator installieren. Da nicht jedem der Umgang mit dem ADB geläufig ist, erläutere ich die Schritte, wie man per Eclipse den Start vornimmt.

Auf dem Datenträger befindet sich ein Workspace für Eclipse. In diesem ist jegliche Software enthalten, welche während des Projekts erstellt oder verwendet wurde. Zum Starten eines Programms wählt man es auf der linken Seite aus und öffnet mit einem Rechtsklick die Optionen. Dort wählt man in dem Menü "Run As" den Unterpunkt "Run Configurations..." aus.

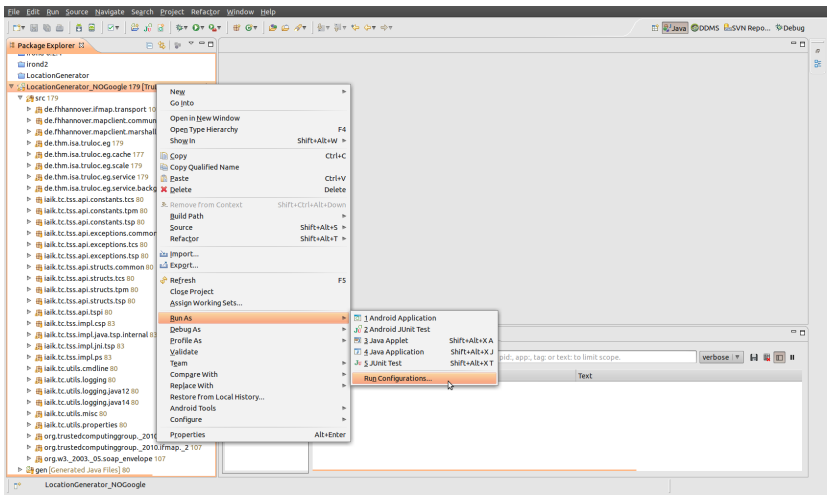

Abb. C.3: Screenshot "Run Configurations" Auswahl

In der erscheinenden Eingabemaske wählt man als "Target" die Option "Manual" aus. Dies bestätigt man mit den Tasten Apply und Run.

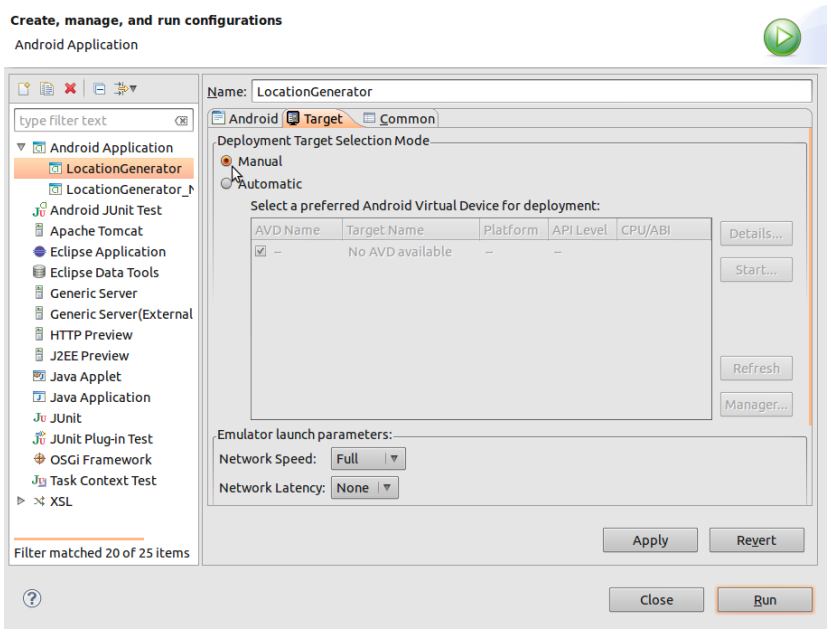

Abb. C.4: Screenshot "Run Configurations" Eingabemaske

Nach kurzer Rechenzeit sollte ein neues Fenster erscheinen, in dem man den bereits laufenden Emulator auswählen kann. Auf diesem wird dann die Applikation installiert und gestartet.

Glossar

AIK	Attestation Identity Key
AVD	Android Virtual Device Manager
CyanogenMod	Eine Firmware für verschiedene Mobilgeräte mit dem Open Source Betriebssystem Android
EG	Evidence Generator
EK	Endorsement Key
Hashtable	Als Hashtabellen werden in der Informatik Tabellen bezeichnet, welche dazu dienen Datenelemente in einer großen Datenstruktur zu finden
IFMapLib	Eine Bibliothek, welche zur Kommunikation über das IF-MAP Protokoll dient
IMA	Integrity Measurement Architecture
JDK	Java Development Kit
Mesurement Log	Ein Logfile in dem die vom TPM gemessenen Werte gespeichert werden
MI	Mobile Inspector
NFC	Near Field Communication
Overhead	Nicht primäre Nutzdaten, sondern benötigte Zusatzinformationen
PCA	Privacy Certification Authority
proprietär	proprietäre Daten sind im Gegensatz zu freier Software nicht als Quelltext erhältlich
reject	Zurückweisung einer oder mehrerer Codezeilen beim Anwenden eines Patches
RFID	radio-fequency identification
SDK	Software Development Kit
Singleton	Ein Singleton ist ein Erzeugungsmuster, mit dem sichergestellt wird, dass von einer Klasse nur ein Objekt erzeugt werden kann
SML	Stored Measurement Log
TA	Timing Authority
TC	Trusted Computing
TCG	Trusted Computing Group
TCPA	Trusted Computing Platform Alliance
TPM	Trusted Platform Module
TPM-Quote	Der TPM-Quote enthält das Mesurement Log des Bootvorgangs
URL	Uniform Resource Locator

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BL08] Brett, Andreas ; Leicher, Andreas: *ETHEMBA*. Dezember 2008. – abgerufen am 31.10.11

[BP10] Becker, Arno ; Pant, Marcus: *Android 2*. Bd. 2. aktualisierte und erweiterte Auflage. dpunkt.verlag, 2010. – ISBN 879–389864–677–2

[CJKR12] Coppolino, Luigi ; Jäger, Michael ; Kuntze, Nicolai ; Rieke, Roland: A Trusted Information Agent for Security Information and Event Management / Icons 2012. 2012. – Forschungsbericht

[Hei10] Heise Security: *Hacker liest Kryptoschlüssel aus TPM-Chip aus*. http://www.heise.de/security/meldung/Hacker-liest-Kryptoschluessel-aus-TPM-Chip-aus-926883.html. Version: Februar 2010

[Heu11] Heuser, Stephan: Vogue Projekt - Bericht AP3 / Fraunhofer Institut. 2011. – Forschungsbericht

[Kir11] Kirchner, Benjamin: Android Smartphone as Trusted Event Generator / TH Mittelhessen. 2011. – Forschungsbericht

[Kra12] Kramer, Dieter: *Konzeption und Entwicklung einer regelbasierten Verarbeitung von Events auf der Basis von Trusted Computing*, Technische Hochschule Mittelhessen, Masterthesis, 2012

[Mei10] Meierhoff, Thomas: *Reproduzierbarkeit von Zuständen im Rahmen der Emulation eines TPM*, Private Fernhochschule Darmstadt, Diplomarbeit, Januar 2010. – abgerufen am 28.11.11

[Ock10] Ockel, Arwed: Messung und Bewertung von AIKs / Fraunhofer Institut. 2010. – Forschungsbericht

[Ric09] Richter, Jennifer: *Secure Digital Evidence*, Technische Universität Darmstadt, Diplomarbeit, April 2009. – abgerufen am 10.03.12

[Ruh10] Ruhe, Tobias: *Visualisierung von Informationen einer zentralen Netzwerk-Datenbank*, FH Hannover, Masterthesis, Juli 2010. – abgerufen am 24.10.11

[Sch10] Schlichting, Markus: Android Security - Sicherheit für mobile Endgeräte / Hochschule der Medien Stuttgart. 2010. – Seminar Ausarbeitung. – abgerufen am 24.10.11

[Sch11a] Schlüter, Mark: Realisierung einer vertrauenswürdigen Geschäftsplattform auf mobilen Endgeräten / TH Mittelhessen. 2011. – Forschungsbericht

[Sch11b] Schneider, Patrick: Using the TCG Architecture for Mobile Trusted Event Documentation / TH Mittelhessen. 2011. – Seminar Ausarbeitung. – abgerufen am 24.10.11

[Sch12] Schlüter, Mark: *Realisierung einer mobilen vertrauenswürdigen Geschäftsplattform auf Basis von Trusted Computing zur gesicherten Datenerfassung*, Technische Hochschule Mittelhessen, Masterthesis, 2012

[Tru09] Trusted Computing Group: TNC Architectur for Interoperability / Trusted Computing Group. 2009 (V1.4 Rev 4). – Forschungsbericht. – abgerufen am 07.11.11

[Tru11] Trusted Computing Group: *FAQ der Trusted Computing Group*. http://www.trustedcomputinggroup.org/media_room/faqs. Version: Oktober 2011. – Stand 31.10.11

[Wei11] Weiss, Dominik: Sichere mobile Systeme / TH Mittelhessen. 2011. – Praktikumsbericht

[Wes10] Westhuis, Johannes: *Integration von Trusted Computing Technologien in die Android-Plattform*, FH Hannover, Masterthesis, August 2010. – abgerufen am 24.10.11

[Wik11a] Wikipedia: *Secure Hash Algorithm*. http://de.wikipedia.org/w/index.php?oldid=94561504. Version: Dezember 2011. – abgerufen am 01.12.11

[Wik11b] Wikipedia: *Trusted Computing*. http://de.wikipedia.org/w/index.php?oldid=92694971. Version: Oktober 2011. – abgerufen am 27.10.11

[Wik11c] Wikipedia: *Trusted Computing Group*. http://de.wikipedia.org/w/index.php?oldid=94632653. Version: Oktober 2011. – abgerufen am 31.10.11

[Wik11d] Wikipedia: *Trusted Platform Module*. http://de.wikipedia.org/w/index.php?oldid=95024207. Version: Oktober 2011. – abgerufen am 31.10.11

Danksagung

An dieser Stelle möchte ich all denen danken, die mich während meines Studiums und bei den Arbeiten an meiner Thesis unterstützt haben. Mein besonderer Dank gilt meinen Eltern, die mir durch ihre Unterstützung mein Studium ermöglicht haben. Bei Herrn Prof. Dr. Jäger bedanke ich mich für die Möglichkeit, dass ich mein Praktikum im Rahmen des Projekts TruCam an der TH-Mittelhessen absolvieren konnte. Weiterhin danke ich ihm und Herrn Dipl. Inf. Kuntze (Fraunhofer SIT) für die Betreuung meiner Abschlussarbeit.

Ein Danke für die gute Zusammenarbeit auch an meine Teamkollegen Mark Schlüter, Patrick Schneider, Dieter Kramer und Michael Eckel.

Eine weitere Person, der ich zu Dank verpflichtet bin, ist Kai Sletten, der mich während meines Studiums mit seinen hervorragenden mathematischen Kenntnissen unterstützt hat.

Datenträger