Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Jörg Eberspächer

Bachelorarbeit

Trust Analyse der HiiMap Next Generation Internet Architektur

Diplomand:	Florian Gasteiger
Matrikelnummer:	3601958
Anschrift:	Johanneck 33
	D-85307 Paunzhausen
Betreuer:	Oliver Hanka
Beginn:	19.09.2011
Abgabe:	12.03.2012

Kurzfassung

Wegen der Überlastung der heutigen Internetprotokolle wurde an der Technischen Universität München eine neue Architektur entwickelt, welche die derzeitigen Probleme überwinden kann. Die Hierarchical Internet Mapping (HiiMap) Architektur setzt dabei unter anderem auf eine Weiterentwicklung der jetzigen IP-Adresse zu einem Locator-Identifier Tupel, eine Unterbringung der gesamten Sicherheitsmechanismen in der Netzwerkschicht, sowie auf eine komplette Dezentralisierung des Internets, was eine flache Hierarchie mit sich bringt.
In der nachfolgenden Arbeit wird nun diese Architektur in Hinblick auf Sicherheit und Trust analysiert. Dabei wird mittels der Secure Tropos Methodologie das HiiMap in mehrere Einzelmodelle unterteilt und die dabei entstehenden Verbindungen auf ihre Gültigkeit überprüft.
Anhand dieser Analyse kommt man zu dem Schluss, dass HiiMap eine - auch nach neuesten Richtlinien - durchaus sichere Architektur darstellt, deren wenige Schwachstellen meist durch einfache Mechanismen eliminiert werden können und es im Vergleich zum heutigen Aufbau des Internets viele Vorteile mit sich bringt und auch für die Zukunft gerüstet ist.

Abstract

Due to the overload of todays internet protocols an institute of the "'Technische Universität"' in Munich designed a new architecture that is able to face these problems. The HiiMap Architecture is trying to archive this - upon other terms - by a separation of todays IP-Address into Locator and Identifier, putting the complete security infrastructure into the network layer and by decentralizing the whole internet, what leads to a very flat hierarchy.
The following work tries to analyze the HiiMap architecture related to Security and Trust. Using the Secure Tropos approach HiiMap gets divided into subschemes where the different dependency links can easily be checked for validity.
On the basis of this analysis HiiMap is - even concerning modern policies - a safe and trustworthy architecture and its few security risks easily can be eliminated by marginal extensions. By comparison with todays internet HiiMap has many advantages and furthermore is an architecture that can face future problems.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Als vor circa 40 Jahren das Internet entwickelt wurde ahnten die damaligen Pioniere noch nicht, dass einmal fast jeder Haushalt einen Computer besitzt und es möglich ist, portable Geräte wie Smartphones oder PDAs mit sich herumzutragen, die drahtlos auf das Internet zugreifen können. Demzufolge hat die damals entwickelte Technologie heute sehr mit der rasanten Zunahme an internetfähigen Geräten zu kämpfen. Über dieses Volumenproblem hinaus durchläuft das Internet derzeit einen Wandel dahingehend, dass nicht nur immer mehr private Personen in sozialen Netzen miteinander kommunizieren, sondern auch Behörden ihre Dienstleistungen online anbieten und sensible Unternehmen mittles e-Business einen Großteil ihrer Lieferkette über das Internet abwickeln. Da der Aspekt der Sicherheit im ursprünglichen Design des Internets nur eine sehr untergeordnete Rolle spielte, ist es heutzutage schwer die auf Grund dieser Enwicklung immer wichtiger werdenden Sicherheitsanforderungen - wie Datenschutz oder Authentizität - in angemessener Weise zu berücksichtigen.
Im Jahr 2009 hat eine Forschungsgruppe der Technischen Universität München auf der „First International Conference on Future Information Networks“ mit HiiMap eine Lösung vorgestellt, wie man die derzeitigen Probleme des Internets beheben und ein für die Zukunft gerüstetes stabiles System aufbauen könnte. Dabei wurde großer Wert darauf gelegt, dass schon während der Entwicklungsphase Aspekte wie Sicherheit und Trust berücksichtigt werden, um so eine Architektur zu entwickeln, in der die nötigen Sicherheitsmaßnahmen tief verankert sind nicht durch nachträgliche Updates und Modifikationen erst hinzugefügt werden müssen.

Die folgende Arbeit befasst sich in Kapitel 2 nun näher mit der HiiMap Architektur und deren Neuerungen. Kapitel 3 beschäftigt sich mit den derzeit aktuellen Sprachen mit denen man in der Lage ist, Sicherheit und Trust zu beschreiben. In Kapital 4 wird dann schließlich die HiiMap Architektur daraufhin untersucht, ob sie den nötigen Sicherheits- und Trustanforderungen entspricht.

Kapitel 2

Hierarchical Internet Mapping (HiiMap) Architekture

2.1 Komponenten der HiiMap Architektur

Die wichtigsten Aspekte, die in der HiiMap Architektur ([HKS+09], [HKSE09]) berücksichtigt werden, sind:

1. Die Adressierung:
   Eine der Besonderheiten der HiiMap Architektur ist die Adressierung eines Nodes mittels Locator und Identifier, deren Konzept in Kapitel 2.2 näher beschrieben wird.

   (a) Unique Identifier (UID): Hierbei bekommt jeder Teilnehmer eine eindeutige 128-bit Adresse, die nur auf besonderen Wunsch hin geändert wird. Diese UID dient dazu, ein Gerät oder einen Benutzer zu identifizieren. Allerdings kann sie nicht direkt geroutet werden. Zu der UID kommt ein 8-bit langes Präfix, das Regional Prefix (RP) genannt wird und dazu dient, die Region zu identifizieren, in der das Mapping zwischen der UID und dem korrespondierenden Locator stattfindet. Wichtig hierbei ist, dass der RP sich ändert, wenn der Benutzer z.B. in eine andere Region umzieht, wohingegen die UID immer fest einem Benutzer zugeordnet bleibt.

   (b) Local Temporary Address (LTA): Der Locator setzt sich zusammen aus der LTA und der Border Gateway Router (BGR) Adresse. Die LTA dient dazu, einen Knoten innerhalb eines autonomen Systems zu lokalisieren. Der zweite Teil des Locators, die BGR Adresse, spezifiziert den Zugangspunkt in das Netzwerk. Im Gegensatz zur UID wird die LTA benutzt um Pakete durch das Netz zu dem gewünschten Knoten zu routen. Die LTA ist im Besitz eines Providers und wird an einen Endknoten vergeben sobald sich dieser ins Netz des Providers einloggt.

Somit ändert sich die LTA auch jedesmal, wenn der Endknoten sich über einen anderen Zugangspunkt anmeldet.

(c) Mapping: Das Mapping-System muss in der Lage sein, den Zugangspunkt in das Autonomes System (AS), sowie die LTA der Knoten zu finden. Da vor allem mobile Geräte oft ihren Zugangspunkt ändern, muss es mit häufig wechselnden Zuordnungen (ChurnRates) umgehen können. Darüber hinaus muss es einem Knoten möglich sein, permanent (relocate) oder kurzzeitig (roaming) in ein anderes AS zu wechseln oder den Anbieter (Provider) zu ändern.

2. Regionen:
Im HiiMap wird vorgeschlagen, das Internet in einzelne Regionen zu unterteilen. Hierbei ist es nahe liegend, dass man die einzelnen Regionen mit Ländern gleichsetzt. Dies hat die Vorteile, dass es nur sehr selten vorkommt, dass die RPs erneuert oder verändert werden müssen, da ein Land eine sehr stabile und langlebige Instanz darstellt, weil es nicht oft geschieht, dass einzelne Länder aufgelöst werden oder neu entstehen. Bei derzeit circa 200 Staaten kann man mit einem 8-bit langen Regional Prefix jeder Region ein Land zuteilen und hat immer noch Platz für weitere Regionen. Der zweite große Vorteil, den die Gleichsetzung der Regionen mit Ländern mit sich bringt ist die relativ stabile und vor allem einheitliche Rechtsgrundlage innerhalb eines Landes. Auf diese Weise können Rechtsverletzungen von den lokalen Gerichten behandelt werden und es ist viel einfacher, vertrauenswürdige Beziehungen zwischen den Providern aufzubauen. Darüber hinaus wird vorgeschlagen, dass eine gemeinnützige Organisation oder eine Regierungsbehörde (goverment authority) die Kontrolle über die regionalen Behörden (regional authorities) eines Landes hat.
Die Regionen sind für das verlässliche Mapping ihrer Nodes verantwortlich. Das heißt auch, dass alle Mapping-Anfragen von Nodes, die dieser Region unterstehen, von dieser beantwortet werden müssen. Selbst wenn der Node sich gerade in einer anderen Region aufhält (roaming) wird das Mapping von seiner Heimregion durchgeführt.

3. Global Authority (GA):
Zusätzlich zu den Regionen wird eine übergeordnete GA benötigt. Diese hat folgende Aufgaben: Wenn der RP eines Node dem Anfragenden unbekannt ist, gibt die GA vorab den gesuchten RP zurück. Außerdem organisiert die GA die Vergabe der UIDs an die Benutzer und stellt einen Single Point of Trust (SPT) dar, was für die Vergabe von Zertifikaten jeder Art benötigt wird.
Um diese Ziele umsetzen zu können, muss die GA sehr ausfallsicher (fail safe) sein. Noch wichtiger ist, dass die GA vollkommen unabhängig und neutral gegenüber allen beteiligten Netzknoten (Node) ist. Um dies zu bewerkstelligen wird vorgeschlagen, die Verwaltung der GA den Vereinten Nationen (UN) zu unterstellen.

4. Hierarchie:
   Abbildung 2.1 zeigt die Hierarchie zwischen der GA, den Regionen und den Providern. Das eigentliche Mapping wird - wie bereits erwähnt - innerhalb der Regionen durchgeführt.

5. Bootstrap:
   Sobald ein Node sich für einen Benutzer mit dem Netzwerk verbindet sendet er an seinem Zugangspunkt die Anfrage nach einer LTA. Zusammen mit der Anfrage schickt er die UID des Benutzers und den RP. Der Zugangspunkt schickt ihm eine LTA zurück und sendet das $< UID; LTA >$ Tupel an das Mapping-System der Region, die zu dem erhaltenen RP gehört. Daraufhin wird von der Heimregion der Mapping-Eintrag aktualisiert.

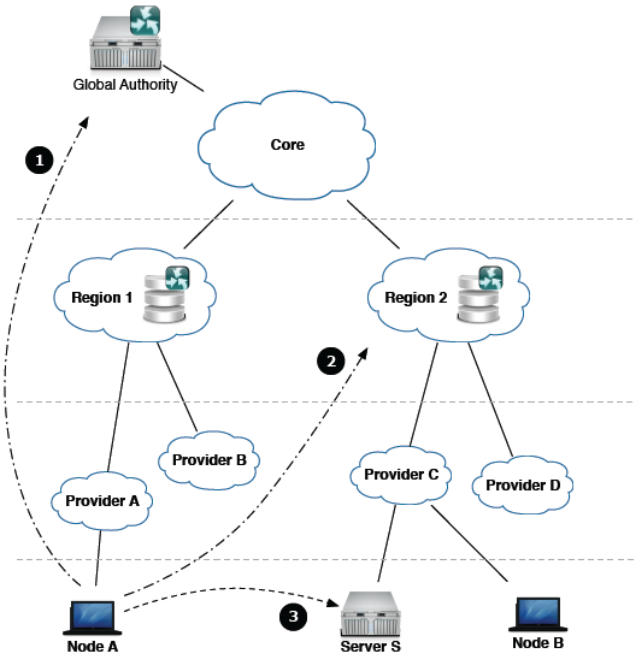

Abbildung 2.1: Hierarchische Struktur des Mapping-Systems

6. Lookup/Mapping:
   Wenn nun Node A eine Verbindung zu Server S herstellen will, werden folgende Schritte durchgeführt:
   Für den Fall, dass A die UID von S und damit auch den nötigen RP des Servers nicht kennt, muss er zuerst das globale Mapping-System der GA kontaktieren (1). Von dort bekommt er den RP der UID des Servers zurück. Danach kontaktiert A nur noch die Region von S um das Mapping für den Locator zu klären (2). Die Region gibt die LTA und die UID des BGR des AS des Providers zurück (Globally Unique Identifier (gUID)).

Im Fall von Node B, der schon mit dem gleichen Provider wie S verbunden ist, wird kein Kontakt zwischen den Anbietern (inter-provider traffic) nötig, da das Mapping von dem Mapping-Service des Providers durchgeführt werden kann (3).

7. Roaming:
   Im Falle von Roaming ändert sich der Locator eines Nodes und muss demzufolge im Mapping-System aktualisiert werden. Allerdings wird das Mapping des Nodes immer noch in seiner Heimregion gespeichert und durchgeführt. Nur die Antwort auf eine Mapping-Anfrage deutet darauf hin, dass der Node sich gerade nicht in seiner Heimregion aufhält. Dies wird dadurch deutlich, dass die gUID kein Teil der Region ist, bei der angefragt worden ist.

8. Relocation:
   Relocation bezeichnet den dauerhaften Wechsel der Region eines Nodes oder eines ganzen Unternehmens (mehrere Nodes). Wenn ein Unternehmen seinen Provider wechselt, musste es bisher eventuell alle Internet Protocol (IP)-Adressen seiner öffentlich sichtbaren Nodes in ein Subnet des neuen Providers ändern oder es bekommt die Möglichkeit von dem alten Provider die IP-Adressen zu übernehmen. Für den letzten Fall wachsen die Tabellen (Global Address Space) noch mehr und auch die Routing Tabellen vergrößern sich.
   Falls ein Unternehmen nicht nur den Provider sondern auch das Land wechselt, wird natürlich auch der Domain-Name geändert. Dies ist aber in den meisten Fällen nicht erwünscht und darum wird der alte Domain-Name oft zusätzlich zu dem neuen weiter verwaltet, was natürlich auch wieder einen Mehraufwand für die Domain Name System (DNS) darstellt.
   In der HiiMap Architektur wird versucht, die Duplizierung der Adressen und den zusätzlichen Verwaltungsaufwand zu umgehen. Wenn jetzt ein Unternehmen das Land und somit auch die Region wechselt, bleiben dennoch die UIDs erhalten. Der alte RP wird ungültig und Nodes, welche die Person oder das Unternehmen erreichen wollen, müssen somit nur ihr lokal gespeichertes RP (im Cache) aktualisieren. Dies wird automatisch durchgeführt, indem die GA kontaktiert wird und von dort das aktuelle RP beschafft wird.

2.2 Locator/Identifier Separation Protokoll (LISP)

Eines der Probleme, die durch die rasante Zunahme an internetfähigen Geräten auftreten, ist die Adressierung der Geräte. Das IPv4 Protokoll, das mit einem 32-bit Adressraum arbeitet und somit nur etwas über 4 Milliarden Adressen zulässt, ist den heutigen Anforderungen kaum noch gewachsen. Durch das IPv6 Protokoll wurde der Adressraum zwar auf 128 Bit erweitert, doch wird dieses Protokoll in der Praxis wenig verwendet. Ein weiteres Problem findet sich in den Routing Tabellen des Border Gateway Protocol (BGP), die eine kaum noch zu bewältigende Anzahl an Einträgen verwalten müssen.
Aus diesem Grund wurde in den letzten Jahren nach Möglichkeiten gesucht, diese Probleme zu beheben. Eine Methode, die auch in der HiiMap Architektur verwendet wird, ist die Ablösung der IP-Adressen durch die Einführung des LISP ([IB07],[QIDLB07]). LISP basiert auf einem einfachen IP-over-IP Tunnelprinzip, das typischerweise auf Border Routern implementiert wird, die als Routing Locator (RLOC)s für die Endsysteme der lokalen Domain agieren. Die Endsysteme senden und empfangen auch hierbei Pakete mittels IP-Adressen, die jetzt aber Endpoint Identifier (EID)s heißen. Die grundsätzliche Idee des LISP besteht darin, Pakete vom RLOC der QuellEID über das Kernnetz zu dem RLOC der ZielEID zu tunneln. Bei einem end-to-end Paketaustausch zwischen zwei Internet Hosts erweitert der Ingress Tunnel Router (ITR) jedes Paket um einen neuen LISP Header, den der Egress Tunnel Router (ETR) wieder entfernt, bevor er das Paket Richtung Zielpunkt weiterleitet.
Dies hat den Vorteil, dass im Kernnetz keine lokalen EIDs angesprochen werden müssen, sondern nur RLOCs, die für das richtige Tunneln benötigt werden. Dadurch entsteht auch der größte Vorteil der Locator/Identifier Trennung: Die Reduktion der BGP Routing Tabellen.

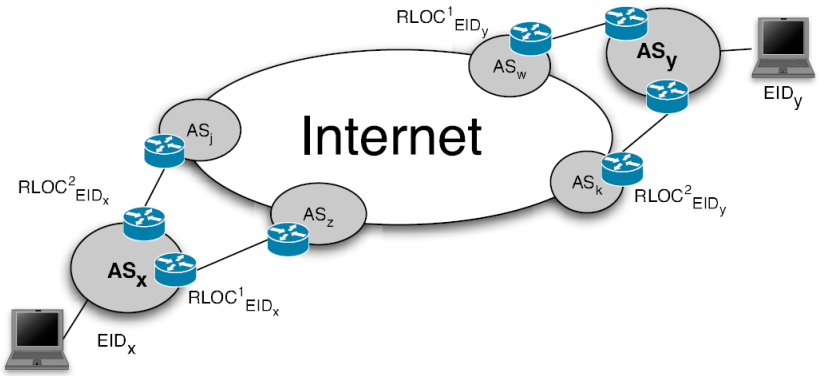

Abbildung 2.2: Positionierung der EIDs und RLOCs im Kernnetz

Wenn nun z.B. EID_x eine Verbindung zu EID_y aufbauen will, müssen die folgenden Schritte ausgeführt werden:

- EID_x schickt ein erstes IP Paket mit seiner ID (EID_x) als Startadresse (SA) und benutzt die EID_y als Zieladresse (ZA). Dieses Paket wird innerhalb des AS_x auf die übliche Weise geroutet, um einen der EID_x RLOCs zu erreichen.
- Der ITR ($\text{RLOC}^1_{\text{EID}_x}$) empfängt das Paket.
  In der Praxis wird dies durch Interdomain-Routing oder TE-Verfahren in Verbindung mit dem lokalen EID-to-RLOC Mapping durchgeführt. Diese Verfahren können von einem AS zum anderen variieren. Dennoch ist die EID von außerhalb durch all seine RLOCs erreichbar. Jeder RLOC kennt die lokalen EID-to-RLOC Mappings, die in der lokalen Datenbank gespeichert werden.
- $\text{RLOC}^1_{\text{EID}_x}$ führt eine EID-to-RLOC Suche in seinem lokalen Cache durch, um den Routing Pfad zum Locator von EID_y zu finden. Hierbei ist zu beachten, dass die eventuell nötige Erstellung eines neuen Mapping-Eintrags mehrere Operationen benötigt, wie die Übertragung einer Suchanfrage auf einen Mapping-Verteilungsdienst. Im Weitern wird davon ausgegangen, dass die Suche im lokalen Cache nun den $\text{RLOC}^2_{\text{EID}_y}$ zurück gibt.
- Ein neuer LISP Header wird dem ursprünglichem IP Paket vorangestellt mit $\text{RLOC}^1_{\text{EID}_x}$ als SA und $\text{RLOC}^2_{\text{EID}_y}$ als ZA. Das Paket wird nun mittels IP in das Kernnetz geroutet. Dabei müssen die Router im Netz nicht zu der EID_y routen. Um ein Paket richtig zu übermitteln, werden nur Routinginformationen für die RLOCs benötigt.
- Wenn das Paket $\text{RLOC}^2_{\text{EID}_y}$ erreicht wird der vorangestellte LISP Header wieder entfernt. Das Paket wird dann ganz normal innerhalb des AS_y zu EID_y übermittelt.

Anmerkung: Nachdem das erste Paket durch den LISP Tunnel geschickt wurde, haben die Caches beider Endpunkte die nötige Information um alle folgenden Pakete richtig zu übermitteln.

2.3 Threshold Kryptographie

Der Sicherheitsgedanke hatte in der ursprünglichen Architektur des Internets nur einen sehr geringen Stellenwert. Allerdings wurde genau das in den letzten Jahren zu einem der größten Herausforderungen der Internet Architekten. Für jede neu entdeckte Sicherheitslücke wurde eine Veränderung nur in der betroffenen Komponente durchgeführt. Das hatte zur Folge, dass sich heute auf unterschiedlichen Schichten viele verschiedene und teilweise sehr spezifische Sicherheitslösungen finden. In der HiiMap Architektur wurde nun versucht einen einheitlichen Sicherheitsmechanismus tief in der Netzwerkschicht zu verankern, so dass ein sauberer und konsequenter Ansatz für alle Teilnehmer zu Verfügung gestellt wird. Darüber hinaus wurde auch von Anfang an berücksichtigt, dass eine vertrauenswürdige Verbindung zwischen zwei Partnern aufgebaut wird.
Eine der grundlegenden Sicherheitsmechanismen, die in der HiiMap Architektur angewandt werden, stellt die Threshold Kryptographie dar. Diese basiert auf dem von Adi Shamir 1979 entwickelten "Secret Sharing" Verfahren, das es möglich macht, ein Geheimnis (Secret) auf mehrere Instanzen aufzuteilen, wobei aber nur eine gewisse Untermenge dieser Instanzen erforderlich ist, um das Geheimnis wieder rekonstruieren zu können. Für das „Secret Sharing" Verfahren benötigt man einen sogenannten „Dealer", dem alle Instanzen vertrauen, da dieser das ganze Geheimnis kennt und dieses auf die anderen Parteien verteilen muss.

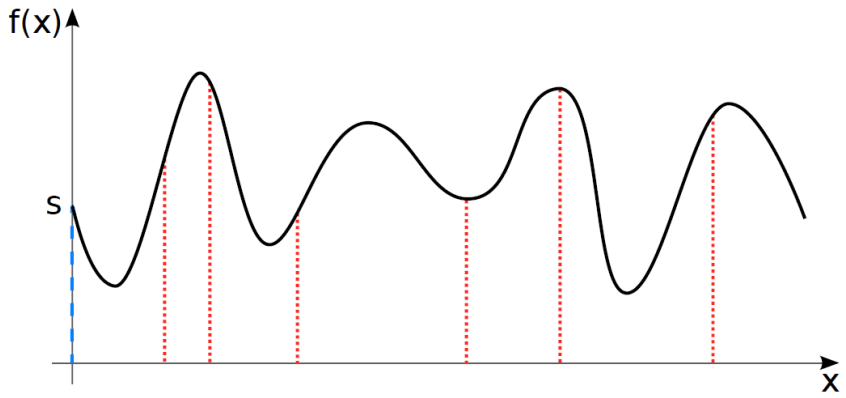

Abbildung 2.3: Geometrische Repräsentation von Shamir's „Secret Sharing"

Shamir's Verfahren beruht auf der Konstruktion eines Polynoms mit dem Muster $f(x) = s + a_1x + a_2x^2 + a_3x^3 + \ldots + a_{k-1}x^{k-1}$, wobei s das Geheimnis repräsentiert. Somit können alle a_i vom Dealer uneingeschränkt bestimmt werden. Daraufhin generiert der Dealer n beliebige Wertepaare $(x, f(x))$, bei denen x bekannt gemacht werden darf, aber $f(x)$ geheimgehalten werden muss, da es ein Teil des Geheimnisses ist. Um das ganze Geheimnis wieder rekonstruieren zu können werden nur $k < n$ Teile gebraucht.

In der HiiMap Architektur erhält jeder User von der GA eine Smart Card mit seiner UID, dem Private Key und Public Key. Die Threshold Kryptographie wird nun dazu benutzt, Letzteren in mehrere Stücke zu unterteilen und diese Stücke an verschiedene Regionen zu schicken. Dabei bekommt der Benutzer von der GA noch eine Lookup Tabelle (Mapping Directive), in der er nachschlagen kann, auf welche Regionen er seine Schlüsselstücke aufteilen soll. Dies hat den Vorteil, dass nicht nur eine Region (seine Heimregion) den Public Key des Benutzers kennt, sondern dieser auf mehrere Regionen aufgeteilt ist und es somit nicht mehr möglich ist, dass eine Region ihr Macht missbraucht und dadurch eine sichere Kommunikation zwischen zwei Benutzern verhindern kann.
Sollte bei Anwendung der Threshold Kryptographie eine Region versuchen, einem Benutzer ein falsches Schlüsselstück zu senden, kann der Benutzer im schlimmsten Fall den Public Key nicht mehr zusammensetzen. Daraufhin kontaktiert er einfach andere Regionen, die weitere Stücke des benötigten Public Keys gespeichert haben und setzt den Schlüssel anhand dieser Teile zusammen. Bei häufiger auftretenden Manipulationsversuchen durch eine einzelne Region kann diese von der GA zu Sanktionen verurteilt werden oder es werden einfach keine Schlüsselstücke mehr dort hinterlegt.

Kapitel 3

Modellierungssprachen für Trust

3.1 Verschiedene Modellierungsansätze

Trust wurde in den letzten Jahren in einer Vielzahl von Forschungsgebieten, wie z.B. Wirtschaft, Marketing, Medizin, Computerwissenschaften zu einem zentralen Thema.

Das beliebte Zitat *„Trust is the willingness to rely on a specific other, based on confidence that one's trust will lead to positive outcomes“* ([CW02]) ist eines der zahlreichen Beispiele, in denen der so vielseitig verwendete Trustbegriff zu erklären versucht wird. Auf Grund der weit gestreuten Anwendungsbereiche gibt es auch eine große Bandbreite an Sprachen oder Frameworks, mit denen es möglich ist Trust zu beschreiben und zu modellieren.

Im Folgenden werden nun einige dieser Frameworks vorgestellt und deren Vor- und Nachteile in Bezug auf die Analyse der HiiMap Architektur erklärt.

Trustcom Framework

Das Trustcom Framework [WASE05] ist ein Ansatz, der versucht Geschäftsverträge in Hinblick auf Trust und Sicherheit innerhalb virtueller Organisationen darzustellen. Die Hauptaufgabe des Trustcom Frameworks ist dabei, eine vertragliche Vereinbarung zwischen Mitgliedern von virtuellen Organisationen auf einer Geschäftsebene zu definieren und diese auf einer technischen Ebene innerhalb einer serviceorientierten Architektur zu spezifizieren, überwachen und aktualisieren. Da dieses Framework allerdings von seinem gesamten Aufbau nur für die Analyse von Verträgen entwickelt wurde, ist es kaum möglich Trustcom für die Analyse einer Internetarchitektur zu verwenden.

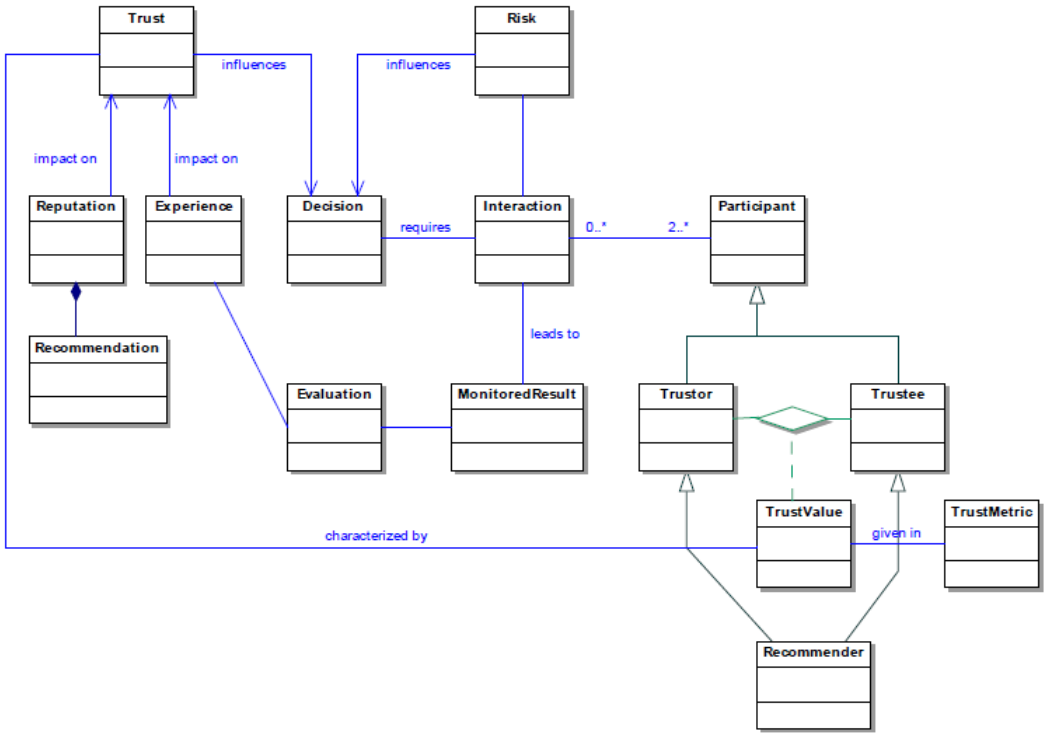

Abbildung 3.1: Modellierung mit Trustcom Framework

A Language for Modelling Trust in Information Systems

Diese Modellierungssprache [BMP10] beschreibt ein Vorgehen, bei dem bereits während der Entwicklung eines Informationssystems Trust berücksichtigt wird. Die Sprache kann als eine Weiterentwicklung der in [BMP07] gesehen werden, für die Analysen von anderen bereits existierenden Ontologien durchgeführt wurden. Die Modellierung an sich erfolgt ähnlich wie bei allen anderen Frameworks durch die Verwendung von Syntaxelementen wie Aktoren, Zielen und Security Constraints. Da Trustverbindungen hier eine Gewichtung erhalten ist es im Gegensatz zu anderen Sprachen nötig, dass das zu betrachtende System bereits implementiert wurde oder sonst ausreichend viele Erfahrungswerte bereitstehen, da die Trustverbindungen mit konkreten Zahlenwerten versehen werden müssen.

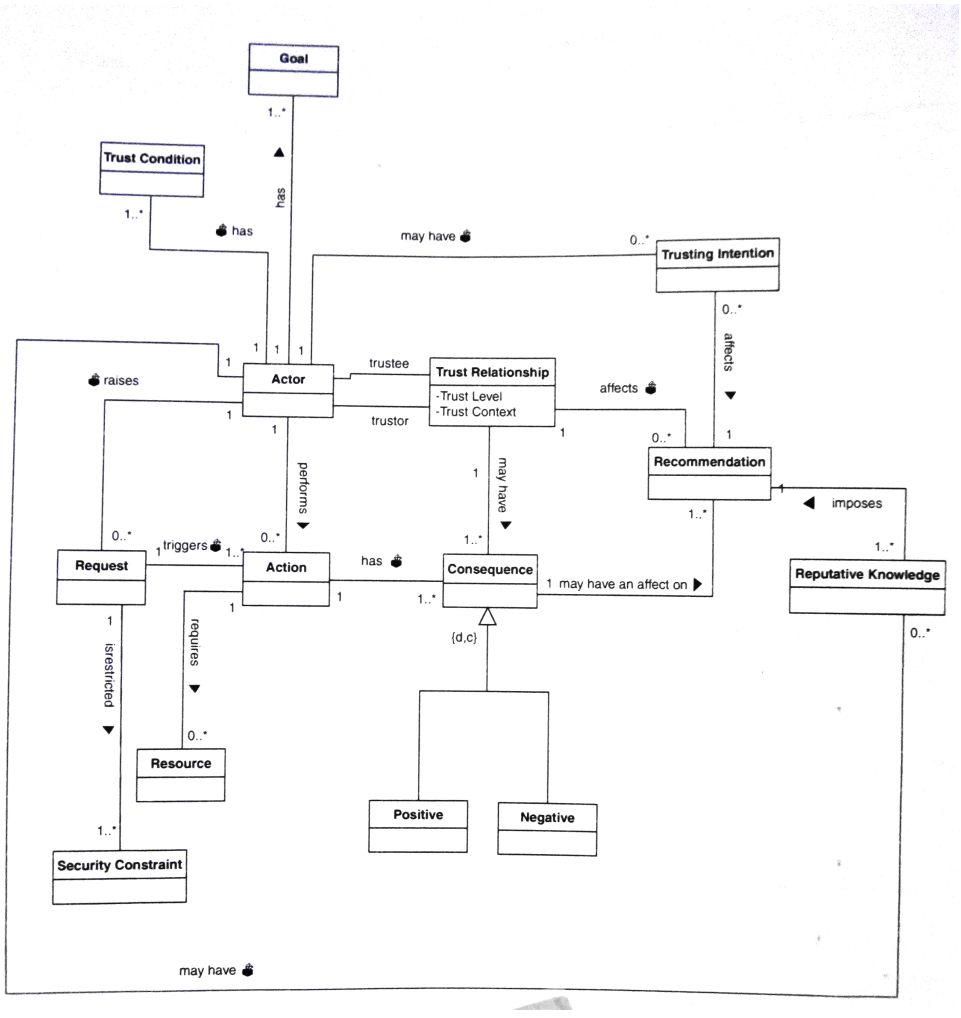

Abbildung 3.2: Metamodel für Modellierung

I* Framework

Das 2001 entwickelte I*Framework ist einer der ersten Versuche, Trust in einer frühen Phase des Designprozesses einbringen, um Trust-Aspekte als treibende Kraft für die Entwicklung potenzieller Designalternativen nutzen zu können. Das I*Framework [YL01] wurde entwickelt um Requirement Analysen und High-Level Design in einem Aktor-orientierten System zu unterstützen. Es modelliert bewusste Abhängigkeitsbeziehungen zwischen strategischen Aktoren und deren Zielen. Da Aktoren voneinander abhängig sind um ihre Ziele zu erreichen, Aufgaben durchzuführen oder Resourcen bereitzustellen, müssen Trust Beziehungen zwischen den Aktoren bedacht werden, um über die Möglichkeiten und Schwachstellen nachzudenken, die diese Beziehungen mit sich bringen. Das Prinzip des Softgoals wird genutzt, um Qualitätsattribute zu modellieren, für die a-priori keine klar abgegrenzten Kriterien zur Befriedigung des Ziels bereitstehen. Im I*Framework wird Trust wie ein solches Softgoal behandelt, dessen Befriedigung vom Standpunkt des Aktors abhängt.
Im Wesentlichen werden zwei Modelle unterschieden. Das Strategic Dependency (SD) Modell, das ein Netzwerk von Beziehungen zwischen den einzelnen Aktoren beschreibt und das Strategic Rationale (SR) Modell, das genutzt wird, um Absichten der Aktoren zu beschreiben und zu unterstützen, die Beziehungen mit anderen Aktoren eingehen.
Die Sichtweisen auf Möglichkeiten und Schwachstellen werden im SD Modell in das zentrale Konstrukt der beabsichtigten Abhängigkeiten eingebaut. Um Trust zu modellieren, muss der Designer sich an dieser Stellen nur noch die Frage stellen, ob jede einzelne dieser Abhängigkeiten in einem solches Modell realisierbar ist. Ob der Depender dem Dependee vertrauen kann, wenn er ihm das Dependum übergibt ist oft der zentrale Punkt bei der Frage nach der Realisierbarkeit.

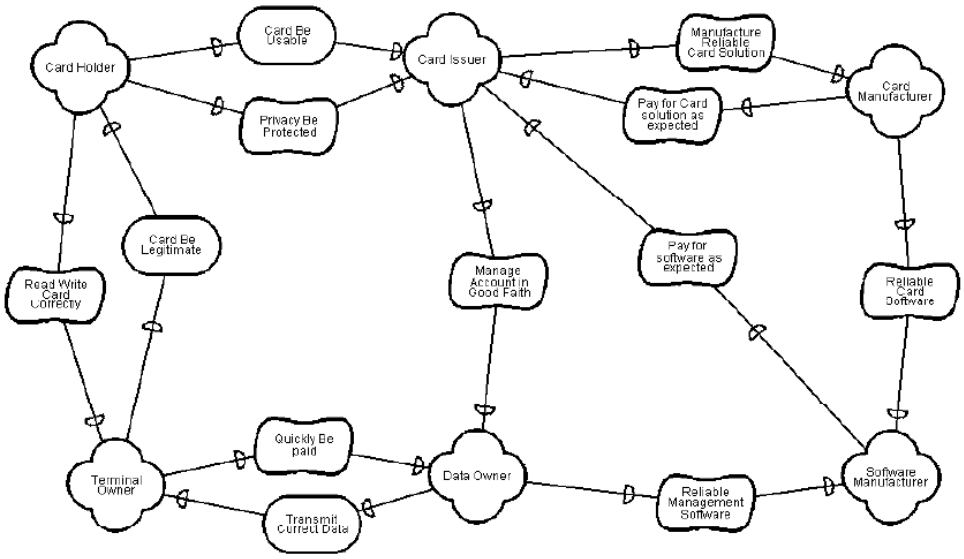

Abbildung 3.3: Modellierung mit I* Framework

Trust-Confidence-Distrust (TCD) Framework

Im Laufe der letzten Jahre entstand eine Vielzahl an Erweiterungen des I*Frameworks. Ein Ansatz dabei war das TCD Framework [KGSS04]. Dieses wurde primär entwickelt, um Anwendungen in einem sozialen Netzwerk zu beschreiben, was von Gulati (1998) beschrieben wurde als ein „set of nodes“ (Personen, Organisationen), welches mit einem „set of social relationships“ (Freundschaften, etc.) eines speziellen Typs verbunden ist. Es adoptiert dabei zu einem gewissen Teil die im I*Framework entwickelten Notationen und Grundgedanken. Allerdings ist es für die Analyse einer Internet Architektur nicht geeignet.

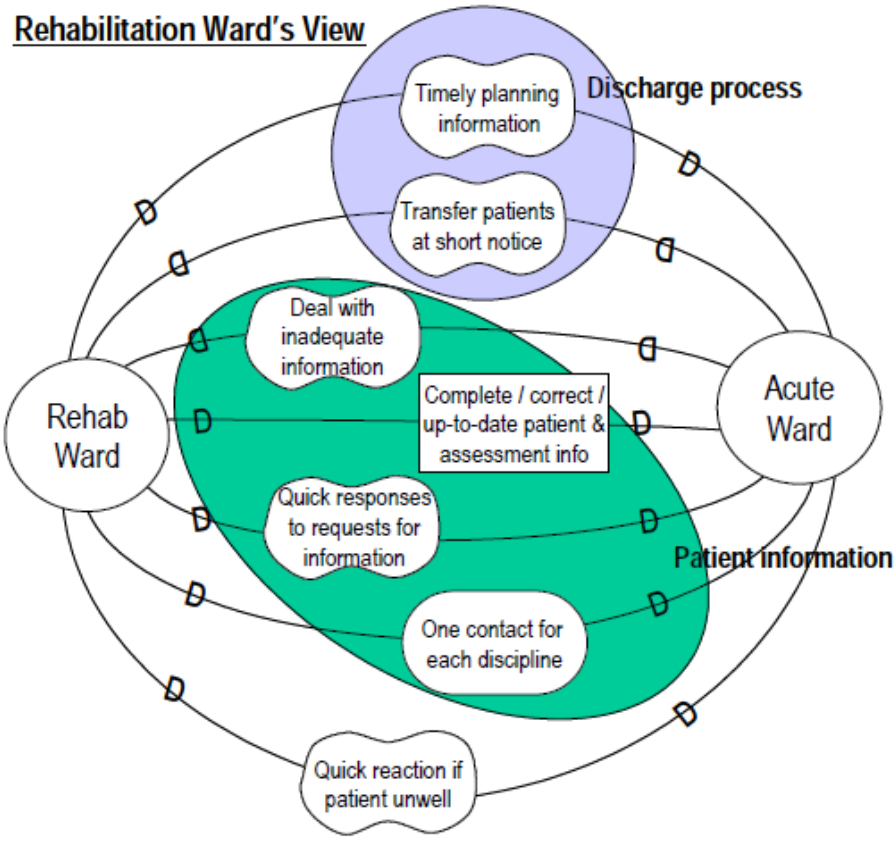

Abbildung 3.4: Modellierung mit TCD Framework

Tropos

Ein weiterer Ansatz, der ebenfalls die Notation des I*Frameworks verwendet, ist Tropos [GMZ08]. Hierbei handelt es sich um eine Zusammenfügung des sicherheitsbasierten Ansatzes von Mouratidis und dem trustbasierten Ansatz von Giorgini, deren Ergebnis eine Methodik ist, die sowohl Sicherheit wie auch Trust als Teil des Entwicklungsprozesses betrachtet. Mit Tropos wird eine Software-Entwicklungsmethodik vorgestellt, die darauf spezialisiert ist, die Umgebung eines Systems und auch das System selbst zu beschreiben. Somit stellt es einen sehr passenden Ansatz dar, um die HiiMap Architektur zu modellieren. Der Vorgang, Sicherheit und funktionale Anforderungen in den gesamten Entwicklungsprozesses einzubinden ist allerdings ziemlich ad hoc. Darüber hinaus versagt oft das Prinzip der Soft-Goals, die Tropos benutzt um Sicherheitsanforderungen umzusetzen, bei der adäquaten Erfassung einiger der Einschränkungen, die in Sicherheitsanforderungen häufig genutzt werden. Darüber hinaus gibt es in Tropos keine Möglichkeit, Trust-Verbindungen zu modellieren.

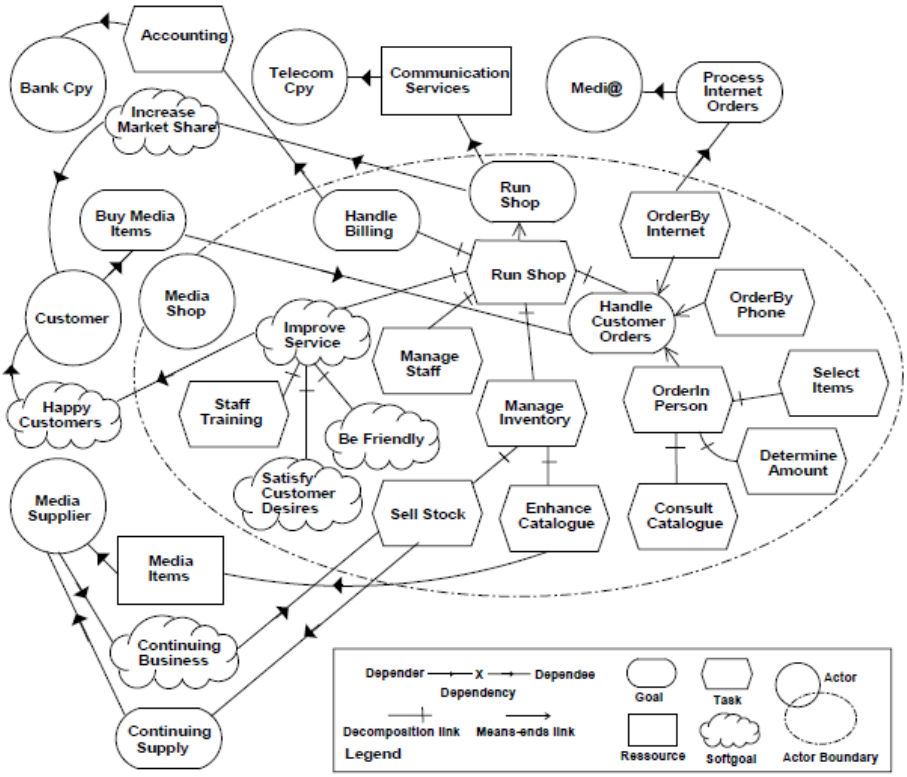

Abbildung 3.5: Modellierung mit Tropos

3.2 Secure Tropos

Um die Einschränkungen des ursprünglichen Tropos Ansatzes zu überwinden, haben sich zwei Arbeitsgruppen gebildet. Die Eine beschäftigt sich damit, Sicherheitsanliegen in einer passenden Art und Weise modellieren zu können. Mit dieser Modifikation, die Secure Tropos genannt wird, werden Sicherheitsauflagen (Security Constraint) und Sicherheitsfähigkeiten (Security Capability) als Grundelemente in allen Entwicklungsschritten der Modellierung eingeführt. Die zweite Arbeitsgruppe, deren Modifikationen später ebenfalls Secure Tropos genannt wurden, führte zusätzlich Konzepte wie Besitzer (Ownership), Trust und Bevollmächtigung ein. Die Idee dahinter ist die Differenzierung zwischen dem „Anbieten eines Dienstes“ und dem „Besitzen eines Dienstes“, sowie die Unterscheidung zwischen funktionalen Abhängigkeiten und Trust Beziehungen.
Mit diesen beiden Erweiterungen wird mit Secure Tropos ein Verfahren zur Verfügung gestellt, mit dem man auch so komplexe Strukturen wie eine Internet Architektur und damit auch HiiMap in einfache Modelle aufteilen kann und anhand dieser Modelle sowohl die funktionalen Beziehungen wie auch die Trustbeziehungen zwischen den einzelnen Aktoren beschreiben kann. ([MGM03], [GMZ08], [MGM05], [MG09])

Syntax

Wie bereits erwähnt, übernimmt Secure Tropos zu einem Großteil die Elemente aus dem I*Framework. Im Folgenden findet sich eine Auflistung der verwendeten Syntaxelemente und deren Bedeutung. Dabei werden bewusst nur die Elemente vorgestellt, die in dieser Arbeit verwendet wurden um die HiiMap Architektur zu analysieren. Abbildung 3.6 zeigt die in den Modellen verwendete graphische Umsetzung der einzelnen Elemente.

a) Aktor: Ein Aktor repräsentiert eine Einheit, die strategische Ziele und Absichten innerhalb eines Systems hat.

b) Security Constraint: Eine Sicherheitsbedingung ist definiert als eine auf Sicherheitsdienste wie z.B. Integrität, Privatsphäre oder Verfügbarkeit bezogene Beschränkung, welche die Analyse und das Design eines Systems beeinflussen kann.

c) Plan: Ein Plan repräsentiert in einer abstrakten Art und Weise, wie etwas getan wird. Die Ausführung eines Plans kann ein Mittel sein, ein Ziel zu befriedigen.

d) Secure Plan: Ein sicherer Plan ist ein Plan der einen Weg darstellt wie ein sicheres Ziel befriedigt werden kann.

e) Hard Goal: Ein Ziel repräsentiert die strategischen Interessen eines Aktors.

f) Secure Hard Goal: Sichere Ziele sind Interessen eines Aktors mit einem Bezug zur Sicherheit. Sichere Ziele werden hauptsächlich verwendet um Security Constraints einzuhalten, die von einem Aktor gefordert werden oder im System existieren.

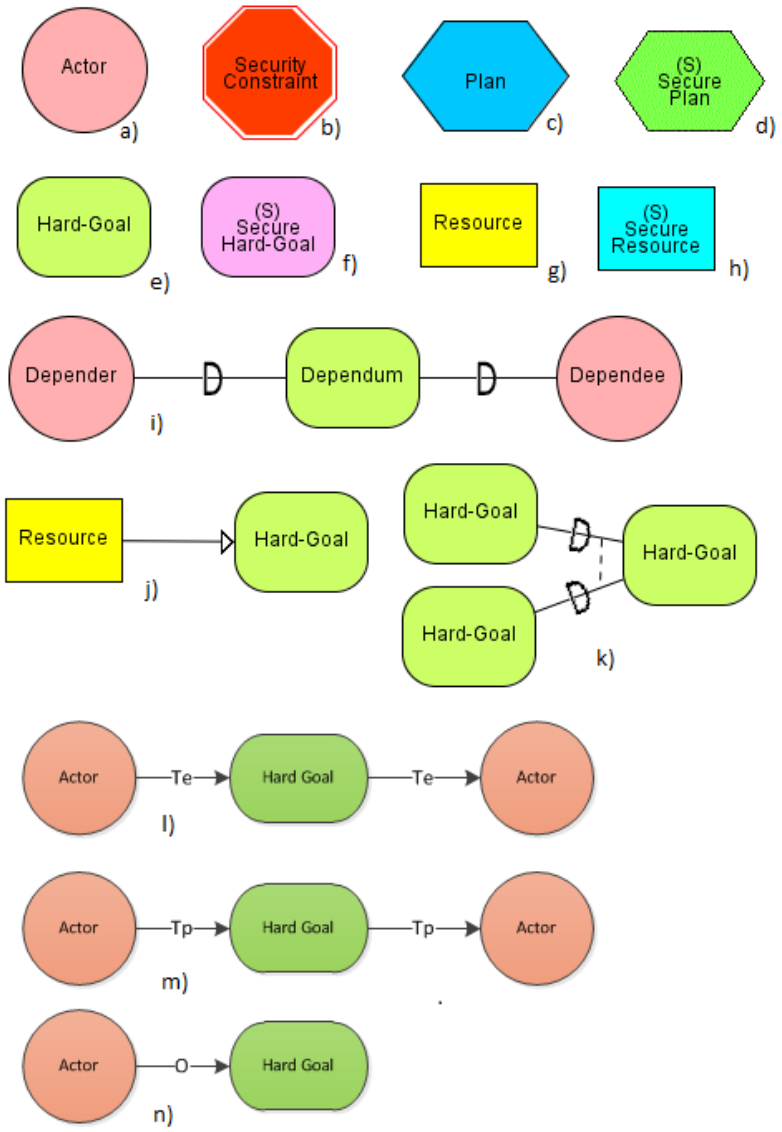

Abbildung 3.6: Verwendete Syntaxelemente

Ein sicheres Ziel zeigt allerdings nicht den speziellen Weg auf, wie ein Security Constraint erfüllt wird .

g) Resource: Ressourcen stellen physische Einheiten oder Informationseinheit dar. Ressourcen haben keine Ziele, sie stehen immer nur am Ende einer Dependency-Kette.

h) Secure Resource: Eine sichere Ressource ist eine Ressource, die Bezug zur Sicherheit des Systems hat. Eine Sichere Ressource wird oft als Oberbegriff für Sicherheitscharakteristika verwendet, die von Aktoren gefordert werden (z.B. sichere Ziele, sichere Pläne)

i) Dependency-Link: Abhängigkeiten zwischen zwei Aktoren zeigen an, dass ein Aktor von einem anderen Aktor abhängig ist um ein Ziel zu erreichen, einen Plan auszuführen oder eine Ressource zu übergeben.
Der erste Aktor wird Depender genannt und der zweite Dependee. Das Objekt das im Mittelpunkt der Abhängigkeit steht heißt Dependum.

j) Means-End-Link: Mittel zum Zweck Beziehungen zeigen an, dass z.B. eine Ressource verwendet wird, ein Ziel zu erreichen.

k) AND Verknüpfung: Eine UND-Verknüpfung verdeutlicht, dass mehrere Abhängigkeiten erfüllt werden müssen, damit ein Ziel erreicht werden kann.

l) Trust by Execution: Trust of execution zwischen zwei Aktoren zeigt, dass ein Truster glaubt, dass der Trustee dazu in der Lage ist, ein Ziel zu erfüllen oder einen Plan auszuführen. Im Allgemeinen nimmt ein Aktor, wenn er dem Anderen vertraut, an, dass der andere Aktor das Trustum richtig übergibt.

m) Trust by Permission: Trust by permission zwischen zwei Aktoren zeigt an, dass der Truster darauf vertraut, dass der Trustee seine Befugnisse nicht missbraucht um ein Ziel zu erreichen.

n) Ownership: Die Besitz-Relation zeigt an, dass ein Aktor der rechtmäßige Besitzer eines Ziels, Plans oder einer Ressource ist. Besitzer haben volle Befugnis bei der Ausführung ihrer Pläne oder Ziele.

Aufbau

Die grundsätzliche Vorgehensweise bei der Beschreibung eines Systems mit Secure Tropos ist ein iterativer Prozess mit verschiedenen Modellierungsschritten um mehrere Arten von Aktor-Diagrammen und Ziel-Diagrammen zu erhalten. Die Modelle, die in einem Schritt produziert werden, werden als Input für nachfolgende Abläufe verwendet. Allgemein beginnt die Modellierung, indem man eine Reihe von Aktoren mit speziellen Zielen definiert und diese mit Hilfe von Dependency-Links miteinander in Verbindung bringt. Daraufhin werden Trustbedingungen und Security Constraints aufgestellt, welche die Aktoren bei der Erfüllung ihrer Ziele behindern könnten.

Man kann die Modellierung eines System mit Secure Tropos im wesentlichen in vier Stufen unterteilen:

1. Early Requirements: In dieser ersten Phase wird ein System anhand der bestehenden Organisationsstruktur analysiert. Zu Beginn wird ein erstes Aktor-Diagramm aufgestellt, in dem die relevanten Aktoren zusammen mit ihren Zielen dargestellt werden. In mehreren Durchläufen wird das Diagramm mit Hilfe von Plan/Ziel und Trust Mo-

dellen soweit verbessert, bis ein Modell vorliegt in dem alle relevanten Aktoren, deren Abhängigkeiten und Sicherheitsbeschränkungen berücksichtigt wurden.

2. Late Requirements: In der zweiten Phase wird nicht länger nur die Umwelt beschrieben, in der sich das System befindet. Vielmehr wird das System an sich innerhalb dieser Umwelt näher betrachtet. Dabei wird das System als eine kleine Zahl von Aktoren aufgefasst, die eine bestimmte Menge von Abhängigkeiten nach sich zieht. Diese Abhängigkeiten definieren die funktionalen Anforderungen des Systems. Wie in der Early Requirement Phase ist auch dieser Schritt ein Prozess, der sich über mehrere Runden strecken kann und auch die hier entwickelten Dependency und Trust Modelle werden iteriert.
3. Architectural Design: In dieser dritten Phase wird die globale Struktur des Systems näher definiert. Dabei wird diese als eine Menge von Untersystemen definiert, die durch Daten- und Kontrollströme verbunden sind. Innerhalb dieses Frameworks werden die Untersysteme als Aktoren und die Daten-/Kontrollströme als Abhängigkeiten modelliert.
4. Detailed Design: In der letzten Phase wird das Architectural Design noch detaillierter in verschiedene Eingangs-, Ausgangs- und Kontrollsysteme unterteilt.

Um die einzelnen Stufen der Modellierung umzusetzen wird eine Vielzahl von Modellierungsmöglichkeiten vorgestellt.

- Actor modelling:
  beinhaltet die Identifikation und Analyse sowohl der Aktoren in der Umwelt des Systems als auch der Aktoren innerhalb des Systems.
- Dependency modelling:
  Hierbei wird versucht Aktoren zu identifizieren, die von anderen Aktoren abhängig sind um Ziele zu erreichen, Pläne umzusetzen oder Ressourcen zu verwalten.
- Goal and plan modelling:
  Diese Modelliereungsmöglichkeit befasst sich mit der Analyse der Ziele eines Aktors ausgehend von seinem Bickpunkt. Dazu stehen drei verschiedene Analysetechniken zur Verfügung:
  Means end Analyse, Contribution Analyse und AND/OR Analyse.
- Trust of permission modeling:
  Die Trust of permission Modellierung dient dazu Aktoren zu finden, die anderen Aktoren ihre Ziele, Pläne oder Ressourcen anvertrauen oder Aktoren zu finden, die Ziele, Pläne oder Ressourcen besitzen.
  In der Early Requirement Phase konzentriert sich diese Modellierungsmöglichkeit darauf, Trust-Beziehungen zwischen sozialen Aktoren und deren Umwelt (organisational setting) zu finden. In der zweiten Phase fokusiert sich die Trust by permission Modellierung darauf, Trust-Beziehungen des Aktors „System“ zu finden.

- Security constraint modelling:
  Diese Modellierungsmöglichkeit befasst sich mit dem Finden von Security Constraints, die von Aktoren gefordert werden. Die Security Constraint Modellierung kann in mehrere Untergruppen aufgeteilt werden:
  - security constraint delegation: Modellierung von Delegierungen von Security Constraints zwischen zwei Aktoren
  - security constraint assignment: Modellierung von Zuordnungen zwischen einem Security Constraint und einem Ziel
  - security constraint analysis: Abbauen von Security Constraints und Identifizierung von sicheren Zielen, die Security Constraints ins System einbringen.
- Secure entity modelling:
  Hier wird versucht sichere Pläne und Ressourcen zu finden, die für die Befriedigung eines sicheren Ziels nötig sind. Auch werden sichere Ziele oder Pläne in Unterziele/Unterpläne aufgeteilt. Wie auch beim Security Constraint Modelling folgt das Secure entity modelling den gleichen Beschreibungstechniken wie means-end oder contribution Analysen, die von Secure Tropos eingesetzt werden um Ziele und Pläne zu analysieren.

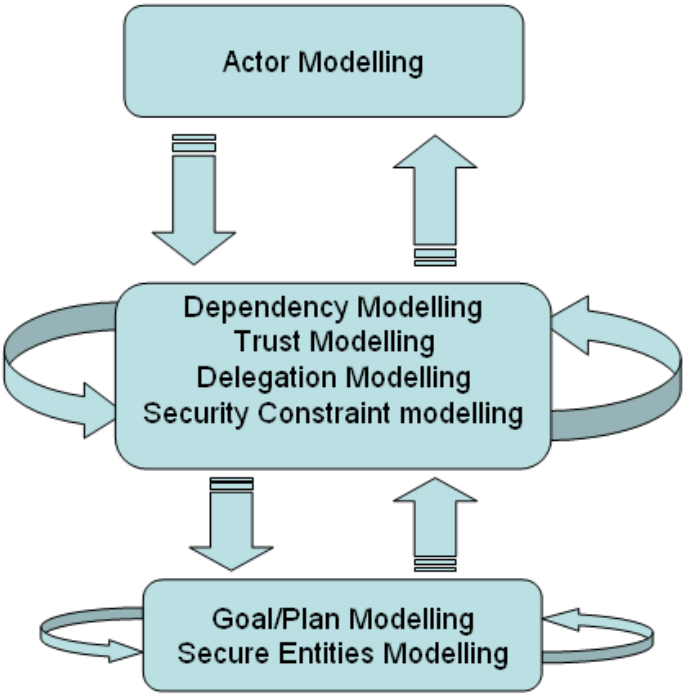

Abbildung 3.7: Typische Vorgehensweise bei der Modelliereung

Kapitel 4

Analyse der HiiMap Architektur

Zur besseren Veranschaulichung werden nicht sämtliche Zwischenmodelle aller vier Phasen vorgestellt, sondern nur auf die wesentlichen Modelle in ihrer jeweiligen Endfassung näher eingegangen.
Für die Analyse wird von drei Aktoren ausgegangen, die als „User“, „End Node“ und „HiiMap“ bezeichnet werden. Der Aktor „HiiMap“ steht dabei für das zu untersuchende System. Im Folgenden werden zunächst zwei Aktor-Diagramme dargestellt, die einmal die nötigen Strukturen ohne den Aktor „HiiMap“ zeigen und einmal mit dem Aktor „HiiMap“ und damit die Einbettung des Systems in seine Umwelt. Anschließend werden für die drei Aktoren Ziel-Diagramme vorgestellt, wobei die einzelnen Ziele, die von den Aktoren erfüllt werden müssen, näher erläutert werden. Kapitel 4.3 zeigt dann exemplarisch einen Trustgraphen und Kapitel 4.4 befasst sich mit der Vergabe der Ziele des Aktors „HiiMap“ an die einzelnen Unteraktoren. In Kapitel 4.5 wird auf die gefundenen Schwachstellen des Systems näher eingegangen und Vorschläge zur Verbesserung dieser erläutert. Kapitel 4.6 fasst die gefundenen Ergebnisse schließlich noch einmal zusammen.

4.1 Security Enhanced Actor Diagram

4.1.1 Aktor-Diagramm ohne System

Das Aktor-Diagramm stellt einen ersten groben Überblick über die Ziele der beiden Aktoren „User“ und „End Node“ dar. Dabei wird weder auf die konkrete Umsetzung der Ziele eingegangen, noch werden potenzielle Sicherheitsrisiken aufgezeigt.

Der Aktor „User“ repräsentiert dabei einen Benutzer, der Zugang zum Internet bekommen will. Dazu muss er vorab die Ressource „Smart Card“ erhalten. Für die weitere Registrierung der „Smart Card“ und der damit verbundenen Aufnahme ins System muss er zusätzlich die „Mapping Directive“ erhalten. Darüber hinaus ist es unumgänglich, dass der

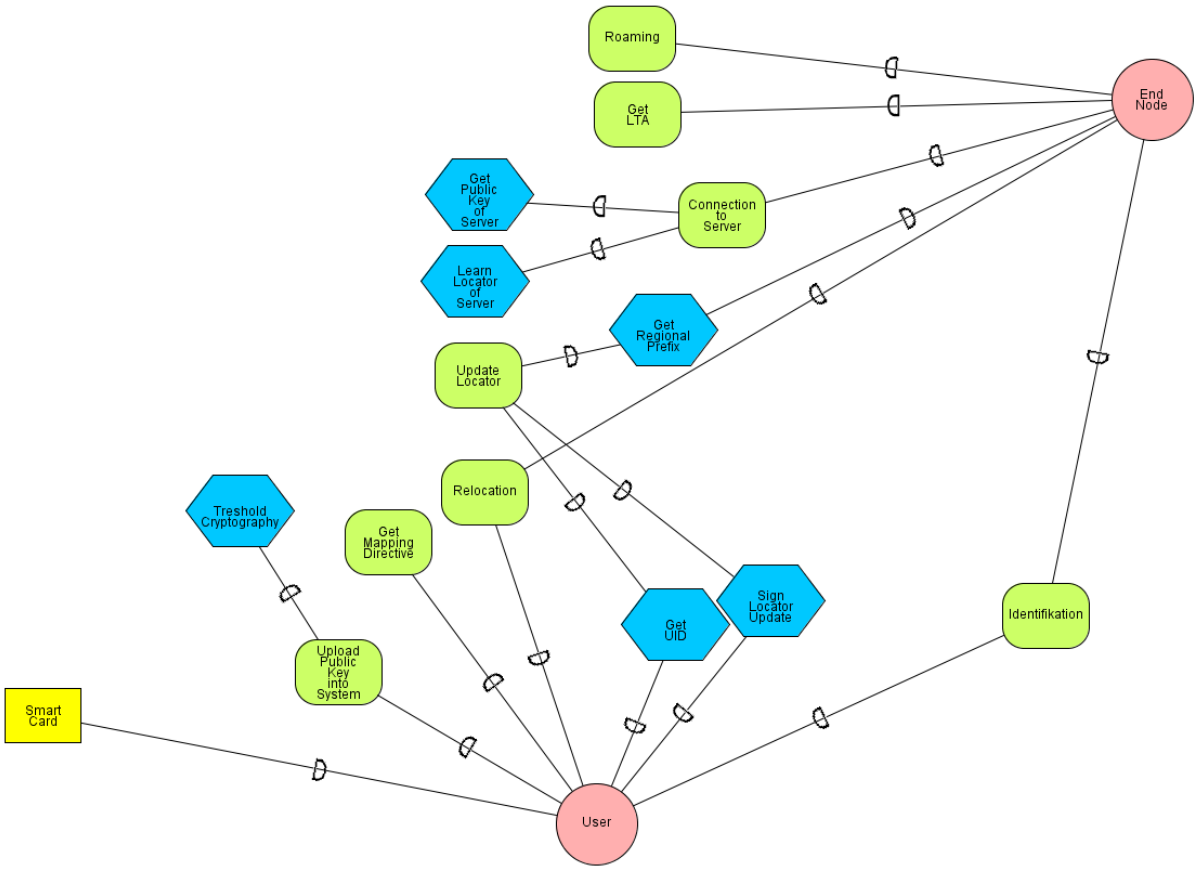

Abbildung 4.1: Aktor-Diagramm ohne System

„Public Key“, der sich auf der „Smart Card“ befindet und laut HiiMap auch in das System geladen werden muss, durch die in Kapitel 2.3 beschriebene „Threshold Kryptographie“ in mehrere Stücke zerteilt wird.

Das Hauptziel des Aktor „End Node“, der einen beliebigen Rechner innerhalb des Netzwerks beschreibt, ist es, eine Verbindung zu einem Server herzustellen. Dazu muss, er sich selbst bei HiiMap anmelden und erhält dadurch die LTA. Nun ist es ihm möglich, eine Verbindung aufzubauen, wenn er zum Einen den RP und zum Anderen den aktuellen Locator des Servers lernt. Als weitere mögliche Ziele werden in diesem Modell auch noch die Mobilitätsmöglichkeiten, „Roaming“ und „Relocation“ aufgezeigt. Die Befriedigung des Zieles „Identification“ hängt - wie man in dem Modell sieht - alleine vom Aktor „User“ ab und bezeichnet die Anmeldung des „Users“ am „End Node“.

4.1.2 Aktor-Diagramm mit System

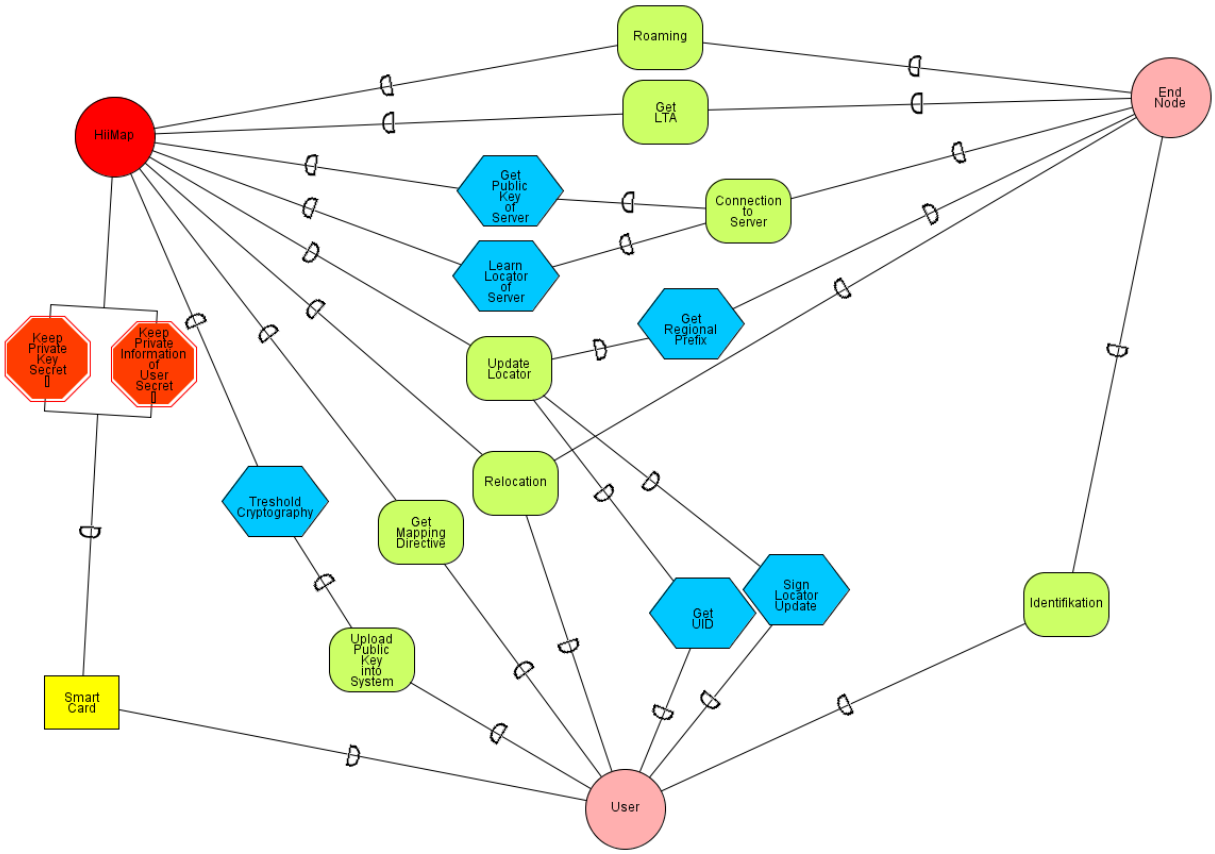

Abbildung 4.2: Aktor-Diagramm mit System

In einem nächsten Schritt wird das System HiiMap Architektur als eigenständiger Aktor in das Modell mit aufgenommen. Dabei sieht man sehr schön wie alle Ziele, die bisher noch nicht von einem Aktor befriedigt werden konnten, nun durch das System erfüllt werden. Auch werden erste Sicherheitsbedingungen aufgezeigt wie die Geheimhaltung der persönlichen Daten des „Users“ oder des Private Keys bei der Übergabe der Smart Card vom „System“ an den „User“.

4.2 Security Enhanced Goal Diagram

Ziel-Diagramme sind das Ergebnis der Analyse der Ziele eines Aktor ausgehend vom Blickpunkt dieses Aktors mit Hilfe von „means end“ und AND-Verbindungen. Die graphische Repräsentation dieser Ziele wird in Blasen dargestellt, in denen Abhängigkeiten mit anderen Aktoren aufgebaut und Ziele und Pläne eines spezifischen Aktors analysiert werden.

4.2.1 Ziel-Diagramm des „End Node“

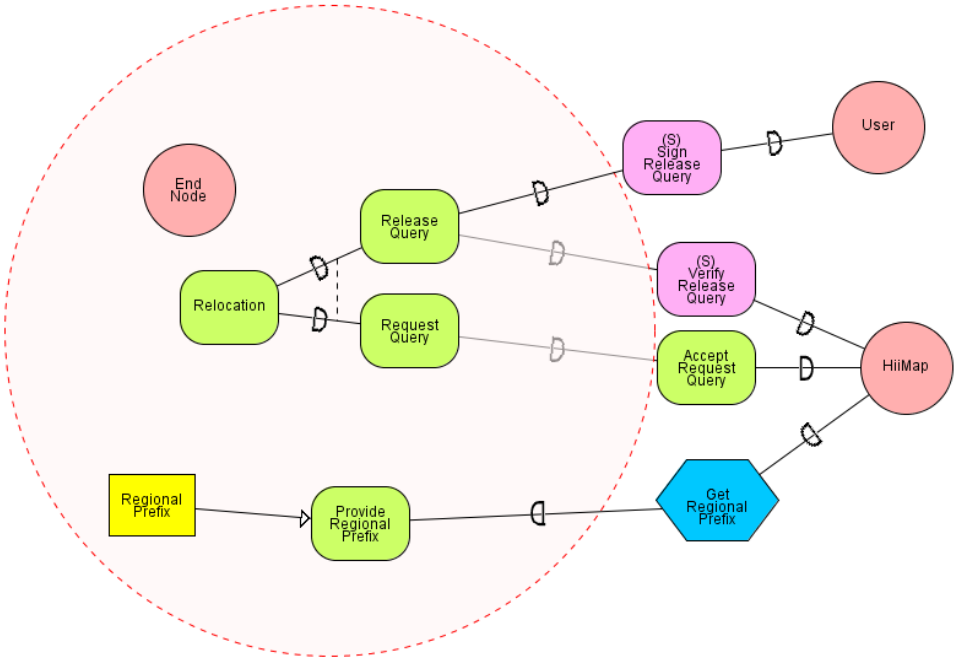

Abbildung 4.3: Ziel-Diagramm End Node

Im Ziel-Diagramm des Aktors „End Node“ wird deutlich, dass um das Ziel „Relocation“ zu erreichen sowohl der Aktor „User“ als auch das System nötig sind. Dabei wird nun das Hauptziel „Relocation“ in die beiden Unterziele „Release Query“ und „Request Query“ unterteilt. Die Release Anfrage muss zuerst vom „User“ mit Hilfe des „Private Keys“, der sich auf der „Smart Card“ befindet unterschrieben werden und daraufhin vom System verifiziert werden. Gleichzeitig wird eine Request Anfrage ans System geschickt, die dort akzeptiert werden muss.
Weiter zeigt das Ziel-Diagramm des „End Node“, wie der Plan „Get Regional Prefix“ des Systems vom Aktor „End Node“ umgesetzt wird. Hierbei wird das Ziel „Provide Regional Prefix“ mit Hilfe der Ressource RP erreicht, die der Aktor besitzt.

4.2.2 Ziel-Diagramm des „User“

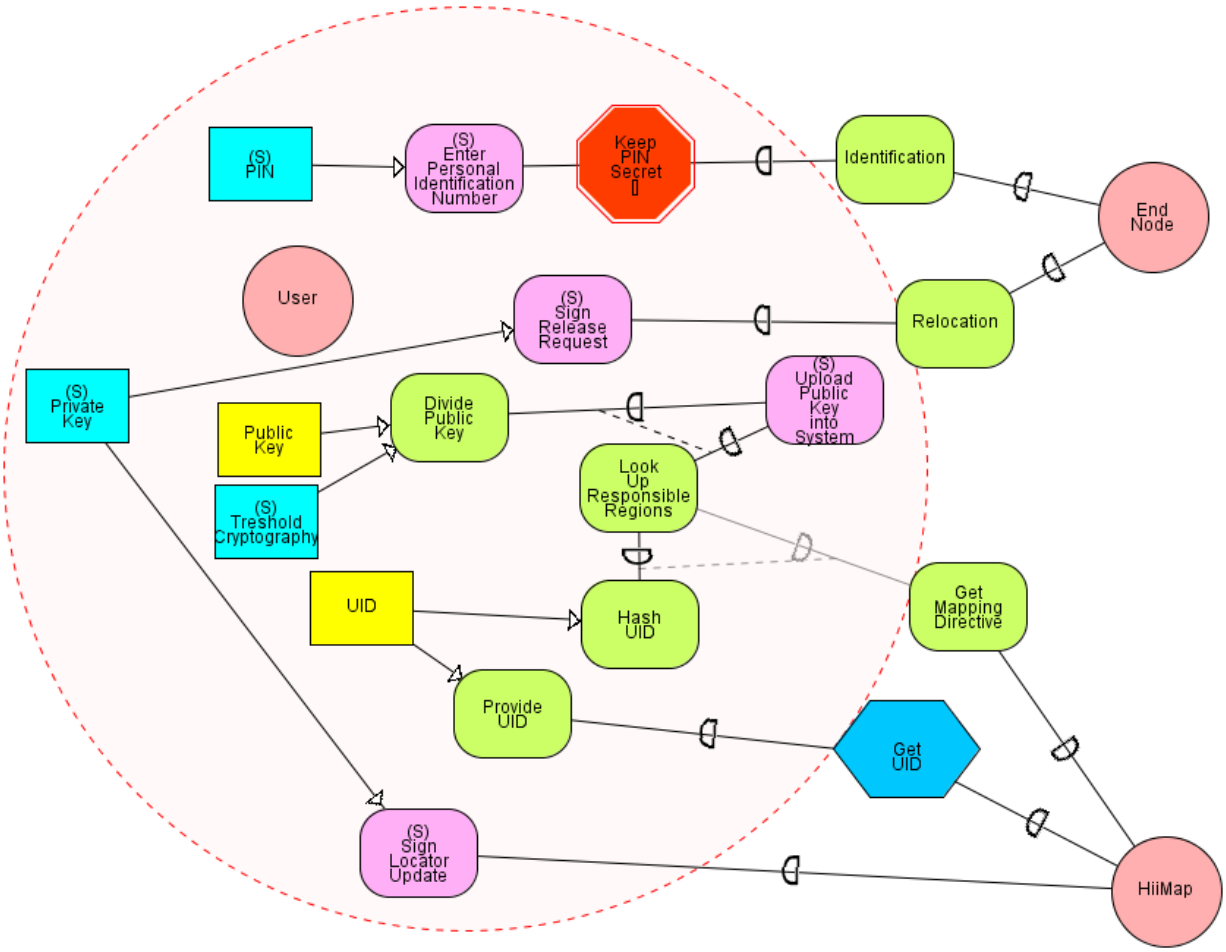

Abbildung 4.4: Ziel-Diagramm User

Das Ziel-Diagramm des „Users“ befasst sich zu einem großen Teil mit der Verarbeitung der Daten auf der Smart Card. Um nun nach Erhalt der Smart Card den darauf befindlichen Public Key ins System bzw. an die einzelnen Regionen zu laden muss zum Einen der Public Key zerteilt werden, was mit Hilfe der Threshold Kryptographie geschieht. Zum Anderen muss der „User“ wissen, an welche Regionen er seine Schlüsselstücke schicken soll. Dies wird erreicht durch Bildung eines Hashwertes aus der UID und dem Vergleich dieses Wertes mit der „Mapping Directive“, die er vom System bereitgestellt bekommt. Mit diesen Informationen können die Regionen ermittelt werden, an die einzelne Schlüsselstücke geschickt werden müssen.
Mit Hilfe der Ressource „Private Key“ werden Signaturen aller Art durchgeführt und somit auch das Ziel „Update Locator“ des Systems sowie „Sign Release Query“, das für die „Relocations“ des „End Node“ nötig ist, befriedigt.

Eine weitere Aufgabe des „Users“, die in diesem Modell aufgezeigt wird, ist die Identifikation und Authorisierung beim Aktor „End Node“. Dies passiert durch Eingabe einer Personal Identification Number (PIN), wobei natürlich darauf zu achten ist, dass diese Nummer geheimgehalten wird und vor dem Zugriff Dritter geschützt werden muss.

4.2.3 Ziel-Diagramm des Systems „HiiMap“

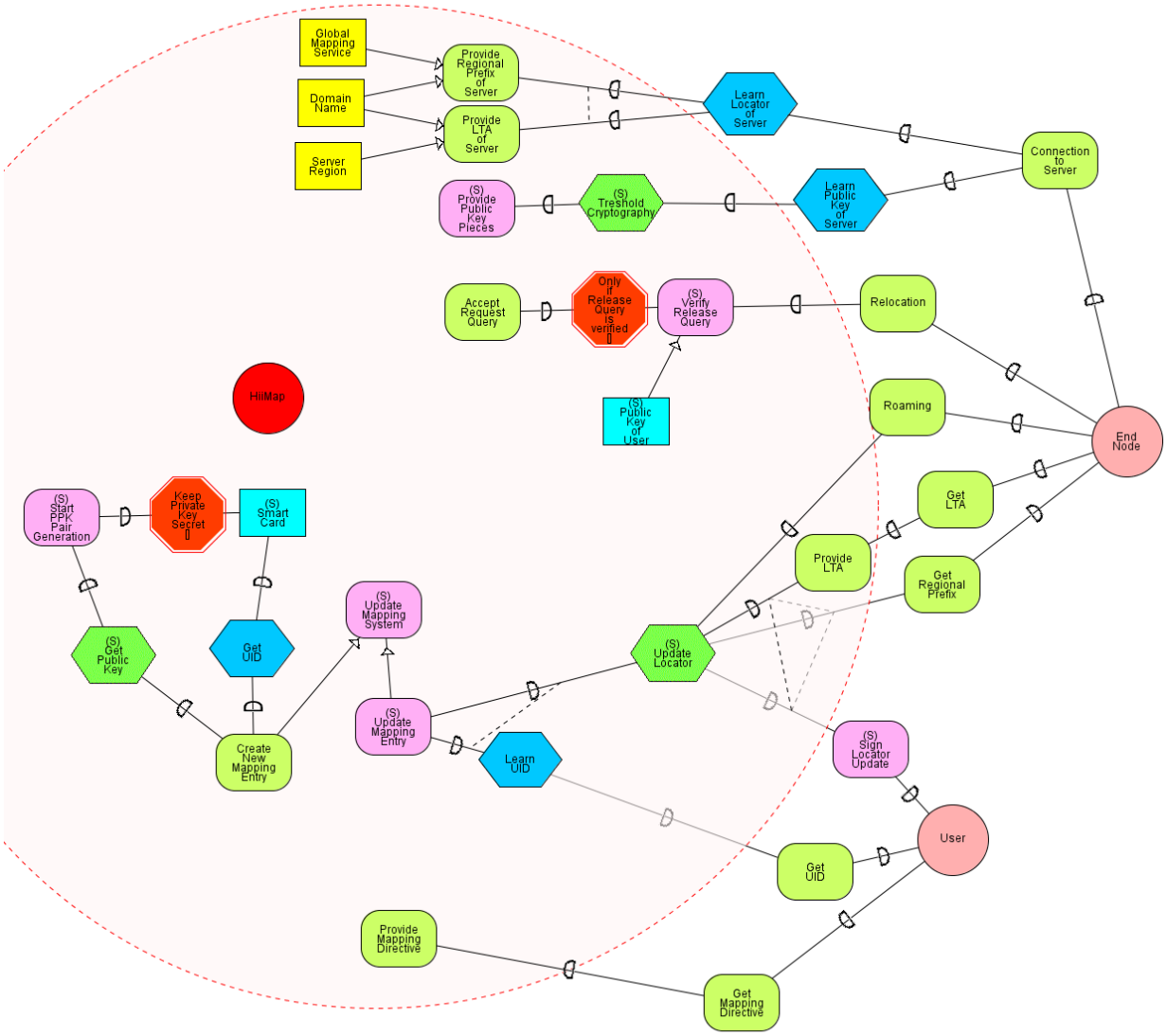

Abbildung 4.5: Ziel-Diagramm HiiMap

Im Ziel-Diagramm des Aktors „HiiMap“, der die gesamte HiiMap Architektur beschreibt,

werden weitere wichtige Aufgaben aufgezeigt. Bei der Erstellung eines ersten Mapping Eintrags wird auf der Smart Card, die zu diesem Zeitpunkt noch im Besitz des „Systems“ ist, zum Einen die UID ausgelesen und zum Anderen die Generierung eines Private-/Public Key Paares angeregt. Dabei ist natürlich besonders darauf zu achten, dass der dabei generierte Private Key zu keinem Zeitpunkt ausgelesen werden kann. Mit Hilfe des Public Keys und der UID wird nun ein erster Mappingeintrag angelegt. Erst jetzt kann die Smart Card an den User verschickt werden.

Ein zweites zentrales Ziel der HiiMap Architektur ist es, die Mappingeinträge ständig zu aktualisieren. Dazu muss das System natürlich die UID des Users lernen, dessen Eintrag aktualisiert werden soll, was durch den Plan „Learn UID“ dargestellt wird und vom Aktor „User“ erfüllt wird. Darüberhinaus muss der Plan „Update Locator“ erfüllt werden, um den gerade aktuellen Locator im Mappingeintrag abspeichern zu können. Dies wird durch die Ziele „Get RP“ des Aktors „End Node“ und „Provide LTA“ des Systems erreicht. Im Anschluss daran wird das Locator Update noch vom User signiert.

Die gerade beschriebenen Ziele „Create New Mapping Entry“ und „Update Mapping Entry“ sind beide Mittel zum Zweck um das Ziel „Update Mapping System“ zu erfüllen.

Als weiteres wird deutlich, wie das Ziel „Connection to Server“ des Aktors „End Node“ durch die beiden Pläne „Learn Public Key of Server“ sowie „Learn Locator of Server“ befriedigt wird. Um den aktuellen Locator zu erfahren muss sowohl das RP sowie die LTA des Servers bekannt werden. Dazu kann für das RP auf den globalen Mapping-Dienst des Systems zugegriffen werden und die LTA durch Kontakieren der Region des Servers erfahren werden. Ansonsten besteht natürlich auch die Möglichkeit, dass der gesamte Locator oder nur ein Teil davon durch den Domain Name bekannt sind. Der Public Key des zu erreichenden Servers wird einfach durch Anfragen bei den Regionen erhalten, bei denen die einzelnen Schlüsselstücke abgelegt wurden. Durch Anwendung der Threshold Krytpgraphie kann dann der gesamte Schlüssel rekonstruiert werden.

Außerdem wird im Diagramm das Ziel „Relocation“ des „End Node“ genauer beleuchtet. Dazu wird „Release Query“ mit Hilfe des Public Keys verifiziert. Im Anschluss daran kann nun - nach erfolgreicher Verifizierung - der „Request Query“ akzeptiert werden und somit die „Relocation“ des „End Node“ abgeschlossen werden.

4.3 Trust-Diagramm

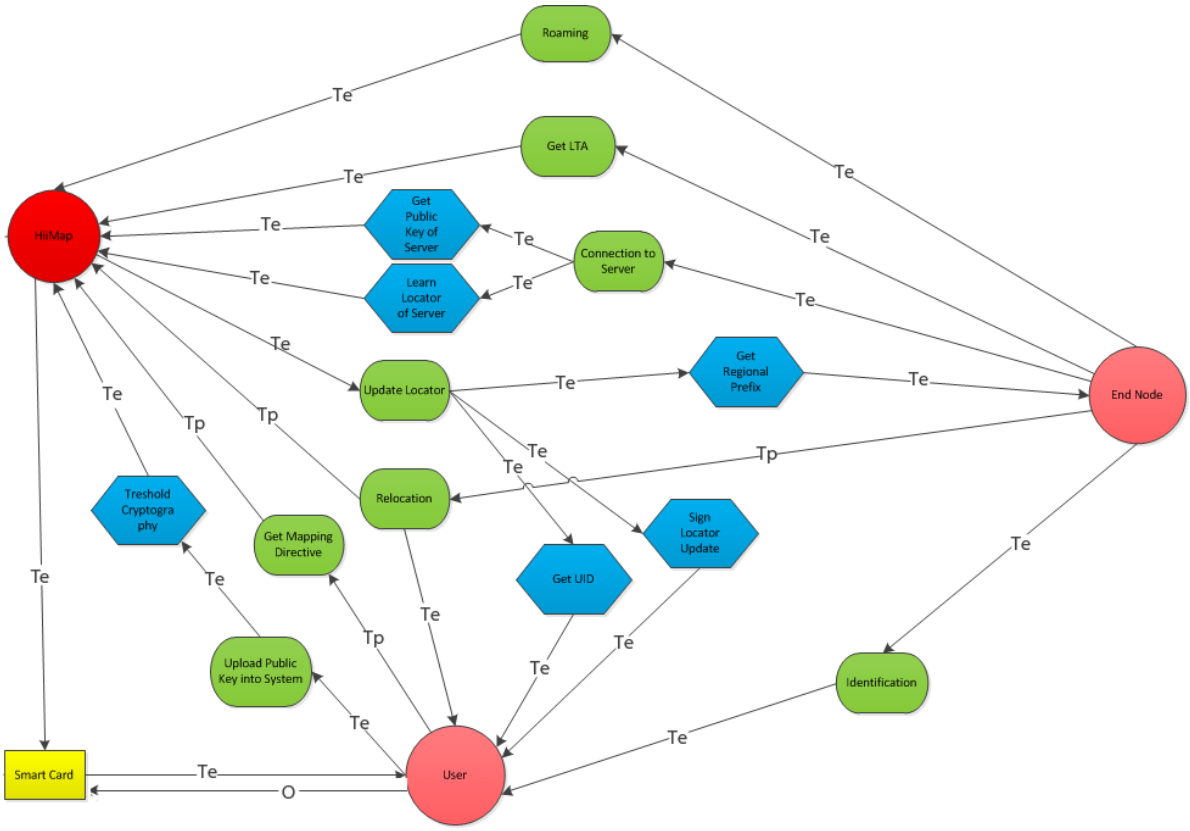

Abbildung 4.6: Trust Diagramm

Das hier gezeigt Trust-Diagramm steht exemplarisch für eines der Modelle, die bei der Erstellung der Aktor-Diagramme entwickelt wurden.
Hierbei werden sowohl die Trust-Beziehungen zwischen den drei Aktoren deutlich als auch die Besitzverhältnisse der Ressource „Smart Card“, die von der GA hergestellt und dann an den „User“ verschickt wird. Ab diesen Zeitpunkt ist der Aktor „User“ Eigentümer der „Smart Card“ und hat somit auch volle Befugnisse über die auf der Karte gespeicherten Daten.

4.4 System Decomposition Diagramm

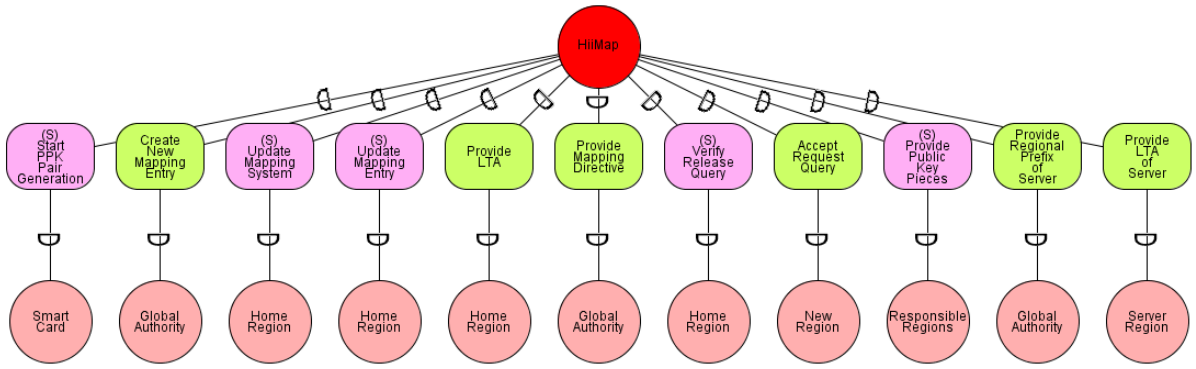

Abbildung 4.7: System Decomposition

Das System Decomposition Diagram zeigt alle Ziele des Aktors „HiiMap“ auf, die am Anfang einer Dependency-Kette stehen und somit bisher noch nicht direkt befriedigt wurden. Diese werden hier einem Aktor innerhalb des Systems selbst zugeordnet. Also wird z.B. das Ziel „Create New Mapping Entry“ dem Aktor „Global Authority“ zugeordnet oder das Ziel „Provide LTA“ dem Aktor „Heimregion“. Hierbei wird auch deutlich, dass die beiden Unteraktoren „Global Authority und „Heimregion“ am meisten Ziele zu verwalten haben und somit auch eine große Verantwortung für das Funktionieren der HiiMap Architektur haben oder anders gesagt diejenigen Aktoren sind, bei denen ein starkes Trustverhältniss unabdingbar ist.

4.5 Kritische Komponenten der HiiMap Architektur

Abbildung 4.8 zeigt noch einmal den gesamten Überblick über die Aktoren und deren Ziele.

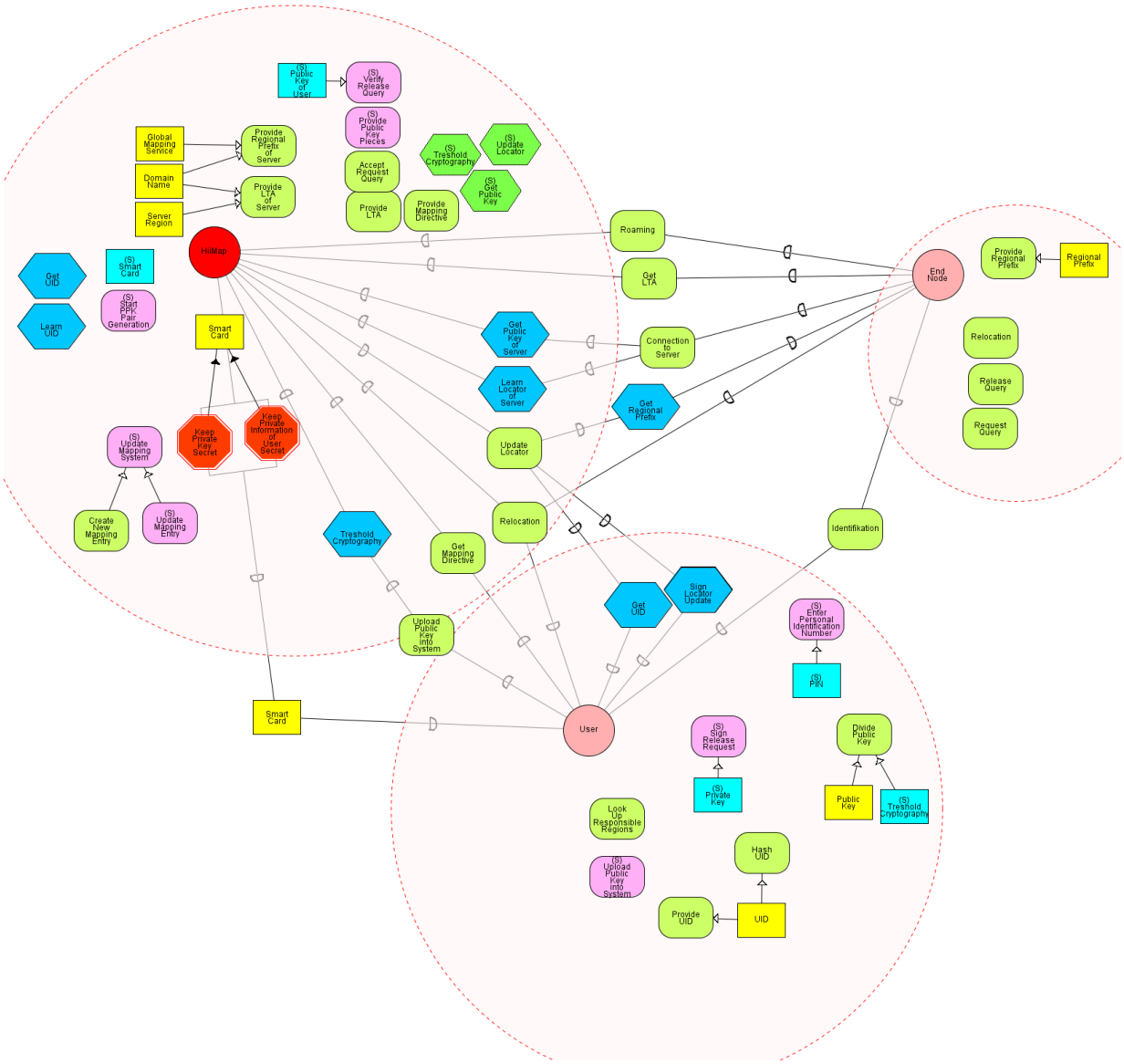

Abbildung 4.8: Systemüberblick

Im Folgenden nun eine kurze Diskussion über kritische Komponenten der HiiMap Architektur und Vorschläge zur Verbesserung Dieser.

- Smart Card:
  Die Smart Card ist eine der wichtigsten Ressourcen in der HiiMap Architektur. Sie beinhaltet die UID sowie den Private und den Public Key des Users. Demzufolge muss die Smart Card auch besonders geschützt werden und vor allem muss sichergestellt werden, dass der Private Key nicht einfach ausgelesen werden kann und auf keinen Fall die Smart Card jemals verlässt, ansonsten könnte sich ein Angreifer ohne Probleme als jemand Anderer ausgeben.
  Die Smart Card wird - wie man in den Abbildungen 4.5 und 4.7 sieht - von der GA oder einem ihr untergeordneten Unternehmen hergestellt. Ein wichtiger Punkt ist die Generierung des Private- und Public Key Paares; die Global Autority liest danach den Public Key sowie die UID der Smart Card aus und generiert damit gleich noch den ersten Eintrag im Mapping-System der für den User zuständigen Region. Auf diese Weise wird sichergestellt, dass zwischen HiiMap und User auch schon beim ersten Einloggen des Users alle nötigen Daten oder Pakete verschlüsselt oder signiert übertragen werden können, da der Public Key dann ja schon in der Region gespeichert ist. Beim anschließenden Versenden der Smart Card sollte auf jeden Fall sichergestellt werden, dass diese auch wirklich beim richtigen User landet und nicht z.B. durch Abfangen der Post ein potentieller Betrüger/Angreifer in Besitz einer Smart Card kommt, die nicht für ihn gedacht war. Dies kann durch eine Überprüfung der Identität des Users beim Übergeben der Smar Card erfolgen, z.B. durch das Postidentverfahren. Darüberhinaus sollte man sich überlegen, wie und ob der Zusammenhang zwischen UID der Smart Card und der tatsächlichen Identität des Users gespeichert wird und auch entsprechend gesichert wird. Zu guter Letzt muss man sich natürlich auch überlegen, was bei Verlust oder Diebstahl der Smart Card passiert. Ein einfacher Weg wäre, dass der User ähnlich wie bei Verlust der EC Karte eine Telefonnummer anrufen kann und mittels Seriennummer der Karte diese bei der GA sperren lässt.

- Public Key:
  Der Public Key sollte prinzipiell natürlich für Jeden verfügbar sein, der Kontakt mit dem User aufnehmen will oder eine Signatur überprüfen soll. Allerdings muss der Public Key an einer Stelle gespeichert sein, der jeder vertrauen kann.
  Bei der Herstellung einer Smart Card wird der Public Key zusammen mit der UID für den ersten Eintrag im Mapping-System bei der zuständigen Region hinterlegt (siehe Abbildung 4.5) und diese könnte dadurch auch Jedem der den Key anfordert diesen zur Vergügung stellen. Zu Bedenken ist, dass die Verwaltungn der Regionen im HiiMap Organisationen unterliegt, die das ihnen erbrachte Vertrauen aber auch missbrauchen und falsche Keys herausgeben oder die Bereitstellung des Public Keys ganz verweigern könnten. Darum hat man sich entschlossen den Public Key des Users nicht nur zentral bei einer Region zu hinterlegen, sonder diesen zu zerteilen und die Stückchen bei mehren Regionen abzulegen.

Hat der User seine Smart Card erhalten muss er deshalb den darauf gespeicherten Public Key in n Stücke teilen (siehe Abbildung 4.4), wobei er n selbst bestimmen kann. Für die Zerteilung des Public Keys greift man auf die Treshold Kryptographie zurück. Danach werden mit der von der GA bereitgestellten Mapping Directive (siehe Abbildung 4.5) und dem Hashwert der UID des Users die Regionen ermittelt an welche die einzelnen Schlüsselstücke verteilt werden. Um sicher zu stellen, dass die Liste auch wirklich von der GA kommt und nicht irgendwie manipuliert wurde sollte diese auf jeden Fall von dieser signiert werden. Ein großer Vorteil der Treshold Kryptographie ist, dass nicht unbedingt alle n Stückchen zur Rekonstruktion des Public Keys gebraucht werden. Da man schon mit $k < n$ Stückchen den Key rekonstruieren kann ist es einer einzelnen Region nicht mehr möglich die Rekonstruktion des ganzen Keys zu sabotieren. Darüber hinaus lässt sich sogar nachvollziehen, welche Region die Bereitstellung verweigert oder ein manipuliertes Stückchen verschickt hat. Dadurch könnte man im Falle wiederholten Missbrauchs Sanktionen gegen eine Region erheben oder dieser zumindest keine Keystückchen mehr zur Aufbewahrung anvertrauen. Sind die Keystückchen dann bei den zuständigen Regionen hinterlegt kann Jeder, der die UID des Users kennt dessen UID hashen und mit Hilfe der Mapping Directive mindestens k Regionen kontaktieren und so den Public Key des Users zusammensetzen.

- Identifikation:
  Damit ein User Zugang zum Internet bekommt muss er sich mit seiner Smart Card bei einem End Node einloggen.
  Dabei ist es wichtig, dass die Identität des Users sichergestellt werden kann um Missbrauch der Karte zu verhindern. Um das zu erreichen wird der User beim Einloggen mit der Smart Card zusätzlich zur Eingabe einer PIN Nummer gezwungen (siehe Abbildung 4.4). Im Grunde gibt es zwei Möglichkeiten, die PIN zu erhalten/generieren:

  – Die PIN wird dem User neben der Smart Card direkt von der GA zugeschickt. Hierbei sollte natürlich auch sichergestellt werden, dass die PIN nur dem User selbst zugänglich ist. Natürlich sollte die PIN auch nicht gemeinsam mit der Karte verschickt werden. (ähnlich wie bei EC Karte)

  – Die zweite Möglichkeit ist, dass der User nach Erhalt der Smart Card sich selbst eine PIN generiert und diese dann beim ersten Benutzen der Karte auf dieser für künftige Authentifizierungen hinterlegt wird. Auf diese Weise wird zwar sichergestellt, dass wirklich nur der User die PIN kennt, allerdings überträgt man diesem damit auch mehr Verantwortung: Wenn er seine PIN vergisst gibt es keine Möglichkeit mehr, die Karte weiter zu benutzen.

  In jedem Fall muss sichergestellt werden, dass die PIN ähnlich wie der Private Key nicht ausgelesen werden kann.

- Relocation:
  Wie man im Ziel-Diagramm des End Node sieht wird für die Relocation ein Release Antrag gestellt, der erst vom Aktor „User“ und speziell mit dem Private Key auf der Smart Card signiert wird, bevor er an „HiiMap“ übermittellt wird. Zusammen mit dem Release Antrag wird auch noch ein Request Antrag gestellt, der direkt zum HiiMap gesendet wird. Die Zusage für eine Relocation kann aber erst gegeben werden, wenn die Heimregion den Umzug akzeptiert. Dies kann zu einem Problem werden, wenn ein Unternehmen eine Relocation wünscht, die Region diese aber verweigert. Solange die Heimregion also nicht der Relocationanfrage zustimmt, hat das Unternehmen keine Möglichkeit einen Standortwechsel durchzuführen. Dies würde Regierungen die Möglichkeit geben einen Wegzug von Unternehmen oder Privatpersonen zu verhindern.
  Man kann dieses Problem einfach lösen, indem man das System so abändert, dass der Relocation Antrag der Heimregion nur zugestellt aber von ihr nicht unbedingt akzeptiert werden muss. Die Verarbeitung des Relocationantrags müsste dann von der Zielregion vorgenommen werden.

4.6 Ergebnis

Zusammenfassend lässt sich festhalten, dass die HiiMap Architektur ein grundsätzlich sicheres System darstellt, das durchaus so implementiert werden könnte.
Die in Kapitel 4.5 aufgezeigten Schwachstellen wie die noch zu klärende PIN Abfrage oder die Verarbeitung des Relocationantrages durch die Heimregion sind alle durch geringe Modifikationen eliminierbar.
Die Global Authority und die Regionen stellen nach wie vor die beiden wichtigsten Trustpunkte der Architektur dar und deshalb sollte bei der Implementierung von HiiMap großer Wert darauf gelegt werden, dass diese absolut ausfallsicher gestaltet werden und keinesfalls von einem Angreifer übernommen werden können. Nach Meinung des Autors stellt insbesondere der Schutz der Regionen ein sehr großes Problem dar, da in HiiMap ja vorgeschlagen wird, dass eine Region einem Land gleichgesetzt werden sollte und die Verwaltung somit einer Regierung unterliegt.
Hier stellt sich ganz allgemein die Frage nach der Vertrauenswürdigkeit eines Staates. Wenn dieser seine Macht missbraucht, könnte er mit großer Exaktheit herausfinden, wo sich eine einzelne Person gerade befindet. Darüber hinaus wäre es ihm problemlos möglich seinen Bürgern den Zugang zum Internet einzuschränken oder gar ganz zu verbieten.

Kapitel 5

Zusammenfassung

Der gerade stattfindende Wandel des Internets und die Überlastung der heutigen Protokolle hat eine Forschungsgruppe der Technischen Universität München dazu bewegt eine neue Internetarchitektur zu entwickeln, die mit diesen Herausforderungen umzugehen weiß und auch für die Zukunft gerüstet ist.
Die HiiMap Architektur setzt auf eine Aufteilung der derzeitigen IP-Adressen in Locator und Identifier. Dadurch werden Mobilitätsbedürfnisse wie Roaming oder Relocation einfacher und ohne Mehraufwand in den Mapping-Systemen durchgeführbar.
Desweiteren wird das Internet in sogenannte Regionen unterteilt, die für das Mapping der Nodes verantwortlich sind und am besten einzelnen Ländern gleichgestellt werden.
Jedem Benutzer wird durch eine Smart Card eine eindeutige Identifikationsnummer (UID) und ein Private- und Public-Key Paar bereitgestellt. Darüber hinaus wird versucht möglichst viele Sicherheitsmaßnahmen - wie z.B. eine Public-Key Infrastruktur - fest in der Netzwerkschicht zu verankern, so dass keine zusätzlichen Modifikationen mehr benötigt werden.

Da heutzutage immer mehr Wert auf Sicherheit und Trust gelegt wird, wurde in dieser Arbeit versucht eine Trust Analyse der HiiMap Architekur durchzuführen. Nach Vergleich mehrerer Modellierungssprachen hat man sich für Secure Tropos entschieden, einer Weiterentwicklung der Tropos Methodologie. Diese Sprache adoptiert viele Elemente aus dem I*Framework und versucht ein System in Aktoren zu unterteilen und deren Ziele miteinander in Relation zu setzen. Vor allem wurde Secure Tropos im Gegensatz zu anderen Modellierungssprachen speziell für die Analyse von Systemen konzipiert, die Trust bereits in einer frühen Stufe der Entwicklung als einen wichtigen Aspekt berücksichtigen.
Secure Tropos unterteilt die Modellierung eines Systems in vier verschiedene Phasen, die zu einem immer höheren Detailierungsgrad der einzelnen Abläufe innerhalb eines Systems führen. Des Weiteren gibt es eine große Menge an verschiedenen Modellierungsmethoden, die es dem Designer erlauben Ziele von Aktoren aus verschiedenen Sichtweisen zu beleuchten und unter Berücksichtigung von Security Constraints alternative Wege zu

entwickeln, wie ein Ziel befriedigt oder ein Plan ausgeführt werden kann.

Die Modellierung und die darauf aufbauende Analyse der HiiMap Architektur hat ergeben, dass HiiMap ein System ist, das - zumindest im derzeitigen Entwicklungsstand - in Bezug auf Trust und Sicherheit keine größeren Schwachstellen aufweist.
Problematisch könnte jedoch eine Relocation werden, also wenn ein Benutzer dauerhaft seine Heimregion ändern will. Ohne die Zustimmung der Heimregion kann so ein Umzug derzeit nicht durchgeführt werden, was dazu führen könnte, dass Länder möglicherweise Unternehmen verbieten ihren Firmenstandort anderswohin zu verlegen. Zur Behebung des Problems wird vorgeschlagen, die Abwicklung der Relocationanfrage der Zielregion zu übertragen und die Heimregion nur noch über die bereits bewilligte Relocation in Kenntnis zu setzen.
Als weitere Schwachstelle wird die bisher noch nicht detailliert beschriebene Versendung der Smart Card aufgeführt. Es wird vorgeschlagen, dass ein Benutzer bei Erhalt seiner Smart Card zweifelsfrei identifiziert werden muss. Zudem sollte eine Authentifizierung bei Verwendung der Smart Card z.B. durch Eingabe einer PIN erfolgen, für die noch geklärt werden müsste, wie der Benutzer diese erhält.
Als größtes Problem wird die eventuell zu große Macht der Regionen gesehen. Hierbei ist zu bedenken, dass im HiiMap vorgeschlagen wird, die Regionen mit Ländern gleichzusetzen und deren Verwaltung durch eine Regierungsbehörde zu regeln. Man sollte bedenken, dass auch Regierungen das ihnen entgegengebrachte Vertrauen missbrauchen können, um die Benutzer und damit ihre Bürger zu überwachen, ihnen den Zugang zum Internet beschränken oder ganz untersagen.
Dieser letzte Punkt ist nach Einschätzung des Autors einer der problematischten Aspekte der HiiMap Architektur und zugleich auch ein Problem, dass sich auf Grund des Aufbaus von HiiMap kaum oder nur sehr schwierig lösen lässt.

Im Falle einer Implementierung der HiiMap Architektur müsste eine erneute Trustanalyse durchgeführt werden, die ihren Schwerpunkt auf die Implementierungsabläufe legt. Für diese Analyse wird dann eine Modellierungssprache wie sie in dem Paper „A Language for Modelling Trust“ [BMP10] beschrieben wurde empfohlen.
Zusammenfassend lässt sich sagen, dass HiiMap viele interessante Ansätze bietet, wie die Probleme des heutigen Internets behoben werden könnten. Es dient auch als Vorbild für anderen Entwicklungen, da es demonstriert wie man unter Berücksichtigung von Trust und Sicherheit in einer frühen Entwicklungsphase ein gut geschütztes System entwickeln kann.

Abbildungsverzeichnis

Abkürzungsverzeichnis

Literaturverzeichnis

[BMP07] K.K. Bimrah, H. Mouratidis, and D. Preston. Trust Ontology for Information Systems Development. *16th International Conference on Information Systems Development*, 2007.

[BMP10] K.K. Bimrah, H. Mouratidis, and D. Preston. A Language for Modelling Trust in Informational Systems. *Information Systems Development: Towards a Service Provision Society*, 2010.

[CW02] K. Chopra and W.A. Wallace. Trust in Electronic Enviroment. *36th Hawaii International Conference on System Sciences*, 2002.

[GMZ08] P. Giorgini, H. Mouratidis, and N. Zannone. Modelling Security and Trust with Secure Tropos. *CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States)*, 2008.

[HKS+09] O. Hanka, G. Kunzmann, C. Spleiß, J. Eberspächer, and A. Bauer. HiiMap: Hierarchical Internet Mapping Architecture. *First International Conference on Future Information Networks*, 2009.

[HKSE09] O. Hanka, G. Kunzmann, C. Spleiß, and J. Eberspächer. A novel DHT-based network architecture for the Next Generation Internet. *Eighth International Conference on Networks*, 2009.

[IB07] L. Iannone and O. Bonaventure. On the Cost of Caching Locator/ID Mappings. *2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, 2007.

[KGSS04] S. Kethers, G. Gans, D. Schmitz, and D. Sier. Modelling Trust Relationships in a Healthcare Network: Experiences with the TCD Framework. *13th European Conference on Information Systems (ECIS)*, 2004.

[MG09] H. Mouratidis and P. Giorgini. Enhancing Secure Tropos to Effectively Deal with Security Requirements in the Developing of Multiagent Systems. *Safety and Security in Multiagent Systems*, pages 8–26, 2009.

[MGM03] H. Mouratidis, P. Giorgini, and G. Manson. An Ontology for Modelling Security: The Tropos Approach. *CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States)*, 2003.

[MGM05] H. Mouratidis, P. Giorgini, and G. Manson. When security meets software engineering: A case of modelling secure information systems. *Information Systems Volume 30 Issue 8*, 2005.

[QIDLB07] B. Quotin, L. Iannone, C. De Launois, and O. Bonaventure. Evaluating the Benefits of the Locator/Identifier Separation. *ACM CoNEXT conference*, 2007.

[WASE05] M. Wilson, A. Arenas, L Schubert, and Ed. The Trustcom Framework V0.5. *CCLRC ePublication Archive [http://epubs.cclrc.ac.uk/oai] (United Kingdom)*, 2005.

[YL01] E. Yu and L. Liu. Modelling Trust for System Design Using i* Strategic Actors Framework. *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference: Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, pages 175–194, 2001.