Universität Stuttgart
Fakultät Informatik

Studiengang:	Informatik
Prüfer:	Prof. Dr. Kurt Rothermel
Betreuer:	Dipl. Inf. Arno Wacker
begonnen am:	24.06.2003
beendet am:	23.12.2003
CR-Klassifikation:	C.2.4, C.4, C.5.3, D.4.6, K.6.5

Diplomarbeit No. 2118

Design, prototypische Implementierung und Evaluierung einer Sicherheitsschicht für ressourcenbeschränkte Geräte

Holger Cermann

Institut für Parallele und
Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Zusammenfassung

In dieser Diplomarbeit wird eine Sicherheitsschicht vorgestellt. Diese ist speziell für den Einsatz in ressourcenbeschränkten Geräten, wie handelsüblichen Mikrocontrollern ausgelegt. Hierbei muss davon ausgegangen werden, dass es auf Grund von Speicherbeschränkungen nicht möglich ist, Schlüssel zu allen Geräten in einem Netzwerk aufzubauen. Ebenso können einzelne Geräte aus Kostengründen nicht so geschützt werden, dass es für einen Angreife unmöglich ist, einzelne Geräte und deren Informationen unter seine Kontrolle zu bekommen.

Inhaltsverzeichnis

Inhaltsverzeichnis **2**

Abbildungsverzeichnis **5**

Tabellenverzeichnis **6**

1 Einleitung **7**
1.1 Aufbau . . . 7
1.2 Motivation . . . 7
1.3 Szenarien . . . 8
1.4 Aufgabenstellung . . . 10
1.5 Anforderungen . . . 10

2 Systemüberblick **12**
2.1 Anforderungen an Kommunikationssemantiken . . . 12
2.2 Beschreibung der Hardwarearchitektur . . . 13
2.3 Beschreibung der Kommunikationsschicht . . . 13
2.4 Cryptoalgorithmen . . . 14

3 Analyse bestehender Ansätze **15**
3.1 Asymmetrische, dezentrale Ansätze . . . 16
3.1.1 Angriffsmöglichkeiten . . . 16
3.1.2 Kurzbeschreibung . . . 17
3.1.3 Netzaufbau . . . 18
3.1.4 Kommunikation . . . 18
3.1.5 Ausfallsicherheit . . . 18
3.2 Zentrale, symmetrische Ansätze . . . 18
3.2.1 Kurzbeschreibung . . . 19
3.2.2 Netzaufbau . . . 19
3.2.3 Kommunikation . . . 19
3.2.4 Ausfallsicherheit . . . 20
3.3 Symmetrische, dezentrale Ansätze mit Gruppenschlüssel . . . 21
3.3.1 Kurzbeschreibung . . . 21
3.3.2 Netzaufbau . . . 21
3.3.3 Kommunikation . . . 22
3.3.4 Ausfallsicherheit . . . 23

3.4 Verwendung einer zufälligen Schlüsselvorverteilung . . . 23
3.4.1 Kurzbeschreibung . . . 23
3.4.2 Netzaufbau . . . 23
3.4.3 Kommunikation . . . 25
3.4.4 Ausfallsicherheit . . . 25
3.5 Erweiterungen der zufälligen Schlüsselvorverteilung . . . 25
3.5.1 Kurzbeschreibung . . . 25
3.5.2 q-zusammengesetztes zufälliges Schlüsselvorverteilungsverfahren . . . 26
3.5.3 Schlüsselerneuerung über mehrere Pfade . . . 26
3.5.4 Zufälliges paarweises Verfahren . . . 27
3.6 Verwendung paarweiser Schlüssel . . . 28
3.6.1 Kurzbeschreibung . . . 28
3.6.2 Netzaufbau . . . 28
3.6.3 Kommunikation . . . 28
3.6.4 Ausfallsicherheit . . . 29
3.7 Zusammenfassung und Auswertung der Ansätze . . . 29

4 Verfahren zur Verteilung von Schlüsseln 31
4.1 Allgemeine Beschreibung . . . 31
4.2 Aufbau des Netzwerkes . . . 32
4.3 Verbindungserzeugung . . . 33

5 Schichtentwurf 35
5.1 Angebotene Dienste . . . 35
5.2 Benötigte Dienste . . . 36
5.3 Resultierender Schichtenaufbau . . . 36
5.3.1 Verschlüsselungsschicht (Frame Encryption and Verification) . . . 37
5.3.2 Imprinten (Imprinting) . . . 38
5.3.3 Schlüsselmanagement (Key Management) . . . 38
5.3.4 Sicherheitsprotokolle (Security Protocols) . . . 38
5.3.5 Kontrollschicht (Controlling) . . . 38
5.4 Paketentwurf . . . 39

6 Protokollentwurf 41
6.1 Benötigte Protokolle . . . 41
6.2 Imprinting . . . 43
6.3 Verwendung von Masterdevices . . . 46
6.3.1 Einbringen eines Masterdevices . . . 47
6.3.2 Imprinting mittels Masterdevice . . . 48
6.4 Linkerweiterung . . . 48
6.4.1 Zwei-Hop Protokolle . . . 49
6.4.2 Multi-hop Protokolle . . . 54
6.4.3 Pfadfindung mittels Delegation . . . 56
6.4.4 Auswahlkriterien für die Pfadwahl . . . 57
6.5 Schlüsselerneuerung . . . 58
6.6 Autonome Netzerhaltung des Systems . . . 58

6.6.1 Hinzufügen weiterer pseudoimprinteter Kanten 59
6.6.2 Abmelden existierender Geräte 60
6.6.3 Erkennung von Knotenausfällen 61
6.6.4 Sichere Heilung von Knotenausfällen 63
6.7 Geführte Netzerhaltung mittels Masterdevice 63
6.8 Gruppenkommunikation 63
6.9 Revocationmechanismen 64
6.9.1 Revocation mittels Masterdevice 65
6.9.2 Autonome Revocation 65

7 Prototypische Implementierung 68
7.1 Ziele der Implementierung 68
7.2 Entwicklung einer Emulationsumgebung 69
7.3 Erstellung einer Administrationssoftware 70
7.3.1 Visualisierung der im Netz befindlichen Geräte 70
7.3.2 Einbringen und Imprinten neuer Geräte 70
7.3.3 Visualisierung der Netztopologie 71
7.3.4 Sammeln von Statistikinformationen 72
7.4 Erstellung einer Plattform für automatisierte Tests 73
7.4.1 Erstellung von Testcases 73
7.4.2 Erstellen und Konfiguration virtueller Geräte 73
7.4.3 Durchführung von Messungen 74
7.5 Verwendete Algorithmen 74
7.6 Resultierende Parameter 75
7.7 Resultierende Codegrössen 76

8 Evaluation 77
8.1 Theoretische Performance von Zwei-Hop-Protokollen 77
8.2 Messungen einzelner Protokolle 78
8.3 Emulation auf dem Network Emulation Testbed 79
8.4 Einbindung der Zugriffskontrollschicht 80
8.5 Mögliche Angriffe 81
8.5.1 Arten von Angreifern 81
8.5.2 Angriffe auf die Kommunikation 82
8.5.3 Angriffe auf die Linkerstellung 83
8.5.4 Angriffe bei der Gruppenkommunikation 84
8.5.5 Angriffe bei der Selbstheilung 85
8.5.6 Übernahme eines Netzes 85

9 Ausblick 87
9.1 Mögliche Erweiterungen und Verbesserungen 87
9.2 Fazit 88

Literaturverzeichnis 89

Abbildungsverzeichnis

1.1 Schichtenarchitektur des Gesamtsystems 10

4.1 Aufbau des Netzwerkes (Quelle: [25]) 33
4.2 Aufbau einer neuen Verbindung (Quelle: [25]) 34

5.1 Aufbau der Sicherheitsschicht 37
5.2 Aufbau eines Paketes 39

6.1 Imprint-Protokoll: Sender 45
6.2 Imprint-Protokoll: Empfänger 45
6.3 Verbindungsaufbau in Zwei-Hop-Protokollen 49
6.4 Pfadfindung-Protokoll: Discovery 50
6.5 Pfadfindung-Protokoll: Initialisierung 51
6.6 Pfadfindung-Protokoll: Weiterleitung 51
6.7 Pfadfindung-Protokoll: Ankunft 52
6.8 Pfadfindung-Protokoll: Rückleitung 53
6.9 Pfadfindung-Protokoll: Antwort 54
6.10 Abmelden eines Gerätes im Netzwerk 61

7.1 Simulationsoberflaeche 71

8.1 Erfolgswahrscheinlichkeit eines Verbindungsaufbaus 78
8.2 Messungen bei verschiedenen s 79
8.3 Messungen bei verschiedenen Verbindungsaufbauwahrscheinlichkeiten . . . 80

Tabellenverzeichnis

1.1 Beispiele fuer Geräte in der Heimautomatisierung 9

3.1 Beispielkonfiguration nach Eschenauer und Gligor 24
3.2 Zusammenfassung der Ansätze, Teil 1 30
3.3 Zusammenfassung der Ansätze, Teil 2 30

5.1 Benötigte Einträge im Schlüsselspeicher 38

6.1 Statusvariablen bei der Verbindungserstellung 51

7.1 Parameter für den Verbindungsaufbau 75
7.2 Codegrössen 76

Kapitel 1

Einleitung

In diesem Kapitel werden – nach einer Einführung in das Gebiet der Heimautomatisierung – Gründe für die Notwendigkeit von Sicherheitsmassnahmen dargestellt. Es folgt eine Beschreibung der in dieser Arbeit betrachteten Szenarien, gefolgt von der Aufgabenstellung. Um eine Einbindung im betrachteten Gesamtsystem darzustellen, werden anschliessend die späteren Einsatzgebiete vorgestellt. Eine Einordnung der Sicherheitsschicht in das Gesamtmodell sowie eine Aufstellung aller Anforderungen schliessen das Kapitel ab.

1.1 Aufbau

Diese Arbeit ist unterteilt in drei Bereiche. Der erste Bereich umfasst die theoretischen Überlegungen des behandelten Gebietes. Hierzu gehören die Einleitung mit einer Analyse der Anforderungen, ein Überblick über das zu entstehende System, sowie die Betrachtung der vorhandenen Ansätze.

Der zweite Teil stellt die Entwicklung der Sicherheitsschicht dar. In diesem wird zuerst das grundlegende Verfahren erläutert, danach schrittweise die Sicherheitsschicht und zuletzt die benötigten Protokolle entwickelt.

Den letzten Teil bildet die Auswertung des vorgestellten Ansatzes. Er besteht unter anderem aus der Erstellung und Analyse einer prototypischen Implementierung, daraus resultierenden Messungen und der Bewertung von möglichen Angriffen.

1.2 Motivation

Während der Unterstützung mittels Mikrokontroller im täglichen Leben noch vor einigen Jahren nur geringe Bedeutung zukam, zeigt sich mehr und mehr ein Anstieg von intelligenten kleinen Systemen, die wir in unserem Umfeld finden. Dies beginnt bei Lampen, die sich durch eine Sensormeldung automatisch bei Betreten des Raumes einschalten oder Feuermeldeanlagen, welche miteinander kommunizieren, um gemeinsam Alarm auszulösen. Des Weiteren existieren bereits intelligente Kühlschränke, welche Lebensmittel autonom nachbestellen und entsorgen können.

All diese Systeme müssen miteinander kommunizieren, um Ihre Aufgaben zu erfüllen. Dies hat zwei Gründe: Erstens werden Einzelgeräte günstiger, wenn Sensorik und Aktorik

nicht gemeinsam verbaut werden müssen, zweitens entstehen mit der Vision des Ubiquitous Computing neue Einsatzzwecke, welche durch das Vereinigen vieler verteilt entstehender Informationen ermöglicht werden. Ein Beispielszenario, die Verwendung solcher Systeme im Bereich der Heimautomatisierung, wird dabei im nächsten Kapitel genauer vorgestellt.

Da eine kabelgebundene Infrastruktur oftmals (aus rechtlichen oder finanziellen Gründen) nicht realisierbar sowie bei mobilen Geräten unvorteilhaft ist, bietet sich eine Kommunikation über Funk an. Eine solche Entscheidung impliziert jedoch auch, dass – wie in vielen bisherigen Automatisierungssystemen [7] angenommen – das Kommunikationsmedium nicht mehr als ein sicherer und vor Angreifern geschützter Kanal anzusehen ist. Daraus entsteht die Notwendigkeit von Sicherheitsmechanismen auf den einzelnen Geräten, um Aussenstehenden das Abhören von Sensorinformationen sowie das unerlaubte Befehligen von Aktoren zu verbieten.

In dieser Arbeit wird deshalb ein neues Verfahren vorgestellt, um Sicherheit bei der Funkkommunikation von kleinen Geräte im Bereich des Ubiquitous Computing zu gewährleisten. Hierbei wird angenommen, dass in Zukunft sehr viele Geräte mittels Mikrocontroller und Funkschnittstellen befähigt werden, intelligent und gemeinsam zu handeln. Des Weiteren werden die Kosten solcher Systeme – auf Grund der Vielzahl an verkauften Geräten – extrem sinken und in vielen Alltagsgegenständen zu finden sein. Dennoch muss dabei von kleinen und ressourcenbeschänkten Geräten ausgegangen werden, welche beispielsweise mittels Batterie versorgt werden. Aus diesem Grund ist weder die Verfügbarkeit modularer Arithmetik (wie sie beispielsweise für asymmetrische, kryptographische Funktionen benötigt wird), noch die grosser Speichermengen anzunehmen. Ebenso ist die Eigenschaft der "Tamper resistance" wichtig. Diese bedeutet einer Sicherung der geheimen Daten vor Angreifern mit physikalischem Kontakt. Solche "Tamper resistant" Geräte löschen zum Beispiel bei Versuch einer Manipulation ihre Daten. Diese Eigenschaft ist meist nur bei Smart Cards oder teuren Kryptoprozessoren annähernd gegeben, jedoch nicht in Alltagsgegenständen. So würde eine solche Glühlampe mit einem Vielfachen des Grundpreises an Mehrkosten kaum Akzeptanz finden.

In dieser Arbeit wird ein Verfahren vorgestellt, welches die hier genannten Anforderungen erfüllt. Eine Zusammenfassung aller Anforderungen ist hierbei in Kapitel 1.5 enthalten.

1.3 Szenarien

Um ein Beispiel für den Einsatz der beschriebenen Systeme zu nennen, wird nun ein Szenario im Bereich der Heimautomatisierung beschrieben. Ebenso zu nennen sind hierbei Unternehmensgebäude oder öffentliche Einrichtungen, welche zur Verwaltung, für Besucher oder für die Gebäudesteuerung hier beschriebene Systeme einsetzen.

Im Bereich der Heimautomatisierung gibt es eine Vielzahl von Sensoren und Aktoren. Einfache Beispiele sind in Tabelle 1.1 aufgeführt. Hierbei ist zu beachten, dass Zielsysteme durchaus auch ressourcenstarke Systeme (wie zum Beispiel Fernseher), jedoch auch Geräte wie eine mit Batterien versorgte Fernbedienung sind.

Ein weiteres Merkmal, insbesondere bei der privaten Heimautomatisierung, ist der inkrementelle Kauf bzw. Zukauf der Systeme. Die einmalige komplette Automatisierung

Sensoren	Aktoren
Fernbedienungen	Lampen
Bewegungsmelder	Heizungen
Schalter	Rolläden
Zeitschalter	Haushaltsartikel
Rauchmelder	Fernseher
Regen- oder Temperatursensoren	Audiosysteme
Personenerkennung (über Biometrie)	Türen (bzw. Türschlösser)
Gestenerkennung	Alarmsysteme
Spracheingabe	Anzeigesysteme

Tabelle 1.1: Beispiele fuer Geräte in der Heimautomatisierung

einer Wohnung – insbesondere aus einer Hand – ist aus folgenden Gründen nicht zu erwarten:

- Gesamtinvestitionen sind für Privatpersonen meist nicht auf einmal tragbar.
- Käufer wollen vor einer grossen Investition erst mit kleineren Systemen Erfahrungen sammeln.
- Restlebenszeiten bereits existierender Geräte sind zu beachten.
- Markenbewusstsein wird nicht wegen des Sicherheitsherstellers geändert. Ein Kunde wechselt demnach nur ungern (wenn überhaupt) den Hersteller seiner Heimelektronik, nur weil dessen System nicht mit dem angedachten Sicherheitssystem kompatibel ist.

Dies führt dazu, dass in ein Heimautomatisierungsnetz nach und nach neue Geräte von verschiedensten Herstellern eingebracht werden müssen. Eine Vorkonfiguration für das Zielnetzwerk auf Seiten des Herstellers oder des Verkäufers ist demnach nicht möglich.

Eine weiteres Merkmal im Gebiet der Heimautomatisierung ist die Anforderung, kein zentrales Steuergerät (einen "Single Point of Failure", SPoF) zu verwenden, da durch einen Ausfall (oder eine Kompromittierung) die gesamte Hauselektronik betroffen ist. Des Weiteren würde dies die Anschaffung einer ggf. teuren Spezialhardware bedeuten – selbst in einem anfänglichen Netz mit nur einer Lampe und einer Fernbedienung.

Funkgebundene Heimautomatisierungselektronik wird derzeit unter anderem von Unternehmen wie Conrad Elektronik oder ELV angeboten. Letztere bietet hierbei insbesondere die Serien FS20 und FS10 an [8]. Diese enthalten neben Licht-, Dimm- und Steckdosenschaltern, Fernbedienungen, Heizungssteuerungen, elektronischen Rolläden, Licht- und Regensensoren auch Telefon- und Computeranbindungen.

Betrachtet man jedoch die Sicherheit dieser ersten verfügbaren Systeme, so muss man – wie auch bei anderen Standards wie X10 [28] – feststellen, dass hierfür keinerlei Mechanismen vorgesehen sind. Lediglich die Existenz sogenannter "Hauscodes" ist festzustellen. Diese lassen sich an den einzelnen Geräten (beispielsweise mittels eines DIP-Schalters) einstellen. Anstelle einer darauf basierenden Verschlüsselung wird dieser Code jedoch bei Funkanweisungen im Klartext mitgeschickt. Er dient damit mehr zur Identifikation des

zugehörigen Netzwerkes, um so zum Beispiel das gegenseitige Beeinflussen mehrerer Haushalte zu vermeiden. Dies führt bei einer grösser werdenden Popularität zu der Notwendigkeit neuer – mit Sicherheitsattributen ausgestatteter – Systeme.

1.4 Aufgabenstellung

Auf handelsüblichen Mikrocontrollern soll ein System entstehen, welches die sichere Kommunikation vieler Geräte miteinander ermöglicht. Das Gesamtsystem hat den in Abbildung 1.1 beschriebenen Aufbau.

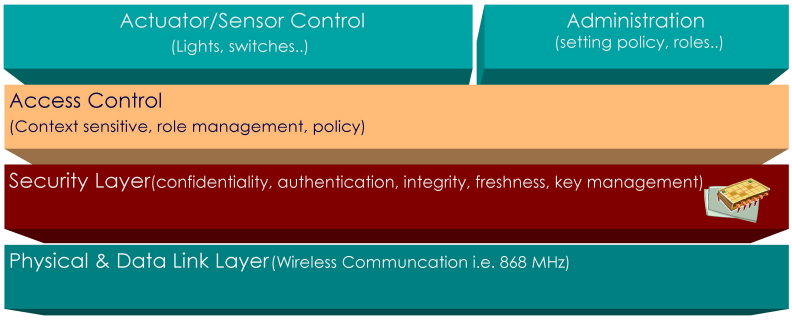

Abbildung 1.1: Schichtenarchitektur des Gesamtsystems

In der Diplomarbeit von Ulrich Wurst, *Einsatz von kryptographischen Verfahren auf stark ressourcenbeschränkten Geräten* [27], wurde untersucht, welche kryptographischen Verfahren (Soft- oder Hardware) sich für solche ressourcenbeschränkte Geräte eignen.

Aufbauend auf den Erkenntnissen dieser Arbeit wird hier eine Architektur für die Sicherheitsschicht (Security Layer) entworfen und prototypisch implementiert. Diese Schicht soll eine sichere Kommunikation zwischen Mitgliedern eines Netzwerkes ermöglichen. Dabei sollen beliebige Geräte untereinander authentifiziert und geheim Nachrichten austauschen können. Die einzelnen Anforderungen hierfür sind in Kapitel 2.1 beschrieben.

Aufbauend auf der Sicherheitsschicht wurde parallel zu dieser Arbeit in der Diplomarbeit von Jochen Noller, *Design und prototypische Implementierung einer Zugriffskontrollschicht für ressourcenbeschränkte Geräte* [19], eine Zugriffskontrolle (Access Control) entworfen, in der einzelnen Geräten ein Rechtemanagement ermöglicht wird, um – aufbauend auf der von der Sicherheitsschicht gebotenen Authentifizierung – eine Autorisierung für bestimmte Aktionen zu bestimmen.

Ziel ist es, diese Aufgaben – zusammen mit einer Kommunikationsschicht für eine Funkschnittstelle und zusätzlicher Anwendungslogik zum Steuern der Aktoren und Abfragen der Sensoren – auf einer gemeinsamen Hardwareplattform mit einem Mikrokontroller zu vereinigen.

1.5 Anforderungen

Im Folgenden werden die Anforderungen, wie sie in den bisherigen Kapiteln auftraten, genauer aufgezeigt.

Bilige Herstellung Da Sicherheit selbst in einfachsten Geräten wie Schaltern oder Lampen benötigt wird, dürfen bei diesen die Kosten für die zusätzliche Hardware nicht überwiegen, um eine Akzeptanz zu erreichen. Daraus leitet sich die nächste Anforderung ab.

Kleine und ressourcenbeschränkte Geräte Kostengünstige Hardware verfügt über wenig Ressourcen wie Programm- oder Datenspeicher. Insbesondere kann nicht davon ausgegangen werden, dass ein Gerät so viel Speicher besitzt, um eine Liste aller anderen Geräte im Netzwerk speichern zu können. Hierzu müssen Verfahren entwickelt werden, die diese Beschränkung tolerieren.

Stromsparende Geräte Bei mobilen, batteriebetriebenen Geräten ist darauf zu achten, das der Stromverbrauch minimiert wird. Dies muss auf der einen Seite durch verbrauchsarme Prozessoren mit geringer Taktrate, auf der anderen Seite durch die Vermeidung unnötiger Kommunikation realisiert werden.

Symmetrische Kryptographie Auf Grund der kleinen und langsamen Geräte ist es nicht möglich, asymmetrische Verfahren zu verwenden. So benötigt zum Beispiel selbst ein relativ ressourcenstarker Prozessor wie der des "Palm Pilot" für die Berechnung einer 1024 Bit RSA-Signatur ungefähr 36 Sekunden [4].

Kein Single Point of Failure (SPoF) Um erstens den Ausfall aller Systeme durch einen einzigen Ausfall zu verhindern sowie zweitens die Notwendigkeit zum Kauf und zur Bereitstellung einer Basisstation zu vermeiden, wird verlangt, dass das System vollkommen dezentral ist und Ausfälle toleriert werden.

Kein ultimatives Vertrauen in Architektur oder Geräte Da die Geräte – wie bereits in der Motivation erläutert – auf Grund der geringen Kosten sowie technischer Erwägungen [24] nicht Tamper resistant sein können, muss davon ausgegangen werden, dass einzelne Geräte von Angreifern gestohlen oder ausgelesen werden können.

Skalierbarkeit des Vefahrens Da keinerlei Beschränkungen in der Grösse des Netzwerkes gemacht werden, müssen die Verfahren mit der Anzahl der Geräte skalieren. So sind – in Anbetracht des expotentiellen Wachstums von Mikrocontrollern in unserem Umfeld – Netzwerke mit mehreren tausend Geräten denkbar.

Keine vorgegebene Netzwerkgrösse Ein Heimautomatisierungsnetzwerk wird nach und nach um Funktionalität erweitert. Demnach ist nicht vorrauszusetzen, dass die spätere Gesamtgrösse bereits bei der Installation des ersten Gerätes bekannt ist.

Keinerlei Vorkonfiguration Auf Grund des inkrementellen Wachstums des Netzwerkes können einzelne Geräte nicht bereits beim Hersteller an das Zielnetzwerk angepasst und konfiguriert werden.

Kapitel 2

Systemüberblick

In diesem Kapitel sollen die theoretischen Anforderungen, wie sie im letzten Kapitel erarbeitet wurden, genauer spezifiziert werden. Dabei werden die tatsächlichen Gegebenheiten der Zielplattform untersucht.

2.1 Anforderungen an Kommunikationssemantiken

Bevor die benötigten Eigenschaften analysiert werden, müssen zuerst die Eigenschaften des Begriffes "Sicherheit" definiert werden. Da meist die englischen Bezeichnungen bekannter sind, werden diese hier mit aufgeführt. Sicherheit beinhaltet:

Vertraulichkeit/Geheimhaltung (Secrecy) Die Vertraulichkeit beschreibt, dass eine Nachricht nicht von Unbefugten gelesen werden kann. Sie ist im hier beschriebenen Szenario essentiell, um Angreifer keine Möglichkeiten zu geben, sämtliche Aktionen mitzulesen.

Authentizität (Authenticy) Die Authentizität stellt bei dem Empfänger einer Nachricht sicher, dass der Sender der Identität entspricht, der er vorgibt zu haben. Speziell für die spätere Verarbeitung einer Nachricht in der Zugriffskontrolle ist die sichere Bestimmung des Senders elementar. Angreifer (oder beliebige Geräte) könnten ohne dieser Eigenschaft sonst Nachrichten im Namen anderer Geräte aussenden.

Frische (Freshness) Unter der Frische wird verstanden, dass eine einzelne Nachricht nur ein einziges Mal verarbeitet werden kann. Ist dies nicht der Fall, so könnte ein Angreifer durch Abspielen bekannter und aufgezeichneter Nachrichten Aktionen erneut auslösen (sogenannte "Replay Atacken").

Nichtabstreitbarkeit (Non-repudiation) Die Nichtabstreitbarkeit sagt aus, dass im Nachhinein beliebige im Netz befindliche Geräte überprüfen können, ob eine Nachricht von dem angegebenen Sender stammt. Dies ist dahingehend eine Erweiterung, dass nicht nur der Empfänger zum Zeitpunkt der Ankunft dies überprüfen kann. Mittels dieser Eigenschaft, welche meist bei asymmetrischer Kryptographie gegeben ist, ist es zum Beispiel möglich, Fehlverhalten einzelner Geräte zu veröffentlichen.

Anonymität (Privacy) Die Anonymität beinhaltet, dass es möglich ist, eine Nachricht zu versenden, ohne dass der Sender zu ermitteln ist.

Um die Eigenschaften der symmetrischen Verschlüsselungsverfahren zu verwenden, werden die verlangten Semantiken nach dem Empfängerkreis unterschieden.

Kommunikation zwischen zwei Geräten (Unicast Kommunikation) hat vertraulich, authentifiziert und frisch zu erfolgen. Dies ermöglicht höheren Schichten, andere Geräte auf eine geheime Art zu befehligen, ohne Angreifern die Möglichkeit zu geben, Nachrichten mitzulesen oder (sei es durch Wiederholen) einzubringen.

Gruppenkommunikation an alle Geräte in einem Netzwerk (Multicast) hat geheim zu erfolgen. Authentizität ist hierbei nur in Bezug auf die Gruppenmitgliedschaft erforderlich. Die Frische wird ebenso herabgestuft: Es wird eine Maximalzeit verlangt, nach der das Wiederholen einer alten Nachricht nicht mehr möglich ist. Anonymität wird in keinem der beiden Komminikationsarten verlangt; es ist jedoch anzumerken, dass eine Gruppennachricht in dieser Arbeit keinerlei Senderkennung verlangt.

2.2 Beschreibung der Hardwarearchitektur

Da die Hardwareplattform nicht vollständig festgelegt und eine Sicherheitsschicht unabhängig von deren konkreten Eigenschaften wünschenswert ist, werden in diesem Kapitel zwei verschiedene Mikrocontroller vorgestellt. Der erste ist der ATMega16L von der Firma Atmel; ein Prozessor, welcher der weit verbreiteten AVR Familie angehört und auch bei anderen bereits bestehenden Ansätzen genannt wird. Der zweite Mikrocontroller ist der ebenfalls verbreitete PIC18F6620 der Firma Microchip, welcher unter anderem in den Teco Smart-Its [23] verwendet wird. Diese stellen um Funkkommunikation, Betriebssystem und Sensorik angereicherte Geräte dar und eignen sich somit für die beschriebenen Szenarien als Prototyp.

Der ATMega16L ist ein 8 Bit RISC Mikrocontroller, welcher im Gegensatz zum ATMega16 für geringen Stromverbrauch optimiert ist. Er verfügt dadurch nur über eine Taktrate von 8 MHz bei einer Betriebsspannung von 3 Volt. Im Bereich der Ressourcen sind insgesamt 16 KB Flash Speicher für Programme, 512 Byte EEPROM Speicher für Daten sowie 1024 Byte RAM verfügbar.

Der PIC18F6620, der im Smart-IT "Particle 2/06" verwendet wird, ist ebenfalls ein 8 Bit RISC Mikrocontroller. Er ist leistungsstärker als der ATMega16 und wird mit 20 MHz betrieben. Er besitzt insgesamt 64 KB Flash Speicher und 1024 Byte EEPROM, sowie 4 KB RAM. Auf dem "Particle 2/06" befinden sich zusätzliche 32 KB EEPROM, auf dem sich weitere Daten auslagern lassen.

2.3 Beschreibung der Kommunikationsschicht

Um einen Einblick in die auf diesen Geräten verfügbaren Kommunikationsmöglichkeiten zu geben, wird die Kommunikation der Smart-Its zuerst kurz beschrieben:

Deren Kommunikation findet auf dem 868.35 oder dem 915 MHz Frequenzband statt. Auf diesem wird ein TDMA (Time Division Multiple Access) Verfahren eingesetzt, bei welchem die Sender Zeitschlitze zur Kommunikation nutzen. Die verwendeten Strategien sind hierbei einerseits das vorherige Abhören des Trägers (Carrier Sense, CS), andererseits die Verwendung von Algorithmen zur Vermeidung (im Gegensatz zur nachträglichen Erkennung) von Kollisionen (Collision Avoidance, CA). Dadurch lässt sich bei den Smart-Its

eine Bandbreite von 125 kbps erreichen.

Darauf aufbauende Dienste (Segmentierung von Nachrichten, bestätigtes Senden von Paketen, ...) verringern diese Datenrate, so dass neben den Speicherressourcen auch die Kommunikationsbandbreite als stark beschränkte Ressource anzusehen ist.

2.4 Cryptoalgorithmen

Als Ergebnis der Diplomarbeit von Ulrich Wurst [27] stellte sich folgende Konfiguration zur Verwendung in der Sicherheitsschicht als am besten geeignet heraus:

Verschlüsselungsalgorithmus AES Zur Verschlüsselung wurde der Advanced Encryption Standard des NIST (National Institut of Standards and Technology), auch als Rijndael bekannt, gewählt. AES ist eine symmetrische Blockchiffre, die auf Blöcken von 128 Bit arbeitet und in diesem Szenario einen ebenfalls 128 Bit langem Schlüssel verwendet.

Secure Hash RIPE-MD128 Der gewählte Hashalgorithmus RIPE-MD128 ist Teil des EU-Projektes RIPE (Race Integrity Primitives Evaluation). Das Verfahren basiert auf dem MD4-Verfahren von Ron Rivest und bildet einen Block von 512 Bit auf 128 Bit ab.

Generator für Zufallszahlen mittels physikalischer Beobachtungen Es wird ein von Herrn Wurst vorgeschlagener Zufallszahlengenerator verwendet, der als Rauschgenerator zwei Transistoren benutzt. Dabei wird ein durch thermisches Rauschen erzeugter Bitstrom generiert, welcher zur Erhöhung der Entropie mittels eines Secure Hashes umgewandelt wird.

Letztendlich ist dies jedoch ein Designkriterium, welches unabhängig von der Struktur und den meisten Protokollen der Sicherheitsschicht ist und somit – abhängig von Zielplattform (Grösse von RAM und Flash, Prozessorarchitektur, ...) und Präferenz der Algorithmen – gewählt werden kann.

Benötigt werden jedoch für jede der drei Kategorien (Verschlüsselungsfunktion, Secure Hash, Zufallszahlengenerator) eine entsprechende Hardware- oder Softwareschnittstelle auf der jeweiligen Zielplattform.

Kapitel 3

Analyse bestehender Ansätze

Sieht man sich existierende Ansätze im Bereich der Heimautomatisierung [7, 28, 8] an, so muss man feststellen, dass diese – selbst wenn die Kommunikation über Funk erfolgt – kaum bzw. keinerlei Sicherheit gegenüber Angreifern bieten. Es ist meisst einzig ein "Geheimnis" in Form eines Hauscodes verfügbar, welches bei jeder Kommunikation übertragen wird und somit mehr der Vermeidung von Beeinträchtigungen zwischen mehreren Haushalten mit der gleichen Technik dient.

Aus diesem Grund werden in diesem Kapitel Ansätze aus anderen Bereichen mit einer ähnlichen Zielsetzung vorgestellt.

Existierende Ansätze in funkgebundenen Ad hoc- sowie Sensor-Netzwerken lassen sich in vier Bereiche unterteilen, basierend auf der Art der Verschlüsselung sowie auf der Entscheidung über die Zuhilfenahme einer zentralen Stelle.

Asymmetrische, zentrale Ansätze Die einfachste Möglichkeit, ein sicheres Netzwerk aufzubauen, ist die Verwendung einer zentralen Certification Authority(CA): Einer Dienststelle, welche für das Signieren und damit für das Bestätigen der Echtheit eines öffentlichen Schlüssels verantwortlich ist. Probleme eines solchen Ansatzes sind neben der Verwendung asymmetrischer Verschlüsselungsalgorithmen, auf die später eingegangen wird, die Ressourcenbeschränkungen der einzelnen Geräte, die notwendige Verfügbarkeit der CA bei Einbringung neuer Geräte bzw. Linkaufbau sowie die Einführung eines SPoF.

Asymmetrische, dezentrale Ansätze Damit ein SPoF – wie bei einer zentralen Authentifizierung – nicht mehr existiert, wird die CA bei dezentralen Ansätzen auf die einzelnen Geräte verteilt. Hierbei wird meist Schwellwertkryptographie eingesetzt, um eine Authentifizierung von einer Mindestmenge zertifizierender Geräte abhängig zu machen [12, 30].

Da asymmetrische im Verhältnis zu symmetrischer Kryptographie um vieles mehr Ressourcen benötigt, ist deren Verwendung bei ressourcenbeschränkten Geräten oftmals auf Grund von Speicher-, Energie- oder Zeitbeschränkungen nicht möglich. So benötigt zum Beispiel die Berechnung einer 1024 Bit RSA-Signatur auf einem "Palm Pilot" ungefähr 36 Sekunden [4]. Es existieren zwar Lösungen, bei denen zusätzliche Geräte mit hoher Performance, welche – ohne das einzelne Geräte sicherheitsrelevante Informationen preisgeben müssen – bei Aufgaben wie der Schlüsselgenerierung helfen können, jedoch werden

dadurch nur Geschwindigkeitszuwächse um etwa den Faktor 2 erreicht.

Symmetrische, zentrale Ansätze Aufgrund der genannten Schwierigkeiten asymmetrischer Verfahren, wurden symmetrische Lösungen in der letzten Zeit in vielen Veröffentlichungen vorgeschlagen.

Wie bei den asymmetrischen Ansätzen ist eine mögliche Lösung der Einsatz eines zentralen Servers, welcher als Mittelsmann für einen paarweisen sicheren Kanalaufbau agiert. Dies wird unter anderem bei SPINS [21] vorgeschlagen, einem System, welches im Kapitel 3.2 genauer betrachtet wird. Ein anderes Beispiel für einen solchen Ansatz in kabelgebundenen Netzwerken ist das Kerberos Protokoll [13]. Hierbei speichert ein zentraler Server die Schlüssel der einzelnen Geräte.

Symmetrische, dezentrale Ansätze Der vermutlich vielversprechendste Ansatz ist die Verwendung von symmetrischer Kryptographie in dezentralen Algorithmen. In diesem Bereich wurden bisher nur wenige Ansätze vorgestellt, welche alle im späteren genauer analysiert werden.

In den nächsten Kapiteln werden einzelne Ansätze, welche für diese Arbeit von besonderer Bedeutung sind, genauer betrachtet. Hierbei werden – wenn möglich – die folgenden Eigenschaften jeweils getrennt analysiert:

Netzaufbau Wie gelangen Geräte in das Netzwerk?

Kommunikation Wie können beliebige Geräte miteinander kommunizieren?

Ausfallsicherheit Wie verhalten sich Geräte bei einzelnen Ausfällen?

3.1 Asymmetrische, dezentrale Ansätze

Die Veröffentlichung "*The Quest for Security in Mobile Ad Hoc Networks*" aus dem Jahr 2001 von Hubaux et al. umreißt zunächst Sicherheitsprobleme, welche bei einem mobilen ad hoc Netzwerk auftreten und liefert danach eine eigene Lösungsmöglichkeit, welche zwar komplett dezentral ist, jedoch auf asymmetrischer Kryptographie aufsetzt. Dies begründet sich durch einen Focus auf leistungsstärkere Geräte (wie z.B. Notebooks) unter Benutzerkontrolle.

3.1.1 Angriffsmöglichkeiten

Hubaux et al. unterteilen Angriffe auf sogenannte Basismechanismen und Sicherheitsmechanismen. In ersteren Bereich fallen folgende Gefahren:

Diebstahl und Kompromittierung Geräte in mobilen Netzwerken können nicht eingeschlossen werden. Als Beispiel liefern die Autoren die Kosten der Mobilfunkbetreiber von mehreren 100 millionen Dollar durch den Diebstahl von Funktelefonen. Lösungsmöglichkeiten seien tamper resistant Smart Cards oder Security Processors [1].

Funkkommunkation Aufgrund dieses Übertragungsmediums sind ad hoc Netzwerke gegen Angriffe wie Abhören, dem aktive Verfälschen von Informationen, sowie dem Stören der Frequenzbereichen verwundbar. Als Lösungen weisen die Autoren auf Techniken wie Spread Spectrum oder Frequency Hopping[9] hin.

Routing Da in einem ad hoc Netzwerk jedes beteiligte Gerät vermittelnde Aufgaben hat, wird dies hier als besonderes Problem herausgestellt. Angriffsmöglichkeiten seien hierbei z.B. das Aussenden von falschen Routinginformationen oder schlicht die Weigerung einzelner Geräte an einer Beteiligung. Eine mögliche Lösung sei hierbei ebenfalls das Sichern der Routingalgorithmen in Smart Cards bzw. Security Prozessoren. Des Weiteren werden Überwachungen mittels "watchdogs" und "pathrater" [16] vorgeschlagen, bei denen Fehlverhalten von benachbarten Knoten erkannt [29] wird und Routingpfade danach an diesen sich fehlerhaft verhaltenden Knoten vorbeigeführt werden.

Vertrauen Das eigene Initialisieren, das Hinzufügen neuer Geräte oder die Entdeckung neuer Nachbarn wird als letztes Problem genannt. Fragen, wem das System (in einem Dezentralen Fall) vertrauen und wem es Informationen preisgeben soll, können nur durch Hersteller oder übergeornete Institutionen beantwortet werden.

Angriffe auf die Sicherheitsmechanismen zielen auf das Schlüsselmanagement des Systems ab. Im speziellen betrachten Hubaux et al. hierbei Probleme bei asymmetrischer Kryptographie:

Verteilen von Schlüsseln und Revocations Existiert keine zentrale Dienststelle, welche für die Schlüsselverwaltung zuständig ist, so müssen die einzelnen Knoten dies gemeinsam ohne den Verlust von einzelnen Schlüsseln gewährleisten

Identitäten Ein Problem von öffentlichen Schlüsseln ist die Verbindung eines Schlüssels mit einer bestimmten Identität. Um dies zu gewährleisten, wird eine "trusted authority" – zumindest bei der Erstellung des Netzwerkes – benötigt. Lösungen dieses Problems seien entweder das Emulieren einer Certification Authority (z.B. mittels Schwellwertkryptographie) oder eine völlig verteilte Lösung, bei der beispielsweise von den Benutzern ein gemeinsamer Schlüssel manuell eingegeben wird.

Im Folgenden wird das Design, welches aus diesen Überlegungen entstand, kurz beschrieben.

3.1.2 Kurzbeschreibung

Der vorgestellte Lösungsansatz bezieht sich stark auf Systeme wie Pretty Good Privacy (PGP) [32], welche ein von den Benutzern selbst erzeugtes "Web of Trust" – ein gerichteter Graph, in denen Benutzer bzw. Schlüssel (Knoten) durch Signaturen (Kanten) anderer Schlüssel deren Identität bezeugen. Durch Pfade zwischen zwei Knoten lässt sich daraufhin ein Vertrauen ableiten. Im Gegensatz zu PGP wird jedoch ein System vorgeschlagen, welches keinerlei zentral verwaltete Server für die Signaturen wie bei PGP verwendet.

3.1.3 Netzaufbau

Bei Einbringung eines neuen Gerätes muss die Identität (in diesem Falle der hierzu korrelierende öffentliche Schlüssel) von einer Menge bereits im Netzwerk befindlicher Knoten zertifiziert werden. Gleiches gillt für die Gegenrichtung: Das neue Gerät zertifiziert eine bestimmte Teilmenge der vorhandenen Geräte. Daraufhin speichert das neue Gerät alle seine selbst erzeugten Zertifikate sowie eine Teilmenge von fremden Zertifikaten. Für die Auswahl der fremden Zertifikate wird der "Shortcut Hunter" Algorithmus vorgestellt. Dieser versucht, möglichst jene Zertifikate (Kanten) zu verwenden, mittels derer der Durchmesser des Gesamtgraphes (des Web of Trust) möglichst gering bleibt. Ein Shortcut ist hierbei folgendermaßen definiert: Wird ein Shortcut zwischen zwei Geräten entfernt, so ist danach der kürzeste Pfad zwischen diesen Geräten (sehr viel) länger als zwei Kanten.

3.1.4 Kommunikation

Damit zwei Geräte miteinander auf eine sichere und authentifizierte Weise kommunizieren können, muss ein (wie in Kap. 3.1.2 beschrieben) Zertifikatpfad zwischen beiden Geräten (in beide Richtungen) bestehen. Hierfür vereinen diese beiden Knoten ihre Zertifikatlisten und versuchen darin einen Pfad zu finden. Dies zeigt einerseits die Notwendigkeit einer guten Auswahlfunktion, andererseits die probabilistische Funktionsweise dieses Verfahrens. Befindet sich in der Vereinigung der beiden Teilmengen kein Pfad, so schlägt eine authentifizierte Kommunikation fehl. Messungen mit dem "Star Shortcut Hunter" – einem optimierten Algorithmus zur Teilmengenbestimmung – ergaben an einer stark vernetzten Teilmenge des original PGP Web of Trust Wahrscheinlichkeiten von über 0,95 bei insgesamt 8659 Knoten, dabei je 10 ausgestellten Zertifikaten und einer Teilmenge von 200 gespeicherten Zertifikaten pro Knoten.

Da davon ausgegangen wird, dass kein Gerät falsche Zertifikate ausstellt, reicht die Findung eines einzige Pfades aus. Lockert man diese Einschränkung, so schlagen die Autoren die Suche mehrerer knotendisjunkter Pfade vor.

Wurde ein Pfad gefunden, so können zum Beispiel symmetrische Schlüssel (sogenannte Session keys) mit dem öffentlichen Schlüssel des einen Gerätes verschlüsselt verschickt und für die weitere Kommunikation verwendet werden.

3.1.5 Ausfallsicherheit

Da bei dem Versuch, eine Kommunikation zwischen zwei Geräten aufzubauen, auch nur diese beiden Geräte aktiv sind, können andere Geräte – auch wenn ihre Zertifikate für eine Pfadfindung benötigt werden – ohne Probleme für den Algorithmus ausfallen (im Sinne von Zerstörung, Stromausfall oder anderer Defekte). Hierbei sei nochmal angemerkt, dass das Auslesen eines privaten Schlüssels dazu führen würde, dass zusätzliche – falsche – Zertifikate erstellt werden könnten und somit sich ein Angreifer unter Umständen als eine beliebige Identität ausgeben könnte.

3.2 Zentrale, symmetrische Ansätze

Bei "*SPINS: Security Protocols for Sensor Networks*" [21] handelt es sich um eine 2002 veröffentlichte Protokollsuite für Sensornetzwerke, entwickelt von Perrig et al. an der UC

Berkeley. Hierbei wird eine zentrale, symmetrisch Lösung für Netzwerke mit mehreren tausend Geräten präsentiert.

3.2.1 Kurzbeschreibung

SPINS bietet insgesamt zwei Bereiche bzw. Protokolle an:

SNEP Das "Secure Network Encryption Protocol" bietet sichere Kommunikation zwischen zwei Geräten mit den Eigenschaften Vertraulichkeit, Authentizität, Nichtabstreitbarkeit sowie Frische.

μTESLA Bei μTESLA handelt es sich um eine auf kleine und ressourcenbeschränkte Geräte angepasste Version des TESLA-Protokolls. Diese bietet authentifizierten Broadcast ohne die Verwendung asymmetrischer Kryptographie in einem lose zeitsynchronisiertem Netzwerk.

Als kryptographische Methoden werden bei beiden Protokollen einzig ein MAC[1] - ein Secure Hash mit sehr kleiner Imlementierung, somit für ressourcenbeschränkte Geräte geeignet - und ein symmetrischer Verschlüsselungsalgorithmus eingesetzt.

Während die Entwendung oder Manipulation einzelner Sensoren nicht ausgeschlossen wird und ein solches auch keine Sicherheitsprobleme bei der Kommunikation zwischen anderen Geräten auslöst, wird die Basisstation als weder manipulier- noch auslesbar angesehen; sie repräsentiert einen SPoF. Des Weiteren wird deren Lebenszeit als größer als die aller Sensoren angesehen – die Basisstation darf demnach nie ausfallen.

3.2.2 Netzaufbau

Bei Einbringung neuer Geräte in ein Netzwerk gehen Perrig et al. davon aus, dass bereits ein Geheimnis existiert, welches nur der Basisstation und diesem neuen Gerät bekannt ist. Demnach kann die Basisstation auf eine geheime und authentifizierte Art mit jedem Gerät kommunizieren.

3.2.3 Kommunikation

Wollen nun zwei Geräte miteinander Informationen austauschen, so wird die Basisstation hierfür als Mittelsmann eingesetzt: Hierfür spricht das erste Gerät das zweite, zu dem es einen Schlüssel aufbauen will, zunächst direkt und unverschlüsselt an. Daraufhin reicht das zweite Gerät diese Anforderung – angereichert mit einer Zufallszahl ("Nounce") zur Überprüfung der Aktualität und mittels eines Secure Hashes (MAC) signiert – an die Basisstation weiter. Diese generiert bei Erhalt der Nachricht einen neuen Schlüssel und schickt diesen - verschlüsselt und signiert – an die beiden Geräte.

Durch dieses Protokoll entsteht ein gemeinsames Geheimnis zwischen jeweils zwei Geräten sowie zusätzlich der Basisstation, die jedoch als vertrauenswürdig eingestuft wird, und demnach selber den Schlüssel nie verwenden oder preisgeben wird.

[1] Message Authentication Code

Broadcasts mittels des μTESLA-Protokolls werden – aufgrund der Ressourcenbeschränkungen – nur von der Basisstation ausgehend unterstützt. Hierfür wird die symmetrische Authentifizierung nach Lamport [15] mittels einer (auf Secure Hashes aufbauenden) Schlüsselkette verwendet. In einer solchen Schlüsselkette wird ein Wert (Schlüssel) in der Kette durch Anwendung eines Secure Hashes auf den nächsten Wert gebildet. Durch dieses Verfahren sind bei Bekanntgabe eines Wertes in der Kette alle vorherigen Werte berechenbar und damit verifizierbar. Der Startschlüssel sowie Informationen zur Zeitsynchronisation werden – für jedes Gerät einmalig – mittels einer Kommunikation per SNEP übertragen..

Benötigen einzelne Geräte eine Broadcastsemantik, so kann dies zum Beispiel durch eine Delegation an die Basisstation geschehen.

Des Weiteren wird ein Routingverfahren vorgestellt, um Nachrichen – in einem großen Netzwerk, wie es bei SNEP angedacht ist – zwischen 2 Geräten zu transportieren. Hierfür wird – ausgehend von der Basisstation – ein Spannbaum mittels Breitensuche nach folgendem Algorithmus erstelt:

Die Basisstation sendet periodisch sogenannte "Beacons" ab, um den Aufbau des Spannbaumes zu starten. Ein solches Beacon enthält eine signierte "Epoche". Dies ist eine laufenden Nummer, die es ermöglicht, unterschiedliche Läufe des Algorithmus zu unterscheiden. Ebenso lassen sich damit Angriffe durch ein Wiederabspielen von Paketen (Replay) verhindern. Neben dieser Epoche enthält das Beacon noch jeweils den Namen des Senders. Empfängt eine Station ein solches Beacon, so geht sie folgendermassen vor:

- Wurde bereits ein Beacon dieser oder einer späteren Epoche empfangen, so wird die Nachricht ignoriert.
- Ansonsten wird das Beacon wiederum gebroadcasted; hierbei wird die eigene Bezeichnung als Sender verwendet. Der Sender des ursprünglich empfangenen Beacons wird danach "Vaterknoten" gemerkt, die empfangene Station bindet sich somit an den Spannbaum unter den ursprünglichen Sender an.

Auf diese Weise entsteht ein Spannbaum, über den Nachrichten geschickt werden können. Da einzelne Geräte nicht als tamper-resistant angesehen werden, wird ein weiterer Mechanismus benötigt, um zu verhindern, dass Angreifer fremde Nachrichten nicht oder fehlerhaft weiterschicken. Hierbei agiert jedes Gerät als "Watchdog" des jeweiligen Vaterknotens: Es überprüft, ob die Nachrichten korrekt weiterleitet werden (aufgrund bidirektionaler Kanäle können die Watchdogs dies erkennen) und ergreift im negativen Fall Maßnahmen, z.B. werden Beacons von diesem Gerät in späteren Epochen ignoriert und damit für Routingaufgaben nicht mehr genutzt.

3.2.4 Ausfallsicherheit

Der Ausfall (also Zerstörung, Manupulation, Diebstahl, Batterieausfall, ...) von Geräten ist bei SPINS explizit vorgesehen. Fällt ein Gerät aus, so wird ab der nächsten Epoche (also dem nächsten Aussenden eines Beacons bei der Basisstation) kein anderes Gerät mehr einen Verweis (z.B. für das Routing) auf dieses Gerät haben. Eine Partition des Netzes ist hierbei nicht vorgesehen, es muss für ein funktionsfähiges Netzwerk also immer ein Pfad zu der Basisstation existieren. Wird ein Gerät kompromittiert, so enthält es nur

Schlüssel für Pfade, in denen dieses Gerät selber eines der Endpunkte ist, es gibt also keinerlei Geheimnisse zwischen noch intakten Geräten preis.

Fällt die Basisstation aus oder wird übernommen, so bedeutet dies – aufgrund des vollständigen Vertrauens der Geräte in diese Station und der zentralen Algorithmen – den Fall des gesamten Netzwerkes.

3.3 Symmetrische, dezentrale Ansätze mit Gruppenschlüssel

Die "*Secure Pebblenets*" [2], ein 2001 als Gemeinschaftsprojekt der Universitäten von Texas und Mailand entwickeltes Design, stellen eine dezentrale Lösung für ein Large Sensor Network dar.

3.3.1 Kurzbeschreibung

Das System bietet innerhalb eines Pebblenets – einer Gruppe von kleinen Geräten (Pebbles) – sichere Kommunikation an. Im Falle von Pebblenets sind dies Vertraulichkeit, Integrität und Verfügbarkeit.

Als kryptographische Methoden werden hierbei einzig ein MAC und ein symmetrischer Verschlüsselungsalgorithmus eingesetzt.

Durch die Verwendung eines einzigen Schlüssels für die Datenkommunikation ist es jedem Pebble möglich, sämtliche Nachrichten – und somit auch Kommunikation zwischen anderen Pebbles – zu decodieren und auch zu manipulieren. Aufgrund der Ressourcenbeschränktheit der einzelnen Geräte war diese Semantik beabsichtigt und als Designkriterium festgelegt. Um dieser Eigenschaft Rechnung zu tragen, wird eine Tamper resistance bei den Geräten verlangt. Des weiteren gelten die Sicherheitseigenschaften nur für Angreifer ausserhalb des Secure Pebblenets.

3.3.2 Netzaufbau

Innerhalb eines Pebblenets wird vorrausgesetzt, das alle Geräte vor ihrem Ausbringen an den Einsatzort bereits mit einem gemeinsamen Geheimnis – einem "Group Identity Key" – versehen sind. Da bei der Entwicklung hauptsächlich von staatlichen Einsatzzwecken (wie beispielsweise der militärischen Nutzung als Sensor- oder Kommunikationsnetzwerk im Kriegsfall oder der zivilen Nutzung zur Verkehrs- oder Wetterüberwachung) ausgegangen wurde, wird eine solche Vorbelegung als realistisch angenommen.

Durch die Verwendung eines einzigen verteilten Geheimnisses wird eine tamper resistance von den einzelnen Geräten verlangt. Es darf einem Angreifer demnach nicht möglich sein, selbst bei physikalischem Zugriff auf das einzelne Gerät an den Schlüssel zu gelangen. Dies soll durch spezielle Hardware erreicht werden [20], welche den Schlüssel bei dem Versuch einer Manipulation zerstört.

Um den Group Identity Key nicht durch zu häufige Verwendung einer Dekodierung mittels Kryptoanalyse zu gefährden, wird er nach jeder Verwendung durch einen neuen - mittels eines Secure Hashes aus dem alten Schlüssel generierten – Schlüssel ersetzt. Durch dieses Verfahren wird zusätzlich die Frische einer Nachricht gewährleistet, da ein Schlüssel nur ein einziges mal verwendet werden kann. Gleichzeitig verlangt dieses Verfahren jedoch, das jedes Gerät für jede Nachricht den Schlüssel modifizieren muss – weshalb

dieser Schlüsselmechanismus auch nur für Kontrollnachrichten verwendet wird. Datennachrichten werden ohne einer Modifikation des Schlüssels gesendet.

Pebbles, welche nachträglich in ein vorhandenes Pebblenet gelangen wollen, erreichen dies, indem sie die "Version" – die Anzahl, wie oft der Gruppenschlüssel inzwischen geändert wurde – des Gruppenschlüssels erfragen und ihren Group Identity Key so oft mittels der Secure Hash Function modifizieren.

3.3.3 Kommunikation

Aufgrund der geringen Leistung der einzelnen Geräte einerseits und dem großflächigen Einsatz andererseits wird bei den Secure Pebblenets nicht von einer einzigen Kollisionsdomäne ausgegangen. Aus diesem Grund bauen die Pebbles ein "Backbonenetz" auf, welches für das Weiterleiten von Nachrichten zuständig ist. Dieses Netz besteht aus einer (minimierten) Teilmenge des Pebblenetz, welche diese Zusatzaufgaben übernimmt. Ein weiterer Zweck des Backbonenetzes ist das periodische Wechseln des TEK's, dem Traffic Encryption Keys, welcher für die Datenverschlüsselung verwendet wird, und – um eine Kryptoanalyse des Schlüssels zu verhindern – innerhalb vorher festgelegter Abstände gewechselt wird.

Der Aufbau des Backbonenetzes geschieht folgendermaßen:
Zunächst wird das Netzwerk in Cluster unterteilt. Dies geschieht, indem jedes einzelne Gerät anhand von folgenden Faktoren einen Gewichtungsfaktor ermittelt:

1. Zustand der Batterien
2. Anzahl und Signalstärke der in Kommunikationsreichweite befindlichen anderen Geräte
3. Mobilität (stationäre Geräte werden als Backbone bevorzugt)
4. Geländespezifischen Eigenschaften (wie z.B. Temperatur oder Lichtstärke)

Ist dieser Gewichtungsfaktorgrößer als der aller in Kommunikationsreichweite befindlichen anderen Geräte, so wird dieser Pebble zu einem sogenannten Clusterhead – und somit auch Teil der Backbonenetzes. Geräte, welche eine solche Entscheidung empfangen, schließen sich automatisch diesem einen Cluster an und nehmen an dem Auswahlverfahren nicht weiter teil. Durch diese Art des Clusteraufbaus sind zwei benachbarte Cluster von den jeweiligen Clusterheads genau zwei oder drei Hops entfernt. Durch ein Routing von Nachrichten über diese Pfade entsteht ein zusammenhängender Graph [6]. Die hierfür notwendige Verschlüsselung der Nachrichten innerhalb des Clusters (Knoten melden ihren Clusterheads angrenzende fremde Cluster) geschieht durch einen weiteren – Clustereigenen – Schlüssel.

Nach dem Aufbau des Backbonenetzes ist eine Teilmenge (im Idealfall nur ein einziges Gerät) dieses Backbones für das periodische Ändern des Traffic Encryption Keys zuständig. Dies geschieht durch die Generierung eines neuen Schlüssels und dessen Multihop Broadcast mittels des Group Encryption Keys (bzw. einer neuen Inkarnation desselben durch Anwendung des Secure Hashes).

3.3.4 Ausfallsicherheit

Durch den optional großflächigen Einsatz der Pebblenets ist der Ausfall einzelner Pebbles möglich, sei es durch Diebstal, dem Versuch einer Modifikation und damit dessen Zerstörung oder "nur" der Entladung der Batterie. Ein solcher Ausfall hat einerseits zur Folge, dass – sollte es sich bei diesem Gerät um einen Clusterhead gehandelt haben – die Cluster ggf. neu aufgebaut werden müssen. Des Weiteren müssen Routinginformationen über diesen Knoten geändert werden und es besteht die Möglichkeit, dass das Netzwerk durch den Ausfall partitioniert wird. Durch die dezentrale Natur des Secure Pebblenets bilden diese daraufhin zwei getrennte Netzwerke, welche bei einer späteren Verbindung (z.B. durch Einbringen neuer Geräte oder durch die Veränderung der Position einzelner Pebbles) durch Anpassen der Group Identity Key-Versionen auf deren Maximum wieder vereinigt werden.

3.4 Verwendung einer zufälligen Schlüsselvorverteilung

Das in "*A Key-Management Scheme for Distributed Sensor Networks*", von Eschenauer und Gligor 2002 vorgestellte Verfahren [10] stellt einen symmetrischen, dezentralen Ansatz dar, der die Probleme durch nur einen einzigen gruppenweiten Schlüssel beseitigt. Wie bei den Pebblenets wird zwar weiterhin die Verwendung von tamper resistant Hardware angenommen, jedoch ist es in diesem Ansatz möglich, dennoch übernommene Geräte ohne einem Neuaufbau des Netzes mit Hilfe eines zentralen Gerätes auszuschließen (Revocation).

3.4.1 Kurzbeschreibung

Dieses System bietet eine sichere (im Sinne von verschlüsselter) Kommunikation an, in der – wie bei den Pebblenets – Authentizität nur die Gruppenzugehörigkeit sichert. Hierfür wird die Technik der zufälligen Schlüsselvorverteilung ("Random Key Predistribution") verwendet: Geräte erhalten – vor dem Ausbringen zu deren Einsatzorten – eine zufällige Schlüsselmenge aus einem vorher generierten Pool und versuchen später, Geräte mit gleichen Schlüsseln zu finden. Diese erstellen danach eine sichere Verbindung und es entsteht ein "Overlay-Netzwerk", in dem Geräte auf eine sichere Weise miteinander kommunizieren. Dies bedeutet jedoch, das ein späterer – für die Funktionalität notwendiger – Zusammenhang des Overlay-Netzwerkes nicht zugesichert werden kann, sondern anhand von gewünschten Wahrscheinlichkeiten (zu Kosten von Ressourcen) nur probabilistisch ist.

3.4.2 Netzaufbau

Vor dem tatsächlichen Netzaufbau werden für den Algorithmus folgende – nachträglich nicht mehr änderbare – Parameter benötigt:

- Die maximale Größe des Netzwerkes n
- Wahrscheinlichkeit für einen korrekt funktionierenden, vollständigen Netzwerkaufbau P_c
- Die Größe des Schlüsselpools P

- Die Anzahl von Schlüsseln pro Gerät k

Die einzelnen Abhängigkeiten werden hier nicht genauer betrachtet, eine Beispielkonfiguration inklusiver aller benötigter Parameter ist in Tabelle 3.1 berechnet.

Maximale Sensorenzahl n	10.000
Verlangte Zusammenhangswahrscheinlichkeit P_c	0.99999
Im Kommunikationsbereich eines Knoten befindliche Nachbarn	40
Größe des Schlüsselpools P	100.000
Benötigte Schlüssel pro Knoten k	200

Tabelle 3.1: Beispielkonfiguration nach Eschenauer und Gligor

Der Netzaufbau wird in 3 Phasen unterteilt:

Schlüssel-Vorverteilung In dieser Phase wird zuerst ein Schlüsselpool mit insgesamt P zufälligen Schlüsseln. Schlüssel bestehen in diesem Ansatz neben dem eigentlichen gemeinsamen Geheimnis über einen Bezeichner, der zur Benennung eines Geheimnisses dient und nicht geheim ist. Danach erhält jedes Gerät eine zufällige Auswahl von k Schlüsseln. Aufgrund der rein zufälligen Auswahl ist hierbei nicht bestimmt, ob ein Schlüssel an genau zwei Geräte verteilt wird; auch eine Ausgabe an keines oder zum Beispiel an drei Geräte ist möglich.

Finden gemeinsamer Schlüssel Nach dem Ausbringen der Geräte zu ihrem Einsatzort und deren Initialisierung versuchen diese, Nachbarn in Kommunikationsreichweite zu finden, welche ebenfalls mindestens einen der Schlüssel besitzen. Dies kann durch Broadcasten der eigenen Schlüsselliste (genauer der Schlüsselbezeichner) oder durch ein Challenge-Response Verfahren geschehen. Besitzen 2 Geräte einen gemeinsamen Schlüssel, so bilden diese eine sichere Verbindung mit eben diesem Schlüssel aus.

Am Ende dieses Schrittes besitzen manche benachbarten (im Sinne des Kommunikationsbereiches) Geräte eine sichere Verbindung, andere noch nicht. Jedoch entstand mit einer Wahrscheinlichkeit P_c ein zusammenhängender Graph, bestehend aus den einzelnen Geräten als Knoten und "sicheren" verschlüsselten Verbindungen als Kanten.

Aufbau vom Pfad-Schlüsseln In dieser letzten Phase bauen benachbarte Gerätepaare ohne gemeinsamen Schlüssel eine Verbindung auf, indem diese einen Schlüssel über einen bereits ausgebildeten Pfad zwischen den beiden Geräten verschicken. Dieser Pfad existiert unter der Annahme eines zusammenhängenden Graphens immer. Hierbei sei vermerkt, das alle auf diesem Pfad liegenden Geräte den Schlüssel mitlesen können. Um die Notwendigkeit einer Schlüsselgenerierung zu umgehen, können die sich verbindenden Geräte einen der noch nicht benutzten Schlüssel in Ihrem Schlüsselspeicher verwenden.

Nach Durchlaufen dieses Verfahrens entsteht mit der vorher gegebenen Wahrscheinlichkeit P_c ein Sensornetzwerk, in welchem jedes Gerät mit allen in Kommunikationsreichweite befindlichen anderen Geräten auf eine – bezüglich der gesamten Gruppe – verschlüsselte und authentifizierte Weise kommunizieren kann.

3.4.3 Kommunikation

Wollen nach dem erfolgten Netzaufbau zwei Geräte miteinander kommunizieren, so verschicken diese die Nachricht – verschlüsselt mit dem jeweils zu dem verwendeten Link zugehörigen Schlüssel – über einen Pfad geroutet zu dem jeweils anderen Gerät. Die Nachricht ist demnach – ebenso wie der Schlüsselaufbau über Pfade – von allen zwischenliegenden Geräten lesbar.

3.4.4 Ausfallsicherheit

Der Ausfall von Knoten lässt sich hier in drei unterschiedlich behandelte Bereiche unterteilen:

Zerstörung, Batterieausfall, Defekte Da bei dem Aufbau des Netzes von beispielsweise durchschnittlich 20 anderen Geräten in der eigenen Kommunikationsreichweite ausgegangen wird und nach dem Netzaufbau auch eine sichere Verbindung zu jedem dieser anderen Geräte existiert, betrifft der Ausfall eines einzelnen Gerätes nicht den Zusammenhang des Netzes. Des Weiteren ist es z.B. möglich, neue Geräte an dieser Stelle auszubringen, sollte die Sensorendichte zu gering werden.

Manipulation von Sensordaten Das Feststellen von fehlerhaften Sensordaten durch manipulierte Sensoren an den einzelnen Geräten kann durch eine (ggf. Offline-) Analyse der Daten und deren Korrelation mit benachbarten und historischen Sensordaten geschehen.

Kompromittierung eines Geräts Sollte es einem Angreifer – trotz der verlangten Tamper resistance – gelungen sein, ein einzelnes Gerät zu übernehmen, so kann dieses Gerät aus dem Netzwerk ausgeschlossen werden. Dies geschieht, in dem eine (ggf. mobile) Basisstation, welche für jedes einzelne Gerät ein zusätzliches Schlüsselpaar besitzt – und somit authentifiziert mit allen anderen Geräten kommunizieren kann – eine Liste aller auf dem kompromittierten Gerät befindlichen Schlüssel verschickt. Daraufhin entfernen alle Geräte diese Schlüssel aus ihrem Schlüsselspeicher und das übernommene Gerät wird somit von dem Rest des Netzwerkes isoliert. Durch die zufällige Verteilung der Schlüssel ist hierbei jedoch nicht gesichert, dass dadurch nicht auch andere Geräte aus dem Netzwerk ausgeschlossen werden.

3.5 Erweiterungen der zufälligen Schlüsselvorverteilung

"*Random Key Predistribution Schemes for Sensor Networks*", von H. Chan, A. Perrig (seinerseits auch Hauptautor von [21]) und D. Song von der Carnegie Mellon University 2003 veröffentlicht [5], behandelt hauptsächlich drei mögliche Verbesserungen zu dem Lösungsansatz von Eschenauer und Gligor [10] (siehe Kapitel 3.4).

3.5.1 Kurzbeschreibung

Chan et al. präsentieren Methoden, mit denen – bei Kompromittierung einer bestimmten Anzahl von Geräten – möglichst wenig andere Geräte bzw. Verbindungen kompromittiert werden. Hierbei stellen die Autoren zunächst ein q-zusammengesetztes zufälliges

Schlüsselvorverteilungsverfahren ("q-composite random key predistribution scheme") vor, in dem – im Gegensatz zu Eschenauer und Gligor – nicht einzelne gemeinsame Schlüssel, sondern gemeinsame Schlüsselmengen verwendet werden. Es reicht demnach nicht der gemeinsame Besitz eines gleichen Schlüssels für den Aufbau einer sicheren Verbindung, sondern es müssen mindestens q gemeinsame Schlüssel übereinstimmen. In der Schlüsselerneuerung über mehrere Pfade ("Multipath Key Reinforcement") werden Schlüssel – nach ihrem ursprünglichen Aufbau, wie bei Eschenauer und Gligor beschrieben – durch das Einbeziehen mehrerer Pfade zwischen den Geräten gefestigt (siehe Kapitel 3.5.3). Im letzten (nach Ansicht der Autoren vielversprechendsten) Ansatz, dem zufälligen paarweisen Verfahren ("Random-pairwise keys scheme"), werden Schlüssel in der Phase der Schlüsselverteilung immer paarweise, also genau zwei mal an Geräte vergeben und damit eine Authentifizierung zwischen diesen zwei Geräten erreicht.

Im Folgenden werden die einzelnen Ansätze genauer beschrieben.

3.5.2 q-zusammengesetztes zufälliges Schlüsselvorverteilungsverfahren

Bei diesem Verfahren werden Schlüsselpool des Netzwerkes und Schlüsselmengengrösse der einzelnen Geräte so gewählt, dass nicht nur – wie in [10] beschrieben – ein einziger Schlüssel mit einem Nachbarn übereinstimmt, sondern das es eine Mindestmenge von übereinstimmenden Schlüsseln mit Größe q geben muss, um eine Verbindung zwischen zwei Geräten auszubilden. Existieren sogar mehr übereinstimmende Schlüssel, so wird diese größere Menge – also die Schnittmenge der beiden Schlüsselmengen – gewählt.

Berechnungen der Autoren ergaben, dass mit den Parametern $q = 2$, $k = 200$ und $P_c = 0.5$ im Gegensatz zu 30 Geräten bei Eschenauer und Gligor nun mindestens 45 Geräte übernommen werden müssen, so das eine Verbindung zwischen zwei unbeteiligten Geräten mit einer Wahrscheinlichkeit von 0.1 abgehört werden kann.

Während bei diesem Verfahren bei einer geringen Zahl von kompromitierten Knoten einem Angreiffer nur eine sehr gerigere weitere Menge an Informationen zugänglich wird, so erhält der Angreiffer bei einer hohen Anzahl von übernommenen Geräten überproportional mehr Informationen (im Gegensatz zu dem ursprünglichen Schema, bei dem dies ein konstanter Zuwachs pro Gerät ist).

3.5.3 Schlüsselerneuerung über mehrere Pfade

Dieses Verfahren ist eine weitere – optional auch zusätzliche – Methode, die Sicherheit der Verbindungen zu erhöhen. Die Motivation des Verfahrens ist Folgende: Zwei Gerät haben ihre Verbindung durch eine Menge von Schlüsseln gesichert, welche sich ebenfalls in einer Vereinigungsmenge von anderen Geräten befindet. Demnach reicht es aus, diese möglicherweise sehr kleine Menge von anderen Geräten zu kompromittieren, um diese Verbindung abhören zu können. Für das Reinforcement bestimmt nun jedes Gerätepaar, welches eine sichere Verbindung besitzt, sämtliche disjunkten Pfade zwischen sich mit einer gegebenen maximalen Länge h. Daraufhin werden auf jedem Pfad ein zufällig generierter Schlüssel ausgetauscht und all diese Pfadschlüssel mit dem ursprünglichen Schlüssel mittels XOR-Operationen verbunden.

Um die dabei gefestigte Verbindung zu übernehmen, muss ein Angreiffer nicht nur den Schlüssel für die ursprüngliche Verbindung besitzen, sondern zusätzlich auch je eine

Verbindung pro Pfad übernommen sowie den darüber ausgetauschten Schlüssel abgehört haben.

3.5.4 Zufälliges paarweises Verfahren

Bei allen bisher verwendeten Verfahren kann auf Grund der zufälligen Schlüsselauswahl nicht sichergestellt werden, dass es nicht ein drittes Gerät mit genau den Schlüsseln gibt, mit welchen zwei Geräte eine Verbindung aufgrund ihrer gemeinsamen Schlüssel aufgebaut haben. Demnach ist eine Authentifizierung einzelner Knoten nicht möglich.

Aus diesem Grund werden in der Phase der "Schlüssel-Vorverteilung" einzelne Schlüssel jeweils an genau zwei Geräte ausgegeben. Damit ist – wenn ein Gerät einen Partner mit einem gleichen Schlüssel findet – die Identität des jeweiligen Kommunikationspartner eindeutig bestimmt.

Die dritte Phase des Netzaufbaus, dem "Aufbau von Pfad-Schlüsseln", entfällt in diesem Verfahren, da – durch das Verbinden von Geräten ohne paarweisen Schlüssel – die Eigenschaft der Authentizität verloren ginge.

Da mittels dieses Verfahrens und insbesondere durch das Fehlen von Pfad-Schlüsseln das resultierende Netzwerk um vieles schwächer verbunden ist, schlagen die Autoren – insbesondere wenn ein zusammenhängendes Netzwerk nicht hergestellt werden konnte – die Technik der "Multi-hop range extension" vor. hierbei dienen Geräte, welche eine Nachricht empfangen, als Repeater und erhöhen damit die Sendereichweite der Geräte. Ein solches Verfahren hat jedoch die höhere Verwundbarkeit bei DoS-Angriffen zur Folge, da das Netzwerk hierbei die Angriffe selbst verstärkt.

Das entwickelte Verfahren hat folgende Eigenschaften:

- Übernahme eines Gerätes gibt keinerlei weitere Verbindungsinformationen preis.
- Geräte kommunizieren direkt nur auf eine authentifizierte Weise.

Auf Grund dieser Eigenschaften lassen sich weitere Technologien einsetzen:

Verteiles Zurückziehen von Geräten (Revocation) ohne Basisstation Da sämtliche Nachrichten, welche ein Gerät absendet, von genau einem der direkt verbundenen Geräte gelesen und überprüft werden können, ist es möglich, dass diese gemeinsam für das Entfernen der Knoten verantwortlich sind. Um dies zu gewährleisten, können die jeweiligen Besitzer der zu einem Gerät zugeordneten Schlüssel jeweils eine – auf Grund von Schwellwertkryptographie erzeugte – Stimme erhalten und diese im Falle eines erkannten Fehlverhaltens broadcasten. Empfangen sämtliche verbundenen Nachbarn genügend Stimmen, so können diese gemeinsam das sich fehlerhaft verhaltende Gerät isolieren.

Verhindern von großflächiger Knotenreplikation und -generierung Um das (ggf. virtuelle) Einfügen beliebig vieler Knoten zu verhindern, lässt sich die Eigenschaft des Netzwerkes nutzen, dass die Anzahl der Verbindungen pro Gerät anhand einer Wahrscheinlichkeitsverteilung festgelegt ist. Somit lässt sich ein Schwellwert angeben, bei dessen Überschreitung von einer bösartigen Knotengenerierung ausgegangen werden kann.

3.6 Verwendung paarweiser Schlüssel

S. Zhu et al. stellten Anfang 2003 ihren Ansatz "*Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach*" für große Sensor Netzwerke vor [31]. Dabei handelt es sich ebenfalls um eine dezentrale, symmetrische Lösung.

3.6.1 Kurzbeschreibung

Das Verfahren verwendet – wie auch die vorherigen – eine Random Predistribution, in welcher Schlüssel zufällig verteilt werden. Im Gegensatz zu den anderen Ansätzen erstellen Zhu et al. darauf aufbauend ein weiteres Netz zur sicheren Kommunikation zwischen zwei Endgeräten. Es wird zwar, aufgrund der zufälligen Anfangsverteilung, keine echte Authentizität erreicht, jedoch lässt sich als Parameter beim Netzaufbau die gewünschte Sicherheit angeben, mit der eine Verbindung nicht – auf Grund von kompromittierten Knoten - abgehört werden kann.

3.6.2 Netzaufbau

Der Netzaufbau geschieht – wie bei Eschenauer und Gligor beschrieben – zuerst durch eine Random Preconfiguration: Aus einem Schlüsselpool erhält jedes Gerät eine kleine Menge an Schlüsseln. Hierbei ist nicht vorgegeben, wie oft ein Schlüssel ausgewählt wird, er darf also öfters als zwei mal vergeben werden. Im Gegensatz zu allen vorherigen Methoden werden hierbei die Schlüssel jedoch nicht streng zufällig, sondern an Hand eines Pseudozufallgenerators in Abhängigkeit der Gerätekennung vergeben. Auf diese Weise ist es jedem Gerät möglich, ohne jeglicher Kommunikation die Schlüsselbezeichner eines anderen Knotens – und damit auch die Schnittmenge zwischen zwei beliebigen Knoten – herauszufinden.

Das Ergebnis dieses Netzaufbaus ist ein mit einer vorgegebenen Wahrscheinlichkeit verbundenes Netzwerk, in dem ein Gerät mit einer bestimmten Menge anderer Geräte kommunizieren kann.

3.6.3 Kommunikation

Verbindungen, wie sie nach dem Netzaufbau entstehen, sind für einen außen stehenden Angreifer verschlüsselt.

Dennoch ist es möglich, dass die Kommunikation zwischen zwei Geräten durch die Kompromittierung eines dritten Gerätes (oder einer Menge von Geräten) – welches die gleichen Schlüssel besitzt wie die Schnittmenge der beiden kommunizierenden – abgehört oder manipuliert wird. Aus diesem Grund müssen Geräte, welche untereinander Daten austauschen wollen, einen "Pairwise key" – also einen Schlüssel, den nur diese beiden Geräte kennen - bilden.

Hierfür sucht das initiierende Gerät eine Menge von Pfaden zwischen sich und dem Zielgerät. Die Länge dieser Pfade ist nicht vorgegeben, Zhu et al. betrachten jedoch aufgrund des erhöhten Kommunikationsaufwandes sowie der verringerten Sicherheit keine Pfadlängen grösser zwei (sogenannte "multi-proxy paths"). Jedoch werden Pfade mit nur einem Proxy betrachtet, welche auf Grund der begrenzten Kommunikationsreichweite von zwischenliegenden Geräten verstärkt werden müssen ("multi-hop-proxies").

Ziel dieser Pfadsuche ist es, den zu verteilenden "Pairwise key" in mehrere Stücke zu teilen und jeweils über einen Pfad zu schicken. Das Zielgerät fügt letztendlich alle Stücke (mittels einer XOR-Operation) zusammen und überprüft die neu aufgebaute Verbindung. Die Sicherheit der neu entstandenen Verbindung lässt sich direkt von der Anzahl der benötigten Schlüssel ableiten, um *alle* Pfade – an zumindest einer Stelle – zu kompromittieren. Auf Grund der Pseudozufälligkeit bei der Schlüsselerzeugung kann ein Knoten, bevor er mit dem Schlüsselaufbau beginnt, überprüfen, ob eine Pfadkonfiguration existiert, mittels der sich die benötigte Sicherheit erreichen lässt.

3.6.4 Ausfallsicherheit

Die Ausfallsicherheit – ein in [31] nicht behandeltes Gebiet – betrifft drei Bereiche:

- Das Netz muss physikalisch so dicht bleiben, dass es nicht auf Grund eines zu geringen Kommunikationsradius der einzelnen Geräte partitioniert wird.
- Es müssen immer ausreichend Geräte vorhanden sein, damit für den Aufbau eines Pairwise keys eine genügend grosse Pfadmenge gefunden werden kann.
- Die Anzahl der kompromittierten Geräte erhöht die Wahrscheinlichkeit einer Kontrolle beliebiger anderer Verbindungen. Hierbei sei angemerkt, dass auch in diesem Ansatz bei einer ungünstigen Preconfiguration die Übernahme eines einzigen Gerätes ausreichen kann, um einen von diesem Gerät unabhängigen "Pairwise key" dekodieren zu können.

3.7 Zusammenfassung und Auswertung der Ansätze

Die in den letzten Kapiteln behandelten Ansätze lassen sich, aufgrund ihrer Eigenschaften, in den Tabellen 3.2 und 3.3 charakterisieren. Zusätzlich sind als letztes die Anforderungen aufgeführt, wie sie teilweise bereits für das hier entstehende Design eines Heimautomatisierungsnetzes in Kapitel 1.5 angeführt wurden. Hierbei haben die einzelnen Zeilen folgende Bedeutung:

Kryptographie Welche Kategorie von Kryptographie wird verwendet?

Aufbau Werden zentrale Server im Betrieb verwendet? (Anm.: Die Verwendung eines zentralen Gerätes vor Installation des Netzwerkes wird hier nicht betrachtet)

Authentizität Lässt sich bei einer Kommunikation der Kommunikationspartner (sicher) einer Identität zuordnen, oder ist dadurch nur dessen Gruppenmitgliedschaft zugesichert?

Beliebiges Hinzufügen neuer Geräte Lassen sich nachträglich beliebig viele Geräte in das Netzwerk einbringen oder ist z.B. die Maximalgröße des Netzwerkes bereits vorgegeben?

Tamper resistance Wird tamper resistant Hardware verlangt?

Zur Übernahme benötigte Geräte Wie viele Geräte werden benötigt, um das Netz oder einzelne – von der Übernahme nicht betroffene Verbindungen – zu kompromittieren?

	The Quest, Kap. 3.1	SPINS, Kap. 3.2	Pebblenets, Kap. 3.3
Kryptographie	Asymmetrisch	Symmetrisch	Symmetrisch
Aufbau	Dezentral	Zentral	Dezentral
Authentizität	Jeder Knoten, Probabilistisch	Jeder Knoten	Nur Gruppenzugehörigkeit
Beliebiges Hinzufügen neuer Geräte	Ja	Ja	Ja
Vorkonfiguration der Geräte	Nein	Ja	Ja
Tamper resistance	verlangt	optional	verlangt
Zur Übernahme benötigte Geräte	ein beliebiges bzw. festlegbar	Basisstation	ein beliebiges

Tabelle 3.2: Zusammenfassung der Ansätze, Teil 1

	Random Predistribution, Kap. 3.4 und 3.5	Pair-wise Keys, Kapitel 3.6	Anforderungen, Kap. 1.5
Kryptographie	Symmetrisch	Symmetrisch	Symmetrisch
Aufbau	Dezentral	Dezentral	Dezentral
Authentizität	Gruppenzugehörigkeit, teilweise Knoten	Gruppenzugehörigkeit, Probabilistisch	Jeder Knoten
Beliebiges Hinzufügen neuer Geräte	Nein	nein	Ja
Vorkonfiguration der Geräte	Ja	Ja	Nein
Tamper resistance	verlangt	optional	optional
Zur Übernahme benötigte Geräte	Probabilistisch	Probabilistisch	festlegbar

Tabelle 3.3: Zusammenfassung der Ansätze, Teil 2

Letztendlich lässt sich erkennen, dass keiner der existierenden Ansätze mit den gegebenen Anforderungen vollständig korrespondiert.

Kapitel 4

Verfahren zur Verteilung von Schlüsseln

In diesem Kapitel wird das grundlegende Verfahren, auf dem diese Arbeit basiert und welches in der während dieser Arbeit entstandenen Veröffentlichung "*A Key-Distribution Scheme for Wireless Home Automation Networks*" [25] detailliert beschrieben ist, vorgestellt.

4.1 Allgemeine Beschreibung

Da symmetrische Schlüssel verwendet werden sollen, muss – um für die jeweils sendenden Geräte authentifizierte Kommunikation auch in einem nicht synchronen Netzwerk zu ermöglichen – zwischen *jedem* miteinander kommunizierenden Gerätepaar ein gemeinsamer Schlüssel gebildet werden. Dies wird auch als Shared bzw. Common Secret bezeichnet.

Stajano und Anderson beschrieben, dass die einzige wirklich sichere Art, einen gemeinsamen Schlüssel aufzubauen, nur das physikalische Verbinden der beiden Geräte sein kann [24].

In grossen Netzwerken ist es jedoch nicht praktikabel, jedes neue Gerät bei der Installation mit jedem anderen zu verbinden. Des Weiteren besitzen (nach den in Kapitel 1.5 beschriebenen Anforderungen) die einzelnen Geräte so wenig Ressourcen, dass es nicht möglich ist, zu jedem anderen Gerät einen Schlüssel zu speichern.

Aus diesem Grund ist der Aufbau eines Schlüssels auf folgende zwei Arten möglich:

1. Austausch während einer physikalischen Verbindung über diesen sicheren Kanal, sogenanntes *Imprinten*. Dies wird in Kapitel 4.2 genauer beschrieben.

2. Aufbau eines neuen Schlüssels über mehrere Pfade im Netzwerk. Dieses Verfahren wird in Kapitel 4.3 beschrieben.

Um Sicherheit in einem Netzwerk zu definieren, wurde in diesem Ansatz ein Sicherheitsfaktor s, auch "Security level", mit folgender Eigenschaft eingeführt:
Solange weniger als s Geräte innerhalb des Netzwerkes gleichzeitig kompromittiert wurden, ist es einem Angreifer nicht möglich, (neu) aufgebaute Schlüssel zwischen nicht kompromittieren Geräten abzufangen oder zu ermitteln.

Das bedeutet, dass ein Angreifer mindestens s Geräte kompromittieren muss, um irgendeinen Erfolg zu erreichen. Da der Sicherheitsfaktor s frei wählbar ist, lässt er sich entsprechend der gegebenen Erwartungen bezüglich der Sicherheit bestimmen. Ein höherer Sicherheitsfaktor bedingt jedoch einen linear höheren Aufwand bei einer dezentralen Einbringung des Gerätes.

Ein zentrales Verfahren mit konstantem Aufwand, jedoch einem Single Point of Trust, ist in Kapitel 6.3.2 beschrieben.

Die weitere Beschreibung des Verfahrens ist in zwei Teile geteilt: Erstens der Aufbau eines Netzes mittels physischen Kontaktes, zweitens die Erstellung von paarweisen Geheimnissen – Schlüsseln zwischen Geräten, die selbst bei Vorhandensein von Angreifern nur diesen Beiden bekannt sind.

4.2 Aufbau des Netzwerkes

Da – wie in den Szenarien für die Heimautomatisierung beschrieben – Geräte inkrementell dem Netzwerk hinzugefügt werden, wird auch hier ein solches Verfahren beschrieben. Die Gesamtgrösse des Netzwerkes n wächst damit bei jedem neu eingebrachten Gerät um eins. Wann immer ein Gerät hinzukommt, so muss es an genau s andere Geräte, welche sich bereits im Netz befinden, angeschlossen werden und mit diesen einen Schlüssel austauschen. Dieser Vorgang wird als "imprinten" bezeichnet. Sollten noch keine s Geräte im Netzwerk existieren (also $n < s$), so wird das neue Gerät an alle vorhandenen angeschlossen.

Ein Beispiel für einen solchen Netzwerkaufbau ist in Abbildung 4.1 illustriert. In Bild (a) befindet sich nur ein einziges Gerät A im Netzwerk. Dieses hat noch keinerlei Schlüssel gespeichert. Kommt – wie in Bild (b) beschrieben – ein neues Gerät hinzu, so wird es an A angeschlossen, und die beiden Geräte generieren einen gemeinsamen Schlüssel k_{AB}. Das Beispiel zeigt ein Netzwerk mit $s = 3$. Demnach führt das Einfügen der Geräte C und D zu einer Beibehaltung der Vollvernetzung. Wird – wie in Bild (e) dargestellt – bei einer Netzwerkgrösse von $n \geq s + 1$ ein weiteres Gerät E hinzugefügt, so können beliebig s bereits vorhandene Geräte (hier A, C und D) gewählt werden und durch imprinten mit E ein gemeinsames Geheimnis austauschen.

Auf diese Weise entsteht zuerst für die ersten $(s+1)$ Geräte ein vollvernetztes Netzwerk. In einer Graphendarstellung, in dem die einzelnen Geräte die Knoten und gemeinsame Geheimnisse die Kanten repräsentieren, ist dieser ebenfalls vollvernetzt. Für grosse n entstehen auf diese Weise jedoch nur $(s \cdot n)$ Kanten, was dazu führt, dass (um ein Gerät in ein Netzwerk einzubringen) durchschnittlich nur s neue Schlüssel erstellt und damit $(2 \cdot s)$ Schlüssel pro Gerät gespeichert werden müssen. Diese Grösse resultiert aus der Tatsache, dass jeder Schlüssel immer genau auf zwei Geräten gespeichert werden muss und so insgesamt $2 \cdot n \cdot s$ Speicher im gesamten Netzwerk benötigt wird.

Diese beim Imprinten entstandenen Schlüssel haben hierbei gegenüber später erstellten neuen Schlüsseln einen besonderen Status und dürfen nicht gelöscht werden. Ein Verfahren, wie mit solchen Schlüsseln bei einem Ausfall von Geräten verfahren werden kann, ist in Kapitel 6.6.4 beschrieben.

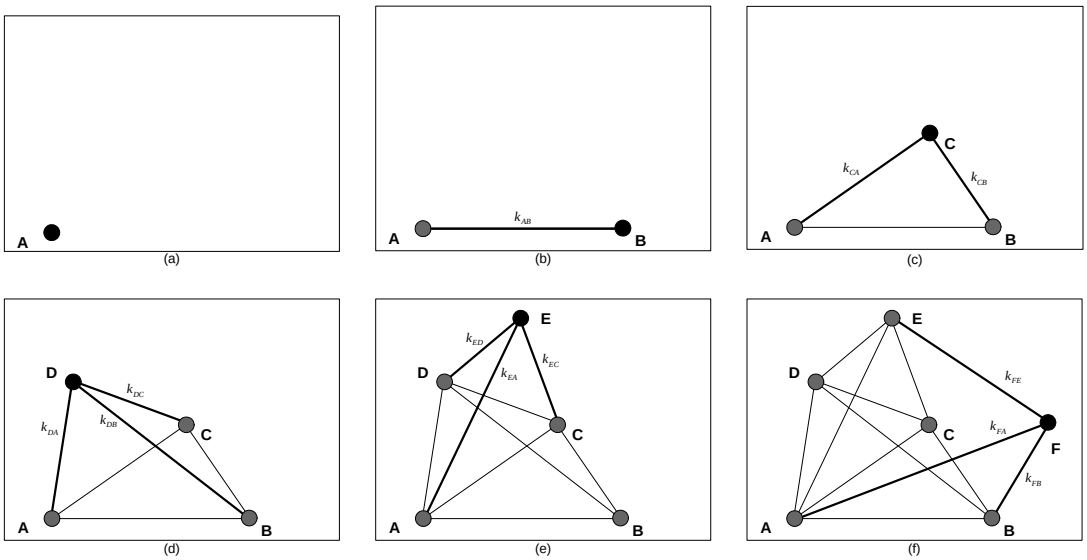

Abbildung 4.1: Aufbau des Netzwerkes (Quelle: [25])

4.3 Verbindungserzeugung

Nach dem Aufbau des Netzwerkes sind zwar einige Geräte miteinander verbunden, jedoch existieren (für $n >> s$) auch viele Gerätepaare, welche über keinen gemeinsamen Schlüssel verfügen, bei denen aber möglicherweise ein Kommunikationsbedarf besteht. In diesem Fall muss zwischen solchen ein neues gemeinsames Geheimnis erstellt werden. Im Gegensatz zum imprinten ist es hier jedoch nicht möglich, eine physikalische Verbindung auszunutzen, da dies bei dem Administrator des Netzes zu einem enormen und unkoordinierten Arbeitsaufwand führen würde. Im Folgenden wird beschrieben, wie ein solcher Schlüssel erstellt werden kann – ohne dass ein Angreifer, der nicht mindestens s Geräte kompromittiert hat, dabei Informationen über das erzeugte gemeinsame Geheimnis gewinnen kann.

Ausgangsbasis hierfür ist das durch das Imprinten entstandene Netzwerk. Durch den besonderen Aufbau mit s neuen Kanten pro entstehenden Knoten an bereits eingebrachten Knoten ist ein Graph entstanden, welcher die Eigenschaft des s-fachen Zusammenhaltes besitzt. Ein Beweis für diese Eigenschaft ist in [25] enthalten. Nach Mengers Theorem [17] bedeutet dies, dass zwischen zwei beliebigen Knoten, welche noch nicht direkt verbunden sind, insgesamt immer s knotendisjunkte Pfade existieren.

Ein neuer Schlüssel wird damit folgendermassen erstellt:

Das initiierende Gerät sucht insgesamt s Pfade über bereits existierende Verbindungen zu dem Zielgerät, bei denen insgesamt kein Gerät mehr als einmal vorkommt. Diese existieren – nach Menger – in einem nach Kapitel 4.2 aufgebauten Netzwerk immer. Danach erstellt das initiierende Gerät insgesamt s neue Schlüssel und verschickt diese über je einen der Pfade. Letztendlich besitzen danach beide Geräte diese s Teilschlüssel ("key shares") und können sie – durch bitweises XOR – zu einem gemeinsamen Schlüssel verbinden. Dieser neu entstandene Schlüssel dient von da an als neues gemeinsames Geheimnis.

Ein Beispiel hierfür (ebenfalls für $s = 3$) ist in Abbildung 4.2 dargestellt. A ermittelt hierbei zuerst drei (aufgrund des s-fachen Zusammenhaltes zwingend existierenden) knotendisjunkte Pfade zwischen sich selbst und B. Danach generiert A insgesamt drei Schlüssel $K_{AB_1}, K_{AB_2}, K_{AB_3}$ und schickt jeden von ihnen zusammen mit der jeweiligen Pfadinformationen an die ausgewählten Nachbarn. Diese leiten das Paket bis zum letztendlichen Ziel B weiter. Danach besitzen A wie B die Schlüsselanteile und können $K_{AB} = (K_{AB_1} \oplus K_{AB_2} \oplus K_{AB_3})$ berechnen und als Geheimnis abspeichern.

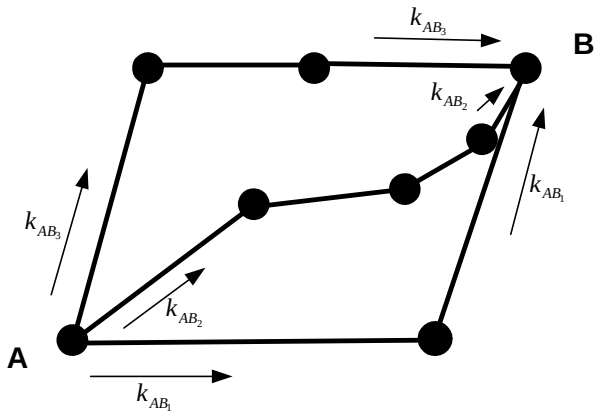

Abbildung 4.2: Aufbau einer neuen Verbindung (Quelle: [25])

Angreifer können bei diesem Austausch genau dann den resultierenden Schlüssel auslesen, wenn sie für jeden Pfad mindestens ein Gerät kompromitiert haben. Besitzt ein Angreifer jedoch nur die Schlüsselinformationen von weniger als s Geräten, so kann er keinerlei Vermutungen über den resultierenden Schlüssel anstellen.

Kapitel 5

Schichtentwurf

Während in Kapitel 6 die einzelnen – für die Funktion der Sicherheitsschicht notwendigen – Protokolle und Kommunikationswege analysiert werden, wird in diesem Kapitel zuerst der grundsätzliche Aufbau der Sicherheitsschicht untersucht.

Dabei wird, neben einer Aufstellung der benötigten und angebotenen Dienstzugangspunkte (Service Access Points), die interne Architektur der Sicherheitsschicht mittels einer weiteren Schichtung dargestellt, um funktionale Gruppen soweit wie möglich zu trennen und somit die Gesamtkomplexität des Systems zu minimieren

5.1 Angebotene Dienste

Die Sicherheitsschicht muss einer aufsetzenden Anwendung folgende Dienste anbieten:

Abfrage über den Zustand einer Verbindung. Obere Schichten müssen, um den Aufwand (und die benötigte Dauer) einer Kommunikation einzuschätzen zu können, ermitteln können, ob eine Verbindung zu einem bestimmten Gerät existiert.

Erstellung einer sicheren Verbindung. Bei Übergabe einer Geräte-ID müssen obere Schichten den Aufbau einer Verbindung zur späteren Kommunikation beeinflussen und initiieren können.

Senden von Nachrichten über bestehende Verbindungen. Besteht eine Verbindung, so kann über diese auf eine sichere Weise (siehe Kapitel 2.1) eine Nachricht übermittelt werden.

Senden von Gruppennachrichten. Gruppeninterne Rundsendungen mittels einer Nachricht (zu Kosten von Sicherheitsaspekten wie der Knoten-zu-Knoten-Authentizität) sind – zum Beispiel zum Senden von Zustandsinformationen – ebenso möglich.

Empfang von Nachrichten. Korrekt versendete Nachrichten (Unicast wie Broadcast) müssen an die oberen Schichten weitergegeben werden.

Neben diesen Diensten existieren noch weitere, optionale, Dienste:

Senden von Nachrichten ohne vorheriger Verbindungsüberprüfung. Um der Semantik einer gewöhnlichen Kommunikationsschicht so nahe wie möglich zu kommen, ist die

Existenz eines Dienstes, welcher bei einer bestehenden Verbindung das Datenpaket sofort sendet und ansonsten zuerst versucht, eine Verbindung aufzubauen und das Paket danach zu senden, sinnvoll. Das Ergebnis der Operation ist genau dann positiv, wenn das Paket versendet werden konnte.

Konfiguration von Verbindungen. Im laufenden Betrieb können Verbindungen, welche erzeugt wurden, auch wieder verworfen werden. Die Notwendigkeit eines solchen Verfahrens wird in Kapitel 6.4.1 beschrieben. Unter diesem Aspekt könnten obere Schichten Verbindungen gemäss ihrer Wichtigkeit und der Kommunikationsfrequenz deklarieren. Beispielsweise könnte eine Verbindung von einem Feuermelder zu einer Sirene als notwendig und nicht löschbar deklariert werden, um eine unnötige Verzögerung im Feuerfall zu vermeiden.

Meldung über existierende Geräte. Da – auf Grund der Ressourcenbeschränkugen – keine komplette Liste aller Geräte im Netzwerk existieren kann, ist es möglich, den Discoverymechanismus, so er von der Sicherheitsschicht implementiert wird, nach oben weiterzugeben. Ein solches Verfahren wird unter anderem in Kapitel 6.4.1 benötigt und entworfen.

Administation der Sicherheitsschicht. Obere Schichten können Verbindungen löschen, Parameter setzen oder die Schicht neu initialisieren.

5.2 Benötigte Dienste

An eine Kommunikationsschicht werden in dieser Arbeit folgende Anforderungen gestellt:

Senden von beliebigen Paketen an alle erreichbaren Geräte (Broadcast). Die Paketgrösse wird u.a. bei der prototypischen Implementierung auf eine für derart kleine Geräte sinnvolle Maximalgrösse begrenzt. Die Kommunikation selbst kann unzuverlässig ("unreliable") sein.

Empfang von Paketen, welche ein anderes Gerät in Kommunikationsreichweite gesendet hat.

Die Unicast-Kommunikation darf zwar unzuverlässig sein, jedoch wird im Folgenden für diese Kommunikationsart eine sehr hohe Empfangswahrscheinlichkeit (beispielsweise von über 95 %) angenommen. Ist dies nicht der Fall, so muss *innerhalb der Sichherheitsschicht* zusätzlich ein Mechanismus zum Bestätigen und wiederholten Senden von Nachrichten integriert werden.

5.3 Resultierender Schichtenaufbau

Abbildung 5.1 zeigt den aus den Anforderungen resultierenden Aufbau der Sicherheitsschicht.

Übereinanderliegende Komponenten symbolisieren hierbei eine benötigte Kommunikation untereinander. Die einzelnen Komponenten und Ihre Aufgaben werden hierbei im Folgenden genauer erläutert.

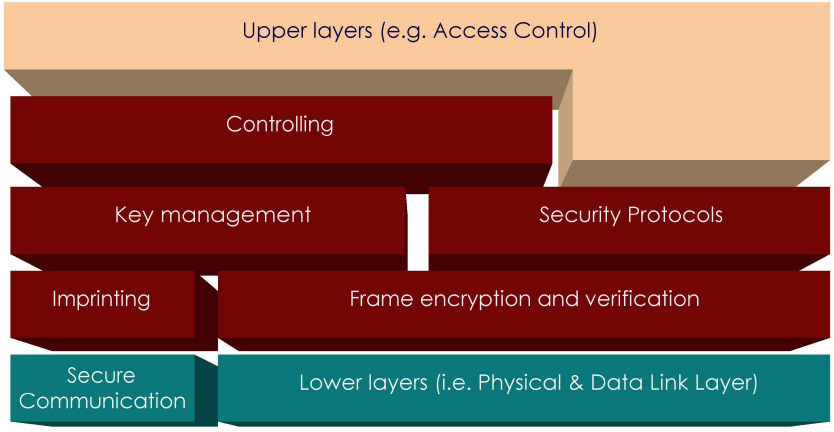

Abbildung 5.1: Aufbau der Sicherheitsschicht

5.3.1 Verschlüsselungsschicht (Frame Encryption and Verification)

Direkt auf der Kommunikationsschicht aufbauend sitzt die Komponente zum Ver- und Entschlüsseln. Bei ausgehenden Paketen hat sie die folgenden Aufgaben:

1. Aus dem Schlüsselmanagement den der Verbindung zugehörigen Schlüssel abfragen. Bei Gruppenkommunikation ist dies der Gruppenschlüssel. Danach werden die zugehörigen Paketnummern zur Zusicherung der Frische abgefragt.
2. Bei fehlenden Schlüsseln einen Fehler signalisieren und abbrechen.
3. Zusammensetzen eines Sicherheitspaketes (siehe Kapitel 5.4).
4. Checksummen berechnen.
5. Paket verschlüsseln.
6. Verwendete Paketnummer im Schlüsselmanagement aktualisieren.
7. Paket an Kommunikationsschicht übergeben.

Bei einem Empfang verhält sie sich dementsprechend:

1. Aus dem Schlüsselmanagement zugehörige Schlüssel und Paketnummern abfragen.
2. Bei fehlenden Schlüsseln abbrechen.
3. Paket entschlüsseln.
4. Checksummen sowie Paketnummer überprüfen.
5. Bei Fehlern abbrechen.
6. Verwendete Paketnummer im Schlüsselmanagement aktualisieren.
7. Paket an Sicherheitsprotokollschicht übergeben.

5.3.2 Imprinten (Imprinting)

Parallel zur Funkkommunikation existiert die Möglichkeit einer Kommunikation über ein sicheres physikalisches Medium zum direkten Schlüsselaustausch. Ein entsprechendes Protokoll wird in Kapitel 6.2 entwickelt. Die Kommunikation an sich wird weder verschlüsselt noch sonst in irgendeiner Weise geschützt und kann beispielsweise über die serielle Schnittstelle, wie sie auf den meisten Microcontrollern existiert, realisiert werden.

5.3.3 Schlüsselmanagement (Key Management)

Das Schlüsselmanagement kapselt eine Datenbank zur Verwaltung von Schlüsseln und weiteren Verbindungseigenschaften. Dabei ist sie für Operationen zum Erstellen, Finden, Auslesen, Ändern und Löschen von Einträgen, sowie auch auf eine Speichereffizienz und eine Auslagerung im EEPROM verantwortlich.

Eine Liste aller benötigten Informationen ist in Tabelle 5.1 ersichtlich. Daraus resultiert ein Platzsaufwand von circa 25 Byte pro existierender Verbindung.

Beschreibung	Typ	Grösse
Zugehöriger Gerätename	Geräte-ID	16 Bit
Gemeinsames Geheimnis	Schlüssel	128 Bit
Verbindungsart (Imprintet, Erzeugt, ...) und -richtung	Typ	8 Bit
Anzahl der Pfade bei der Erzeugung	$1..s$	8 Bit
Laufende Paketnummer eingehend	Unique-ID	16 Bit
Laufende Paketnummer ausgehend	Unique-ID	16 Bit
Softstate zum Entfernen nichtbenötigter Verbindungen	Zähler	8 Bit

Tabelle 5.1: Benötigte Einträge im Schlüsselspeicher

5.3.4 Sicherheitsprotokolle (Security Protocols)

In den Sicherheitsprotokollen befinden sich sämtliche Algorithmen zum Erstellen neuer Verbindungen, zum Austausch von internen Informationen wie Gruppenschlüsseln oder zum Entfernen von Geräten aus einem Netzwerk. Details dieser Schicht werden in Kapitel 6 genau betrachtet.

Diese Schicht verhält sich rein passiv, reagiert also nur auf Stimuli bezüglich der unterliegenden Schicht (Ankunft von Paketen), der Kontrollschicht-Schicht (z.B. Timeouts oder Verbindungserzeugungen) oder der nach oben offenen Dienstschnittstellen (Senden von Paketen, Verbindungsaufbauten, ...)

5.3.5 Kontrollschicht (Controlling)

Als letztes und oberstes Element ist die Kontrollschicht vorhanden. In dieser werden aktiv Entscheidungen für das Schlüsselmanagement und die Sicherheitsprotokolle getroffen:

- Entfernen alter Schlüssel.
- Erzeugung von Timeouts bei den Sicherheitsprotokollen.

- Erzeugen zufälliger Verbindungen (siehe Kapitel 6.4.1).
- Senden von Discovery-Nachrichten.
- Überprüfen von bestehenden Verbindungen auf Korrektheit.

Ferner verfügt die Kontrollschicht über Möglichkeiten zur Administration der gesamten Sicherheitsschicht (z.B. der Reinitialisierung der Schlüsseldatenbank) wie in Kapitel 5.1 beschrieben .

5.4 Paketentwurf

Sämtliche Pakete, welche an die Kommunikationsschicht übergeben werden, haben den in Abbildung 5.2 dargestellten Aufbau:

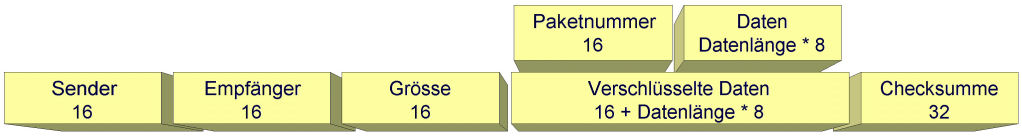

Abbildung 5.2: Aufbau eines Paketes

Die Felder für den Sender, den Empfänger und die Paketgrösse bilden hierbei einen Pseudoheader, welcher bei der Berechnung der Checksumme (einem Secure Hash, um eine Manipulation zu verhindern) verwendet wird, jedoch wenn möglich später als Parameter an die Kommunikationsschicht geht. Besteht nur die Möglichkeit eines Broadcasts (die Metadaten werden demnach nicht von einer untenliegenden Schicht eingebunden), so wird der komplette Header weitergereicht. Zur Berechnung der Checksumme wird das zugehörige Checksummenfeld vorher auf 0 gesetzt.

Einfache CRC-Summen innerhalb des verschlüsselten Blockes sind – wie in [22] beschrieben – zu vermeiden. Würde sich die der Authentifizierung dienende Checksumme innerhalb des verschlüsselten Bereiches befinden, so wäre dies nach [14] ebenfalls ein Sicherheitrisiko. Deshalb ist nachträglich das verschlüsselte Datenfeld mit einem Secure Hash, welcher zusätzlich das Shared Secret beinhaltet, authentifiziert.

Sender- und Empfängerfelder können (so es keine Anonymitätsanforderungen gibt) unverschlüsselt bleiben, ohne einen Angriff zu ermöglichen.

Die laufende Paketnummer dient der Verhinderung von Replayattacken und besteht aus einem Zähler, welcher für jedes übermittelte Paket um eins inkrementiert wird. Aufgrund von einer zeitlichen Änderung der Schlüssel (siehe Kapitel 6.5) ist hierbei ein 16 Bit Zähler ausreichend. Der Kommunikationspartner akzeptiert dabei nur Pakete, deren Paketnummer echt grösser ist als die zuletzt über diese Verbindung empfangene Paketnummer.

Das Typ-Feld beschreibt die Art des Paketes. Die Pakete werden entsprechend dieses Typ-Wertes innerhalb der "Security Protocols"-Schicht behandelt und die im Datenbereich liegenden Daten dementsprechend entpackt.

Die Verschlüsselung erfolgt entweder mit dem Gruppenschlüssel oder dem jeweiligen Verbindungsschlüssel bei Unicast-Kommunikation. Hierbei ist darauf zu achten, dass eine

Chaining-Methode wie z.B. CBC (Cipher Block Chaining) [27] verwendet wird, um ein Dechiffrieren von identischen Teilblöcken zu verhindern.

Kapitel 6

Protokollentwurf

In diesem Kapitel wird schrittweise der funktionale Aufbau der Sicherheitsschicht behandelt. Es werden zunächst die benötigten Komponenten hierfür betrachtet und diese daraufhin genauer analysiert.

6.1 Benötigte Protokolle

Um die benötigten Protokolle genauer untersuchen zu können, wird zunächst der Lebenszyklus eines Gerätes dargestellt:

1. Herstellung
2. Kauf eines Gerätes
3. Eingliederung in eigene Infrastruktur
4. Optionale Konfiguration mittels eines Masterdevices
5. Kommunikation mit anderen Geräten sowie Gruppenkommunikation
6. Wechseln von Schlüsseln über die Zeit
7. Entfernen aus der Infrastruktur
8. Verschrottung

Im Folgenden werden die Aufgaben innerhalb der einzelnen Phasen des Lifecycles genauer erläutert.

Herstellung Da Geräte nicht von Herstellerseite – in Bezug auf das Zielnetzwerk – vorkonfiguriert werden sollen, fallen weder bei der Herstellung noch beim Verkauf Aufgaben an den einzelnen Geräten an. Trotzdem ist beispielsweise die Vergabe einer eindeutigen ID möglich, wie sie beispielsweise – trotz einer Vielfalt von Herstellern – bei der Vergabe von MAC-Adressen bei Netzwerkkarten angewandt wird. Da jedoch immer nur Geräte einer einzigen Infrastruktur miteinander kommunizieren, ist es ebenso eine realistische Möglichkeit, die Geräte-ID mittels des, für die Kryptographie ohnehin benötigten, integrierten Zufallszahlengenerators festzulegen. Hierbei würde mit einer sehr geringen

Wahrscheinlichkeit einer Mehrfachverwendung von IDs der Aufwand für das individuelle Beschreiben der einzelnen Geräte entfallen. Wird eine solche Strategie gewählt, so gibt es zwei alternative Realisierungen. Entweder muss das mehrfache Vorhandensein einer einzigen ID dem Benutzer angezeigt werden. Dadürch kann dieser das entsprechende Gerät von Hand neu initialisieren ("Reset"). Die zweite Alternative ist das Sicherstellen einer dem Netz unbekannten ID bei Einbringung in die Infrastruktur (z.B. durch zusätzliche Protokolle).

Kauf eines Gerätes Bezüglich des Faktors s oder anderer algorithmenspezifischer Parameter kann eine Voreinstellung nötig sein. Da diese von Netzwerk zu Netzwerk unterschiedlich sein können, bestehen folgende Möglichkeiten:

1. Entscheidung für ein Verfahren mit festen Parametern sowie entsprechender Standardisierung.
2. Entscheidung für ein Verfahren, jedoch mit variablen Parametern, welche beispielsweise mittels DIP-Schaltern eingestellt werden können.
3. Entscheidung für frei konfigurierbare Geräte, welche mittels spezieller Hardware zu Beginn (beispielsweise beim Kauf des Gerätes von dem Verkäufer) konfiguriert werden.

Hierbei ist zu erwähnen, dass nur das allererste Gerät auf eine solche Weise konfiguriert werden muss. Weitere Geräte können bei Einbringung in das Netzwerk die Netzparameter erfragen und sich dementsprechend konfigurieren.

Um eine möglichst hohe Vertrauensbasis bei Käufern zu erreichen, ist eine Variante, in der das Gerät *nicht* von fremden Personen – in diesem Fall dem Verkäufer – konfiguriert wird, anzuraten. Um dadurch nicht die Notwendigkeit spezieller Konfigurationshardware zu erhalten, ist eine Verwendung von bspw. DIP-Schaltern empfehlenswert.

Eingliederung in eigene Infrastruktur Geräte werden – wie in Kapitel 4.1 beschrieben – mittels Imprinting an eine bestehende Infrastruktur gebunden.

Hierfür ist speziell ein Protokoll für das Übertragen von Einstellungen und Schlüsseln notwendig, welches in Kapitel 6.2 genauer analysiert wird.

Optionale Konfiguration mittels eines Masterdevices Ein Masterdevice ist ein Gerät mit dem absoluten Vertrauen aller anderen Geräte.

Für manche Einsatzzwecke kann das Vorhandensein eines einzelnen Gerätes, welches über höhere Rechte verfügt, nötig sein. Beispiele hierfür sind eine Rechteverwaltung oder ein Gerät zum Administrieren des Netzes. Dieses könnte unter anderem zum Reparieren des Netzes bei Partitionierungen oder für das Entfernen von kompromitierten bzw. sonst fehlerhaften Geräten zuständig sein. Ein weiteres Einsatzgebiet ist das Verbinden mehrerer Netze und somit eine Internetzwerkkommunikation durch solche Masterdevices.

Hierbei sei nochmals angemerkt, dass die Verwendung eines Masterdevices rein optional ist, also sämtliche Funktionalitäten der Sicherheitsschicht ohne einen solchen funktionieren. Vielmehr dient er zur Vereinfachung mancher Aufgaben und als Angebot an obere Schichten.

Kommunikation mit anderen Geräten sowie Gruppenkommunikation Nach dem Einbringen in ein Netzwerk ist eine Kommunikation möglich. Da jedoch anfangs nur eine geringe Anzahl von sicheren Verbindungen existiert, müssen – wie in Kapitel 4.3 beschrieben – neue Verbindungen erstellt werden. Verschiedene Möglichkeiten, wie dieses Ziel praktisch realisiert werden kann, werden in Kapitel 6.4 beschrieben.

Neben der direkten Kommunikation kann die Anforderung bestehen, eine Nachricht – ebenso wie in den Pebblenets [2] – gruppenauthentifiziert innerhalb der Gruppe zu broadcasten. Auf Grund von Speicherrestriktionen kann ein Gerät jedoch nicht gleichzeitig Verbindungen zu allen anderen Geräten halten. Deshalb wird in Kapitel 6.8 ein mögliches Verfahren vorgeschlagen, eine sichere Gruppenkommunikation mit einer einzigen versendeten Nachricht pro gewünschten Gruppennachricht zu ermöglichen.

Wechseln von Schlüsseln über die Zeit Um eine Kryptoanalyse zu verhindern, wird in Kapitel 6.5 ein Verfahren zur zeitlichen Erneuerung der gemeinsamen Geheimnisse vorgestellt.

Entfernen aus der Infrastruktur Für die Entfernung eines Gerätes kann es unter anderem folgende Gründe geben:

1. Planmässige Abschaltung
2. Zerstörung bzw. Defekte
3. Diebstahl
4. Kompromitierung

Während sich im ersten Fall das Netz darauf einstellen könnte, dass ein Gerät zukünftig nicht mehr vorhanden sein wird und Vorkehrungen treffen kann, wird bei allen anderen Fällen dieses zu spät festgestellt werden. In den Fällen 3 und 4 ist zusätzlich ein sofortiger Ausschluss aus dem Netz nötig. Methoden für die Revocation von Geräten werden im Kapitel 6.9 beschrieben. Ferner werden Methoden, um die Integrität des Netzes auch bei einzelnen Ausfällen zu gewährleisten, in Kapitel 6.6 behandelt.

Sollte das Netzwerk trotzdem bei Verwendung eines Masterdevices geteilt werden oder sollen Geräte mittels des Masterdevices auf eine einfachere Weise ausgeschlossen werden, so werden in Kapitel 6.7 Möglichkeiten aufgezeigt.

Verschrottung Da nach einem ordnungsgemässen Ausschluss eines Gerätes keinerlei Verbindungen mehr existieren, enthält dieses Gerät keine sensiblen Informationen mehr. Bei einem einfachen Entfernen ohne eine Revocation müssen jedoch sämtliche Schlüssel auf dem Gerät entfernt werden, welches durch ein Löschen des Speichers (Reset) geschieht.

6.2 Imprinting

In diesem Kapitel wird ein einfaches Protokoll für das Imprinten neuer Geräte beschrieben. Dabei wird das Verfahren, wie es in Kapitel 4.2 beschrieben wurde, angewandt.

Im einfachsten Fall – also ohne das Einbeziehen von Aspekten wie Revocation oder Masterdevices – hat das Imprinting die Aufgabe, auf einem sicheren Kanal zwischen zwei Geräten die Namen der beiden Geräte sowie ein Geheimnis auszutauschen.

Als sicherer Kanal bietet sich – da ja eine physikalische Verbindung verlangt wird [24] – zum Beispiel die serielle Schnittstelle an. Zwar ist ebenso ein proprietäres Format oder Technologien wie USB, Firewire usw möglich, jedoch besitzen die meisten Microcontroller bereits eine integrierte serielle Schnittstelle.

Eine Möglichkeit des Imprintens ist folgendes Vorgehen:

1. Der Käufer schliesst das neue Gerät – mittels des speziellen Kabels – an ein bereits vorhandenes Gerät an.

2. Ist die physische Verbindung hergestellt, drückt er bei dem neuen Gerät auf einen Taster. Dadurch schickt dieses eine Nachricht mit der eigenen ID, einem soeben erzeugten Geheimnis und der Bitte um das Imprinten.

3. Das Zielgerät empfängt diese Nachricht; überprüft, ob es zum Imprinten berechtigt ist und antwortet im positiven Fall mit der eigenen ID.

4. Beide Geräte speichern diese Informationen in ihrer Schlüsseldatenbank.

5. Dieser Vorgang wird insgesamt s mal wiederholt.

Da Geräte nicht nur unter Aufsicht, sondern auch unbeaufsichtigt (zum Beispiel im Garten oder Treppenhaus) installiert werden können, ist zu verhindern, dass ein Angreifer ohne Manipulation der Geräte ein weiteres in das Netzwerk einbringen kann. Hierfür muss eine Möglichkeit vorgesehen werden, dass ein Gerät sämtliche Kooperation beim Imprinten neuer Geräte verweigert. Ein praktikables Beispiel wäre ein Taster an dem Gerät, bei dessen Druck dieses Gerät einen Eintrag in seinem Speicher vorsieht, keinerlei weitere Geräte zu imprinten. Ein Angreifer müsste demnach – hat er keinen Zugriff auf zum Imprinten authorisierte Geräte – den Speicher eines Gerätes ändern, was einer Kompromittierung in Aufwand und Ergebnis gleichkommt. Ein solches Flag würde erst durch einen Reset gelöscht, bei dem aber auch alle gespeicherten Schlüssel gelöscht werden und einem Angreifer somit keinerlei Vorteil verschafft wird.

Das Kommunikationsprotokoll ist hierbei – in der Spezifikationssprache SDL – in Abbildung 6.1 beschrieben.

Das neue Gerät erhält – durch Knopfdruck – den Befehl zum Imprinten. Daraufhin generiert es einen zufälligen Schlüssel und sendet diesen, zusammen mit der eigenen ID, über den speziellen sicheren Kanal. Danach befindet es sich in einem Zustand, welches entweder – aufgrund eines "Not Acknowledge" oder eines Timeouts bei Kommunikationsproblemen – ohne einen Erfolg zum Startzustand führt oder es erhält ein "Acknowledge" und beendet das Imprinten erfolgreich mit einem Update der Schlüsseldatenbank.

Für die Gegenseite existiert das in Abbildung 6.2 beschriebene Kommunikationsprotokoll:

Da sich der Empfänger des Antrages bereits innerhalb des Netzwerkes befindet und ggf. zeitkritische Nachrichten senden und empfangen muss, ändert sich der Systemzustand hier nicht und das Gerät reagiert nur auf Anfragen. Empfängt es einen Antrag, so überprüft es seine Berechtigung zum Imprinten und antwortet im negativen Fall mit

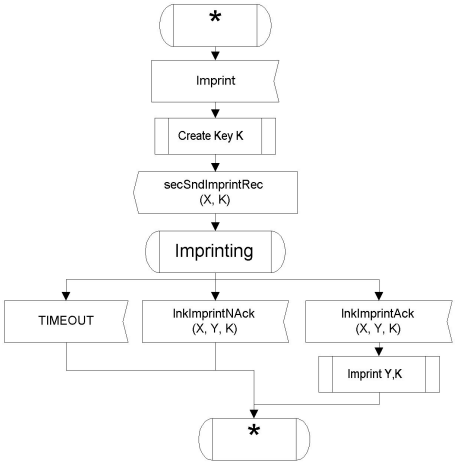

Abbildung 6.1: Imprint-Protokoll: Sender

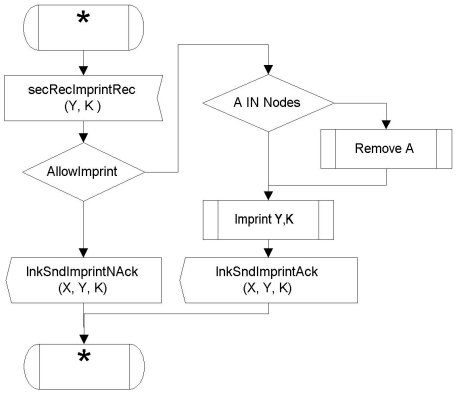

Abbildung 6.2: Imprint-Protokoll: Empfänger

einem "Not Acknowledge". Hat es die Berechtigung, so erstellt es einen neuen Eintrag in dessen Schlüsseldatenbank (und löscht wenn nötig einen bereits vorher existierenden) und antwortet mit einem "Acknowledge".

Erweiterungen hierzu sind:

Vorhandensein eines Masterdevices Eine Protokollerweiterung zum netzwerkweiten Konsens über ein Masterdevice wird in Kapitel 6.3.1 beschrieben.

Mögliche spätere Revocation durch die imprintenden Geräte Um den Geräten, welche am Einbringen eines (beispielsweise getamperten) Gerätes beteiligt waren, einen späteren Ausschluss dessen zu ermöglichen, wird in Kapitel 6.9.2 eine entsprechende Erweiterung aufgezeigt.

Verhinderung doppelter IDs Existieren mehrere Geräte mit der gleichen ID in einem Netzwerk, so ergeben sich daraus eine Vielzahl von Problemen: Die Addressierung ist nicht mehr eindeutig, etwaige Pfade stimmen nicht, die Netzintegrität

wird falsch bewertet und dadurch ggf. fehlerhafte Entscheidungen getroffen, welche unter Umständen zu einer Partitionierung des Netzes führen können. Sollte eine zufällige Vergabe der ID erfolgt sein, so muss sich das Gerät, welches sich bereits im Netz befindet, vor Annahme der Imprint-Aufforderung überzeugen, dass sich zu diesem Zeitpunt im Netz kein weiteres Gerät mit der gleichen ID befindet. Desweiteren ist – um einer nebenläufigen Einbringung zweier Geräte entgegenzuwirken – das Vorhandensein von fremden Paketen mit der eigenen ID anzuzeigen.

6.3 Verwendung von Masterdevices

Wie bereits in Kapitel 6.1 erwähnt, gibt es mehrere Vorteile durch das Einbringen eines optionalen Masterdevices:

Rechteverwaltung bzw. Geräte-Administration In einem Netzwerk, in dem zu Beginn alle Geräte über die gleichen Rechte verfügen, ist eine Administration schwer zu verwirklichen. Demnach kann bereits bei Aufbau des Netzes ein Gerät mit – für obere Schichten sichtbar – hervorgehobenem Vertrauen sinnvoll sein, um von diesem Masterdevice aus andere Geräte zu steuern.

Sicherheitsschicht-Administration Das Entfernen und Umkonfigurieren von einzelnen Geräten ist bei der Verwendung eines Masterdevices mit dem absoluten Vertrauen aller anderen simpel: Das Masterdevice schickt an die einzelnen Geräte authentifizierte Befehle und diese können – ohne weiterem Kommunikationsaufwand – entsprechend reagieren. Einige Beispiele hierfür sind:

Entfernen von Geräten Geräte, welche entfernt werden sollen, zerstört oder kompromitiert wurden, lassen sich ohne Spuren aus dem Netzwerk entfernen. Ein mögliches Protokoll zur Realisierung dieser Aufgabe zeigt Kapitel 6.9.1.

Netzreparatur Im Falle einer Netzpartitionierung kann das Mastedevice – unabhängig von dem im Kapitel 4.3 beschriebenen Verfahren zur Linkerzeugung – alleine neue Kanten zwischen Geräten erstellen (also neue Schlüssel erzeugen und an 2 Knoten verteilen).

Einfacheres Imprinten Gerade bei sehr hohen Werten für s ist das Imprinten neuer Geräte umständlich. Durch Zuhilfenahme des Mastedevices kann dieser Aufwand auf das einmalige Einstecken des neuen Gerätes verringert werden.

Realisierung eines IDS Befindet sich ein aktives Masterdevice in einem Netz, so kann dieses die Kommunikation überwachen und bei anormalem Verhalten einzelner Geräte dieses sofort aus dem Netz aussperren.

Internetzwerkkommunikation Existieren mehrere Netzwerke, von denen *manche* Geräte unterschiedlicher Netzwerke miteinander kommunizieren möchten, jedoch trotzdem die Netzwerke nicht vereinigt werden sollen, so kann dies mittels der Masterdevices erreicht werden. Hierfür bauen diese einen Tunnel für solche Nachrichten zwischen den Netzen auf.

Neben den Vorteilen eines solchen Masterdevices besteht auf der anderen Seite jedoch immer das Risiko einer kompletten Netzzerstörung bzw. Übernahme bei Kompromitierung des Masterdevices.

Im Folgenden wird zuerst eine Möglichkeit zum Einbringen eines Masterdevices in ein Netzwerk vorgestellt. Danach wird exemplarisch für die daraus resultierenden Möglichkeiten das einfache Imprinten vorgestellt.

6.3.1 Einbringen eines Masterdevices

In diesem Kapitel wird ein Verfahren vorgestellt, mittels dem beim Netzaufbau das allererste Gerät zu einem Masterdevice bestimmt werden kann. Dies geschieht folgendermassen:

Jedes Gerät enthält einen speziellen Speicherplatz, in dem optional eine ID für das Masterdevice stehen kann. Wird das erste Gerät als Masterdevice gewählt (z.B. durch Drücken eines speziellen Tasters), so speichert dieses Gerät seine eigene ID auf diesem Speicherplatz.

Das Imprintingprotokoll wird folgendermassen abgewandelt: Mit einem "Acknowledge", welches das im Netz befindliche Gerät an das Neue sendet, wird zusätzlich die Masterdevice-ID mitgeschickt. Bei Empfang eines solchen Paketes reagiert das neue Gerät folgendermassen:

- Ist es das allererste "Acknowledge", welches das Gerät empfängt, so speichert es selber die Masterdevice-ID und das Imprinten wird weitergeführt.
- Ist die empfangene Master-ID gleich der bereits gespeicherten, so wird das Imprinten weitergeführt.
- Ist die empfangene Master-ID anders als die gespeicherte (oder ist in einem der beiden kein Master angegeben), so wird visuell ein Fehler angezeigt und das Imprinten abgebrochen. Dieser Fall resultiert, wenn mindestens ein Gerät im Netzwerk über eine andere Masterdevice Information verfügt – ein Fall, der bei einem korrekt funktionierenden Netzwerk (ohne Angreifer) nicht vorkommen kann

Auf diese Weise kann sichergestellt werden, dass jedes Gerät, welches sich im Netzwerk befindet, am Ende des Imprinting-Prozesses den gleichen Master-ID-Eintrag hat, denn würde ein kompromittiertes Gerät einen unterschiedlichen Eintrag (z.B. sich selbst) liefern, so würde das neue Gerät zwei unterschiedliche IDs empfangen. Dementsprechend müssten mindestens s Geräte (bzw. alle Geräte, sollten sich noch nicht s Geräte im Netz befinden) von einem Angreifer übernommen sein, um einem neuen Gerät ein anderes Masterdevice vorzugeben – eine Anforderung, die der Übernahme des gesamten Netzes gleichkommt. Gleiches gilt, will ein Angreifer durch Vortäuschen der Masterdevice-ID als eigene ID ein anderes Gerät befehligen. Mindestens s Geräte müssten für einen Linkaufau diese ID in Verbindung mit einem beim Imprinten abgelegten Schlüssel bezeugen.

Dieses Verfahren setzt voraus, dass das Masterdevice vor dem Netzaufbau festgelegt ist. Dies bedeutet, dass ein nachträgliches Austauschen des Masterdevices auf Grund von Defekten oder aus anderen Gründen nicht ohne Weiteres möglich ist. Mögliche Lösungen sind das Sichern der Schlüssel das Masterdevices, eine komplette Emulation der Sicherheitsschicht auf Softwareebene oder ein Protokoll ähnlich dem Linkaufbau über s Pfade, auf dem die neue Master-ID bekannt gegeben wird.

6.3.2 Imprinting mittels Masterdevice

Ein Masterdevice kann – auf Grund des absoluten Vertrauens aller anderen Geräte – an Stelle der physikalischen Verbindung als Mittelsmann fungieren. Das Imprinten kann demnach so abgekürzt werden, dass das neue Gerät nur an das Masterdevice verbunden wird und dieses danach mit $s-1$ beliebigen anderen Geräten einen Schlüsselaustausch organisiert.

Dies kann folgendermassen geschehen:

Das neue Gerät imprintet sich an das Masterdevice. Danach wird es nicht weiter an andere Geräte angeschlossen. Da die Kompromitierung des Masterdevices ausgeschlossen wird, reicht hier das einmalige Einstecken. Daraufhin wählt das Masterdevice eine Teilmenge von $s-1$ *aktiven* anderen Geräten (z.B. durch Anfragen auf Verfügbarkeit). Ist das Masterdevice im normalen Betrieb nicht eingeschaltet, so werden s Geräte ausgewählt.

Danach geht das Masterdevice nacheinander für jedes der gewählten Geräte folgendermassen vor: Es generiert einen geheimen Schlüssel und schickt diesen als 3-Tupel (neues Gerät, bestehendes Gerät, Schlüssel) jeweils verschlüsselt und authentifiziert an die beiden Geräte. Diese nehmen jeweils den Schlüssel und somit die neue Verbindung in ihren Schlüsselspeicher auf.

Ein solches Vorgehen hat zur Folge, dass das Masterdevice sämtliche Kommunikation zwischen dem neuen Gerät und allen anderen Geräten (auch denen, welche nachträglich über den normalen Linkaufbau nach 4.3 ihre Verbindung erzeugt haben) lesen und fälschen kann. Auf der einen Seite untermauert dies den "Single Point Of Trust"-Aspekt des Masterdevices, auf der anderen Seite kann dadurch auch das Masterdevice im Rahmen seiner Aufgabe als IDS den Verkehr überwachen.

6.4 Linkerweiterung

Nach dem Einbringen der Geräte existiert ein s-fach zusammenhängender Graph. Direkt nach dem Imprinten sind die Geräte jedoch einerseits bestrebt, die Anzahl der sicheren Verbindungen – also der eigenen Schlüssel – von s auf z zu erhöhen (um eine Ausfallsicherheit zu erlangen, siehe Kapitel 6.6.1), des Weiteren können diese Geräte einen Kommunikationsbedarf mit Geräten besitzen, mit denen diese noch kein Geheimnis ausgetauscht haben.

Um beliebigen Geräten miteinander eine Kommunikation zu erlauben, müssen Algorithen zum Einsatz kommen, welche

1. s knotendisjunkte Pfade zwischen den beiden Kommunikationspartnern finden.
2. ein Geheimnis mittels dieser Pfade bei den Partnern etablieren.

Während es – wie in Kapitel 4.3 angegeben – auf Grund des Imprintingverfahrens immer diese s Pfade gibt, ist festzustellen, dass sich die maximale Pfadlänge hierbei nur auf $n-s+1$ einschränken lässt, sie also linear abhängig von der Gesamtgrösse des Netzwerkes ist.

Schon alleine bei dieser Beschränkung ist festzustellen, dass eine allgemeine Pfadsuche nicht grundsätzlich möglich sein kann. Der Speicher, der zum Abspeichern der Pfade verwendet wird, ist durch die Wahl der Mikrocontroller begrenzt.

Even und Tarjan [11] stellten fest, dass sich bei einem fest vorgegebenem s für eine Pfadsuche der Aufwand sowie die benötigte Zeit in $O(n + m_{all})$ befinden, also $O(n \cdot m_{node})$. Daraus resultiert die Notwendigkeit der Verwendung probabilistischer Methoden bei Geräten mit zu geringem Speicher, um sämtliche Pfade zu finden. Battista et al. [3] zeigten eine Verbesserung für $s = 2$ auf $O(n)$, dies führt jedoch – da der verfügbare Speicherplatz m_{node} herstellungsbedingt konstant ist – zu keinen weiteren Vorteilen.

Im Folgenden werden verschiedene Methoden, mittels derer die Pfadsuche erfolgen kann, vorgestellt. Dabei werden zuerst 2 unabhängige Verfahren entwickelt, sowie danach mögliche Erweiterungen untersucht.

6.4.1 Zwei-Hop Protokolle

Zwei-Hop-Protokolle bilden die einfachste Art der Pfadfindung: Es werden nur Pfade mit einer Länge von 3 betrachtet. Einfach ausgedrückt bedeutet dies: Zwei Geräte können genau dann eine sichere Verbindung zwischen sich aufbauen, wenn es mindestens s Geräte gibt, welche bereits über eine sichere Verbindung zu diesen *beiden* Geräten verfügen. Abbildung 6.3 illustriert dies.

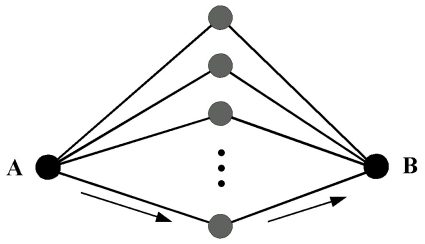

Abbildung 6.3: Verbindungsaufbau in Zwei-Hop-Protokollen

Es ist klar, dass es hierbei nicht immer zu einem Erfolg kommen kann. Kapitel 8.1 befasst sich mit der theoretischen Erfolgsquote und zeigt dabei auf, dass sich bei geeigneter Parameterwahl sehr gute Erfolgswahrscheinlichkeiten erreichen lassen.

Unabhängig von jeglicher Parameterwahl gibt es jedoch immer *mindestens* eine Verbindung in einem nicht vollvernetzten Netzwerk, welche mittels dieses Verfahrens erstellbar ist:

Gegeben sei ein Netzwerk, welches nach den in Kapitel 4.1 vorgegebenen Regeln zum Imprinten erstellt wurde. Hierbei seien die einzelnen Geräte gemäss ihree Einbringungszeitpunktes durchnumeriert. Sei nun das erste Gerät, welches nicht mit allen vorherigen Geräten verbunden ist K_i. Da alle Vorherigen mit allen anderen noch weiter vorher eingebrachten Geräten verbunden sind, ist dieses Teilnetz vollvernetzt. Der Knoten K_i hat – um ordnungsgemäss imprintet zu sein – mindestens s Kanten in diese vollvernetzte Menge und es existiert auf Grund der Auswahl von K_i mindestens ein Knoten K_j mit $j < i$, zu dem K_i über keine Verbindung verfügt. Da K_j vor K_i eingebracht wurde, ist dieser demnach ebenso Teil der vollvernetzten Menge. Folglich existieren für jede Kante von K_i in diese vollvernetze Menge genau ein Pfad der Länge 3 zu dem Zielknoten K_j.

Mit dieser bewiesenen Eigenschaft lässt sich die Eigenschaft der *Lifeness*, der Lebendigkeit des Netzwerkes einführen. Bildet das Netzwerk eigenständig zufällige neue Verbin-

dungen aus (und verwirft auch zufällig wieder alte, um die Gesamtmenge aller Schlüssel konstant zu halten), so verändert sich die Netzwerkstruktur über die Zeit. Dies führt dazu, dass die Wahrscheinlichkeit, dass sich eine bestimmte Verbindung *irgendwann* erzeugen lässt, gegen eins geht.

Hierbei sei angemerkt, dass bei dem zufälligen Löschen nur die durch das Linkaufbau-Protokoll entstandene und keine imprinteten Verbindungen gelöscht werden dürfen, um die durch den Aufbau erhaltenen Eigenschaften des Netzwerkes beizubehalten.

Ein einfaches und unoptimiertes Protokoll, welches den Linkaufbau, mit $O(1)$ Speicherplatz und $O(m_{node})$ Zeitaufwand im initiierenden Knoten und dem Zielknoten, erreicht sowie keinerlei Zustandsinformationen bei anderen Knoten benötigt, wird im Folgenden beschrieben.

Discovery Um eine Lifeness zu ermöglichen, muss ein Knoten seine Nachbarn kennen. Da er jedoch auf Grund der Ressourcenbeschränkungen keine Liste aller Knoten speichern kann, wird hier das periodische Aussenden von "Hello"-Nachrichten vorgeschlagen. Dieses geschieht per unverschlüsseltem Broadcast (in diesem Fall für die Gruppe verschlüsselt, siehe Kapitel 6.8). Abbildung 6.4 zeigt dieses in SDL. Durch einen Impuls (sei dieser durch einen Timer gesteuert oder auf Grund anderer Vorgaben) wird ein Paket mit der eigenen ID verschickt.

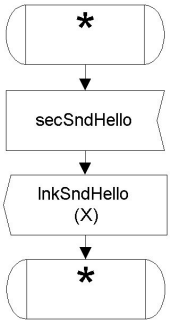

Abbildung 6.4: Pfadfindung-Protokoll: Discovery

Initialisierung Empfängt ein anderes Gerät das "Hello"-Paket eines Knotens (in einem Zustand, in dem es imprintet und bereit ist), so meldet dieses (in Abb. 6.5) zuerst das Vorhandensein des anderen Gerätes an obere Schichten, da zum Beispiel auch eine Rechteverwaltung das Vorhandensein anderer Geräte interessiert. Danach wird – sollte noch keine Verbindung zu dem Sender vorhanden sein – probabilistisch eine Verbindungserstellung eingeleitet oder an dieser Stelle abgebrochen. Im Linkaufbau werden die in Tabelle 6.1 aufgeführten Statusvariablen verwendet. Zu Beginn des Linkaufbaus werden die Schlüssel generiert und eine Anfrage an ein "bekanntes" Gerät (das an der Stelle *pos* im Schlüsselspeicher, hier mit N_{pos} gekennzeichnet) geschickt.

Weiterleitung Empfängt ein Gerät beliebigen Zustandes ein "Create"-Paket, so unterscheidet es – wie in Abbildung 6.6 dargestellt – folgende zwei Fälle:

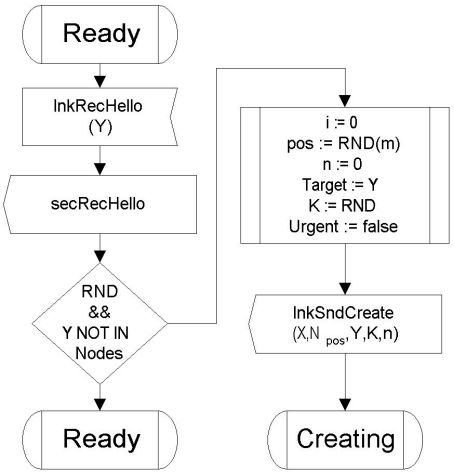

Abbildung 6.5: Pfadfindung-Protokoll: Initialisierung

i	Eine Variable, welche über den gesamten Schlüsselspeicher läuft
pos	Eine zu i zugehörige Schlüsselposition
n	Die Anzahl der gefundenen Pfade
$Target$	Die ID des Zielgerätes
$K_{1..s}$	Die Teilschlüssel
$Urgent$	Ein Flag welches den angibt, ob der Verbindungsaufbau auf Wunsch oberer Schichten abgebrochen werden darf

Tabelle 6.1: Statusvariablen bei der Verbindungserstellung

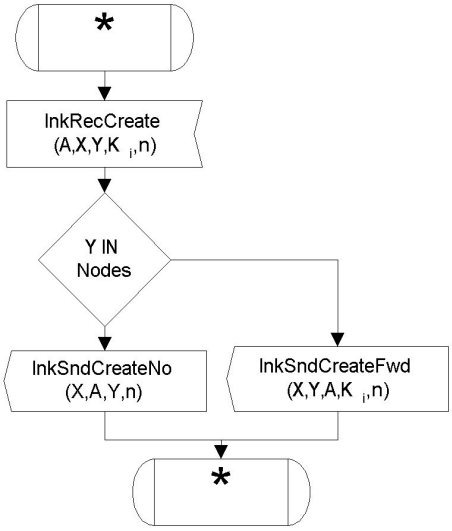

Abbildung 6.6: Pfadfindung-Protokoll: Weiterleitung

- Ist das Zielgerät bekannt, so wird die Anfrage an dieses weitergeschickt.
- Ist das Zielgerät nicht bekannt, so wird eine Negativmeldung zurückgeschickt.

Ankunft War das Zielgerät auf dem zwischenliegenden Gerät bekannt, so empfängt dieses danach die Weiterleitung und geht – wie in Abbildung 6.7 beschrieben – folgendermassen vor:

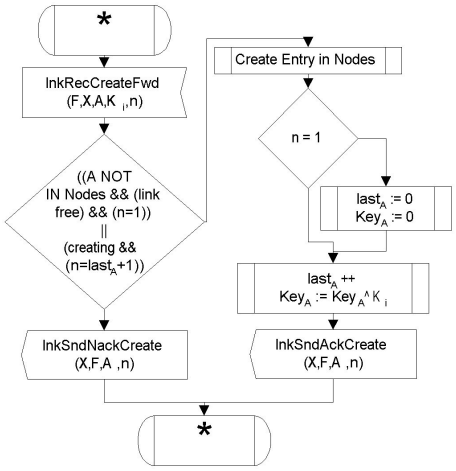

Abbildung 6.7: Pfadfindung-Protokoll: Ankunft

Ist eine der folgenden Bedingungen wahr, so wird dieses Paket nicht akzeptiert und mit einer "Not Acknowledge"-Nachricht geantwortet:

- Es ist bereits ein Schlüssel zu dem Startgerät vorhanden.
- Es ist kein Speicherplatz für einen weiteren Schlüssel vorhanden und dies der erste Pfad.
- Dies ist weder der erste aufgebaute Pfad noch (bei bereits vorhandenen Pfaden) der erwartete.

In allen anderen Fällen wird das Paket akzeptiert. Für den Schlüsselaufbau wird zuerst ein gegebenenfalls unvollständig vorhandener Schlüssel bei Eintreffen der ersten Pfadnachricht gelöscht ($Key_A = 0$) und dessen Zustand zurückgesetzt ($last_A = 0$). Sollte noch kein Eintrag vorhanden sein, wird hierbei ein Neuer erstellt. Danach wird der Eintrag in der Schlüsseldatenbank erneuert: Der bisher vorhandene Schlüssel (zu Beginn 0) wird mit dem neu erhaltenen Schlüssel per XOR-Operation verbunden und abgespeichert. Ferner wird die Pfadzahl, mittels derer dieser Schlüssel entstand, wird um eins erhöht. Hierbei ist anzumerken, dass ein Schlüssel erst dann verwendet werden darf, wenn dessen Pfadzahl s erreicht hat.

Danach wird ein "Acknowledge"-Paket an das zwischenliegende Gerät zurückgeschickt.

Rückleitung Nach dem Weiterleiten der Anfrage erhält ein zwischenliegendes Gerät – sollte es zu keinem Paketverlust gekommen sein – entweder ein "Acknowledge" oder ein "Not Acknowledge". Das Eintreffen eines solchen Paketes meldet es mittels einer positiven oder negativen Antwort an den ursprünglichen Anfragen-Sender gemäss Abbildung 6.8 weiter.

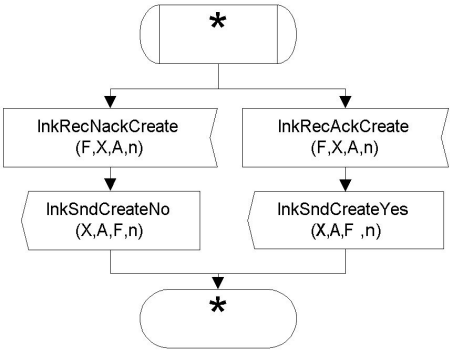

Abbildung 6.8: Pfadfindung-Protokoll: Rückleitung

Antwort Der letzte Teil des Linkaufbauprotokolls besteht in der Annahme von Antworten auf Initiatorseite. Da der Initiator das einzige Gerät ist, welches bei einem Linkaufbau aktiv seinen Zustand ändert, ist hier die Hauptkomplexität zu finden (siehe Abbildung 6.9).

Im Zustand der Linkerstellung gibt es vier mögliche Eingaben:

- Ein internes AbortCreate-Ereignis bricht den Linkaufbau sofortig ab. Eventuell zum Senden anstehende Nachrichten werden dabei verworfen.
- Die negative Antwort eines zwischenliegenden Knotens.
- Ein Timeout. Der zugehörige Timer wird mit jedem Aussenden einer "Create"-Nachricht gestartet. Da bei Eintreffen dieses Ereignisses keine Annahmen über positive Ergebnisse gemacht werden können, wird dies mit einer negativen Antwort gleichgesetzt.
- Die positive Antwort eines zwischenliegenden Knotens.

Wird eine negative Antwort (beziehungsweise ein Timeout) empfangen, so wird die Variable i inkrementiert und die nächste Geräte-ID im Schlüsselspeicher abgefragt, um an diesen die nächste Anfrage zu stellen. Wurden alle Geräte im Schlüsselspeicher befragt (ist i also grösser der Gesamtanzahl m), so kommt dies einem AbortCreate gleich und der Verbindungsaufbau schlägt fehl. Ansonsten wird das nächste "Create"-Paket gesendet und der Timeout neu gestartet.

Wird eine positive Antwort empfangen, so wird zuerst die Anzahl der gefundenen Pfade inkrementiert und der eigene Schlüssel – analog zum Empfänger – durch ein XOR modifiziert. Sind noch keine s Pfade gefunden, so wird danach wie bei dem Eintreffen einer negativen Antwort fortgefahren. Ansonsten gilt die Verbindung als aufgebaut und ein – gegebenenfalls wartendes Paket – kann gesendet werden.

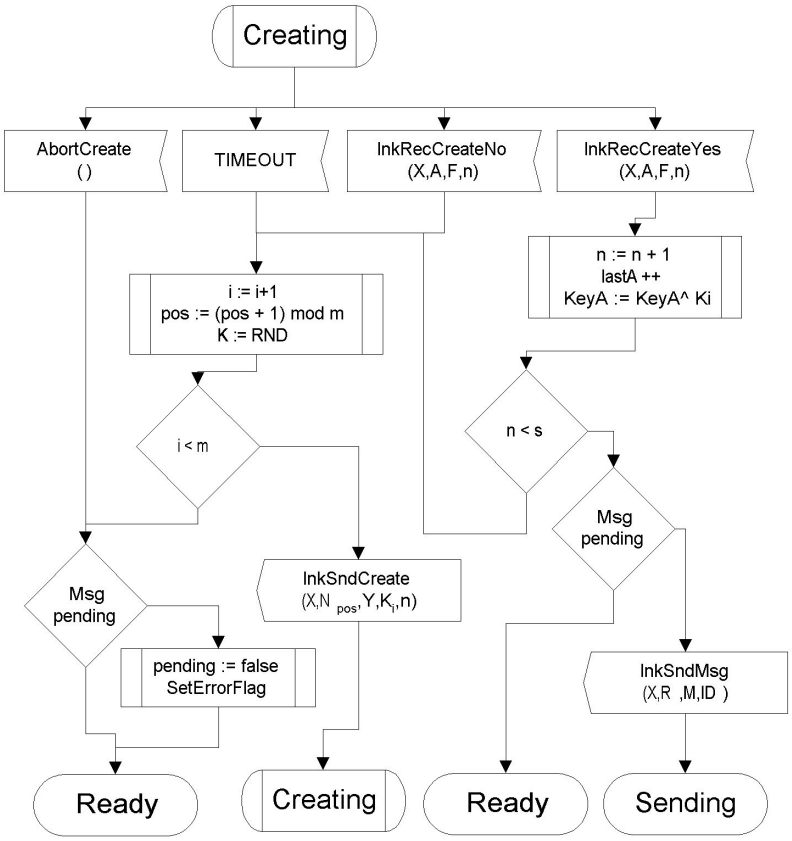

Abbildung 6.9: Pfadfindung-Protokoll: Antwort

6.4.2 Multi-hop Protokolle

Um höhere Wahrscheinlichkeiten zu erreichen, beschäftigen sich Multi-Hop-Protokolle mit der Suche von Pfaden mit Längen grösser drei. Da nicht nur ein Pfad, sondern s knotendisjunkte Pfade gefunden werden müssen, eignen sich effiziente Wellenalgorithmen weniger, denn ein Gerät kann – ohne globales Wissen – nicht einschätzen, welche Pfade wichtig sind, und welche nicht. Deshalb werden im Folgenden drei Ansätze vorgestellt:

- Pfadfindung mittels Tiefensuche, in der eine simple Tiefensuche fest vorgegebener Tiefe zum Einsatz kommt.
- Pfadfindung mittels Ringsuche, um die Anzahl der Nachrichten zu minimieren, wenn die Maximaltiefe zur Pfadfindung nicht notwendig ist.
- Pfadfindung durch lokal nachgebildeten Graphen, in denen – um die Nachrichtenzahl zu verteilen und zu minimieren – die Pfadsuche rein lokal geschieht.

Pfadfindung mittels Tiefensuche

Bei der traditionellen Tiefensuche schickt ein initiierendes Gerät eine Anfrage mit einer Resttiefe nacheinander an alle ihm bekannten Geräte. Diese Geräte decrementieren die

Resttiefe und schicken, wenn sie der Zielknoten sind, ihren Pfad zurück oder rufen ihrerseits nacheinander alle Nachbarn auf. Dies führt dazu, dass

- jeder beteiligte Knoten für jede Anfrage mindestens zwei Zustandsvariablen halten muss (nämlich Anfragequelle und aktuell befragter Knoten), sollten Paketverluste möglich sein.
- es maximal m_{max}^{depth} Nachrichten gibt, welche alle nacheinander ablaufen.

Vor allem der zweite Punkt verhindert diese Möglichkeit, da – selbst wenn man von einer Sendedauer von $10us$ ausgeht – eine Pfadsuche dadurch bei $m = 40$ und $depth = 5$ mit $t = 10us \cdot 40^5$ über 17 Minuten dauern könnte (im schlechtesten und unwahrscheinlichsten Fall). Des Weiteren würden – da theoretisch sämtliche Geräte gleichzeitig einen Linkaufbau initiieren könnten – in jedem Gerät $O(n)$ Speicher benötigt. Eine Eigenschaft ist per Definition nicht möglich. Da es – sollten alle Nachrichten gleichzeitig gesendet werden – zu einem "Broadcast Storm" kommen würde (im obrigen Beispiel ergäbe dies 2.5 Millionen "gleichzeitiger" Nachrichten in der fünften Zeiteinheit), ist dies ebenso nicht möglich.

Pfadfindung mittels Ringsuche

Eine Möglichkeit, die Nachrichtenmenge zu reduzieren, ist die Ringsuche (auch Depth first iterative deepening). Hierbei wird zuerst mit einer Tiefe von 1, danach mit 2 usw. bis zu *depth* gesucht; mit dem Vorteil, dass – sollte die Lösung vor der Maximaltiefe gefunden werden – jederzeit abgebrochen werden kann. Bei Verwendung dieses Verfahren entstehen im Worstcase doppelt so viele Nachrichten, aber auch nur die Einsparung eines Tiefenschrittes verringert die Nachrichtenzahl auf das $1/m$-fache. Auch ist bei geringen Tiefen das gleichzeitige Senden aller Nachrichten ohne das Aufkommen eines Broadcast Storms möglich, was die durchschnittliche Wartezeit bei einem Linkaufbau weiter minimiert.

Dennoch ist dieses Verfahren gerade für hohe Pfadtiefen nicht geeignet. Insbesondere bleiben die Speicheranforderungen auf allen Geräten erhalten.

Pfadfindung durch lokal nachgebildeten Graphen

Eine völlig unabhängige Methode der Pfadfindung ist das lokale Nachbilden des Graphen mittels der Discovery-Nachrichten anderer Geräte und das spätere Suchen in diesen Daten.

Für eine Beschreibung des Verfahrens wird zuerst von einem genügend grossen Speicherplatz ausgegangen, um den gesamten Graphen abzuspeichern ($n \cdot m_{node} = m_{all}$ Speicher pro Gerät). Diese Eigenschaft wird später gelockert.

Um den Graphen nachzubilden, senden die einzelnen Geräte in ihren "Hello"-Paketen neben ihrer eigenen ID noch die ihrer Nachbarn, zu denen sie eine feste Verbindung besitzen. Fest bedeutet hier, dass diese mit einer sehr hohen Wahrscheinlichkeit bis zu der nächsten Discovery Nachricht existent bleibt und folglich nicht auf einem lokalen Subgraph ein Pfad gefunden wird, welcher in der Realität bereits nicht mehr existiert. Die Auswahl dieser Kanten ist von den Verwerfungsalgorithmen für die Lifeness abhängig. Während imprintete Pfade grundsätzlich nie verworfen werden dürfen, können z.B. von oberen Schichten angeforderte oder häufig benutzte Verbindungen eine geringere Verwurfswahrscheinlichkeit besitzen und somit auch verschickt werden.

Wird ein solches Paket empfangen, so wird die lokale Nachbildung des Graphen gemäss den Angaben des Paketes aktualisiert.

Bei einer späteren Pfadsuche können dann – an Stelle von verteilten Suchalgorithmen – lokale Algorithen wie [11], [3] ohne eine Ressourcenbelastung anderer Knoten verwendet werden. Nach dem Finden der Pfade kann danach ein Geheimnis zerteilt und über die s Pfade verschickt werden.

Probleme dieses Verfahrens bleiben:

- Der Speicherbedarf beträgt O(n)
- Die Pfadlänge selber liegt in O(n); somit wachsen Pfadinformationen, welche zwingend in den Paketen zum Verschicken der Geheimnisse vorhanden sein müssen, linear mit der Netzwerkgrösse.

Eine Möglichkeit zum Überwinden dieser Probleme ist das Begrenzen der maximalen Pfadlänge und des lokalen Subgraphen auf eine (eventuell geringere) Teilmenge von Geräten.

Auswahlkriterien können hierbei sein:

- Grundsätzlich werden nur Pakete von Knoten angenommen, welche im derzeitigen lokalen Subgraphen mit einer maximalen Entfernung von d_max über *einen* Pfad erreichbar sind
- Zusätzlich werden – sollte trotzdem eine Maximalzahl von Knoten erreicht sein – zufällige Geräte aus dem Subgraphen entfernt. Hiermit wird die, zu Beginn geforderte, ausreichende Speichergrösse hinfällig.

Wird ein solcher Subgraph verwendet, so müssen in den Discoverynachrichten zwingend auch nicht imprintete, zufällige Kanten (welche dann z.B. bis zu der nächsten eigenen Discovery Nachricht nicht gelöscht werden) mitgesendet werden, um die, wie in Kapitel 6.4.1 beschriebene, Lifeness bei diesem probabilistischen Verfahren zu gewährleisten.

6.4.3 Pfadfindung mittels Delegation

Unabhängig von der verwendeten Methode zur Pfadfindung können am Ende einer Suche nicht genügend (oder keine) Pfade gefunden worden sein. Werden keine zusätzlichen Vorkehrungen getroffen, so bedeutet dies, dass eine Meldung über das Nichtgelingen des Linkaufbaus gemeldet wird und es "später" erneut probiert werden muss.

Eine Möglichkeit, dieses Warten zu minimieren, ist das Delegieren des Verbindungswunsches an andere Geräte. Bei einer Suche mit einer Tiefe von vier ist es so beispielsweise möglich, bei einem nicht geglücktem Linkaufbau, in dem zwei Pfade zu wenig gefunden wurden, folgendermassen vorzugehen:

Einer gewissen Menge von Knoten, welche drei Hops entfernt sind und zu denen ein Pfad existiert, wird eine Aufforderung geschickt, selber einen Linkaufbau zu versuchen. Glückt dieses bei zweien der aufgeforderten Geräte, so kann danach die ursprüngliche Verbinung etabliert werden. Analog funktioniert dieses Verfahren bei einem Zwei-Hop-Protokoll, indem Nachbarn, die den Zielknoten nicht kennen, zu einem Aufbau aufgefordert werden.

Für eine ordnungsgemässe Funktion ist zu beachten, dass

- die Delegation nicht Rekursiv angewandt wird, zum Beispiel nur bei von oberen Schichten initiierten Verbindungen.
- an nicht zu viele andere Knoten delegiert wird, um einerseits den verursachten Traffic zu minimieren, andererseits den Schlüsselspeicher des Zielgerätes nicht durch zu viele delegierte Kanten so zu füllen, dass eine weitere Verbindung nicht mehr möglich wird.

6.4.4 Auswahlkriterien für die Pfadwahl

Während im vorherigen Kapitel das Verhalten bei zu wenig Pfaden betrachtet wurde, kommt als zweites der Aspekt der Pfadwahl bei mehreren Alternativen. Dabei stellen sich folgende Fragen:

- Wie sind längere Pfade zu beurteilen?
- In wie weit sind imprintete Verbindungen anderen vorzuziehen?

Letztendlich sind für die Analyse die drei in Kapitel 8.5.1 angeführten Arten eines Angriffes zu verwenden.

Kein Gerät ist getampered Hierbei verläuft jegliche Kommunikation verschlüsselt und Pakettyp und Pfadlänge sind für einen Angreiffer dabei nicht erkennbar. Ebenso verhalten sich imprintete Verbindungen gleich wie nachträglich erstellte Verbindungen.

1 bis $s-1$ Geräte sind getampered Hier steigt die Wahrscheinlichkeit für einen Angreiffer, dass einer oder mehrere (aber niemals alle) Pfade über getamperte Geräte laufen, mit der Pfadlänge. Der Angreiffer hat hiebei folgende Möglichkeiten:

- Er verhält sich bei der Weiterleitung ordnungsgemäss.
- Er verfälscht die Informationen (z.B. den Schlüssel).

Während der Angreiffer im ersten Fall keinerlei Informationen, die ihm bei einer Kompromitierung der Verbindung helfen, erhält, verhindert er im zweiten Fall höchstens die Verbindung, da kein gemeinsamer Schlüssel zustande kommt.

Unabhängig von seinem Verhalten erhält er jedoch keinen eigenen Vorteil ausser dem Stören des Netzes und somit dem eigenen Kenntlichmachen bei Intrusion Detection Systemen oder dem Benutzer.

s oder mehr Geräte sind getampered In diesem Fall steigt zwar die Wahrscheinlichkeit, dass bei einem längeren Pfad dieser kompromitiert ist und letztendlich, dass alle Pfade kompromitiert sind, jedoch gilt in diesem Fall das Netz an sich für kompromitiert und es werden keinerlei Zusagen an jegliche Sicherheit gemacht. Der Angreiffer könnte beispielsweise unter anderem einen Knoten mit der gleichen ID wie dem Zielknoten erzeugen und diesen als "Man in the middle" agieren lassen.

Somit ist festzustellen, dass weder Pfadlänge, noch Verbindungstyp die Sicherheit einer Verbindung erhöhen. Einzig die Wahrscheinlichkeit eines ungestörten Verbindungsaufbaus steigt mit kleineren Pfadlängen.

6.5 Schlüsselerneuerung

Um einem aussenstehenden Angreifer keine grosse Datenmenge, welche mittels eines einzigen Schlüssels verschlüsselt wurde, zu liefern – und somit Grundlage wie Zeit für eine Kryptoanalyse zu schaffen – ist eine Änderung des Schlüssels von Zeit zu Zeit sinnvoll.

Hierbei existieren drei Möglichkeiten für eine Schlüsselerneuerung:

- Verschicken des neuen Schlüssels über die bisherige Verbindung.
- Verwenden des normalen Protokolls zur Linkerstellung (siehe Kapitel 4.3)
- Verschicken des Schlüssels über die bestehende Verbindung und – wen möglich – zusätzliche Stärkung des Schlüssels über weitere Pfade.

Während die erste Methode in ihrer Realisierung simpel ist (nach einer gewissen Anzahl von Nachrichten oder einer vorgegebenen Maximalzeit wird ein neuer Schlüssel generiert und dieser bestätigt an den Kommunikationspartner geschickt), bedeutet die Analyse eines einzigen Schlüssels die Möglichkeit zum Erhalt aller folgenden Schlüssel.

Die zweite Möglichkeit, den nachträglichen Schlüsselaustausch zu realisieren, ist – wie bei der Erstellung neuer Verbindungen – das Finden von s Pfaden und ein Austausch über diese. Während ein solches Verfahren die geringsten Möglichkeiten für passive Angriffe bietet, verlangt dieses jedoch das Vorhandensein von s nutzbaren Pfaden, welche (wie im letzten Kapitel beschrieben) zwar existieren, jedoch nicht immer gefunden werden können. Dies führt dazu, dass zwei Kommunikationspartner so lange ihr Schlüsselpaar behalten, wie ein Finden von s Pfaden nicht glückt oder verhindert wird.

Die dritte Möglichkeit verwendet die Eigenschaft, dass für eine Kommunikation zwischen den Knoten A und B dem Pfad AB vollständig vertraut wird und somit nicht die Notwendigkeit von s knotendisjunkter Pfade gegeben ist. In diesem Fall wird vor dem Schlüsselaustausch eine herkömmliche Pfadsuche initiiert und die gefundenen Pfade (unabhägig von einer Mindestzahl) zusätzlich zu dem bereits existierenden Pfad AB hinzugenommen. Dies bedeutet, dass ein Angreifer, welcher einen Schlüssel zu einem bestimmten Zeitpunkt besitzt, die Kommunikation nach einem Schlüsselaustausch nur dann weiter dekodieren kann, wenn er auch mindestens eine Verbindung (oder ein Gerät) auf jedem der zusätzlich verwendeten Pfade "besitzt".

6.6 Autonome Netzerhaltung des Systems

Unter der Netzerhaltung wird verstanden, dass die Eigenschaften des Netzwerkes, wie sie nach dem Imprinten bestehen, während des Betriebes erhalten bleiben. Die Haupteigenschaft hierbei ist die Zusicherung, dass zwischen zwei beliebigen Geräten immer mindestens s Pfade existieren und das Netz damit einen s-fachen Zusammenhalt besitzt.

Sobald diese Eigenschaft verletzt ist, ist es mittels des normalen Verbindungsaufbaus, wie es im Kapitel 4.3 beschrieben wurde, nicht mehr möglich, das Netzwerk wieder in eine Form zu bringen, in der diese s Pfade existieren.

Gründe für eine Veränderung des Netzwerkes sind:

- Ausfall eines Gerätes auf Grund zu Neige gehender Energieversorgung

- Entfernen eines Gerätes aus dem Kommunikationsbereich des Netzwerkes
- Ausschalten eines Gerätes
- Ausschluss/Revocation eines Gerätes (Kap. 6.9)
- Defekte oder Zerstörung eines Gerätes

Während das Gerät bei den ersten drei Fällen frühzeitig sein baldiges Ausscheiden aus dem Netzwerk feststellen und geeignete Massnahmen ergreifen kann, ist dies bei den weiteren Beispielen auf Grund von sofortigen Ausfällen, Fehlern oder bösartigen Angreifern nicht gegeben. Dies führt dazu, das für alle möglichen Fälle die jeweils geeignete Methode zur Selbstheilung des Netzes gewählt werden muss.

Im Folgenden wird zuerst beschrieben, wie – durch ein anfangs gewähltes Hinzufügen von zusätzlichen Verbindungen – eine Fehlertoleranz des unmodifizierten Netzwerkes erreicht werden kann. Danach wird ein geordneter Austritt eines Gerätes aus dem Netzwerk betrachtet und letztendlich die Möglichkeiten bei einem unvorbereiteten Ausfall eines Gerätes analysiert.

All diese Fälle werden – zusammen mit zugehörigen Beweisen – genauer in [26] behandelt.

6.6.1 Hinzufügen weiterer pseudoimprinteter Kanten

Ein wichtiges Vorgehen ist das Absichern des Netzwerkes gegenüber einer gewissen – zu Beginn festgelegten – Anzahl von Knotenausfällen. Dadurch kann einerseits ein späterer Recoverymechanismus das Netz "heilen", ohne die Sicherheitskriterien zu gefärden, des Weiteren kann, selbst wenn keinerlei autonome Algorithmen verwendet werden, ein funktionsfähiger Betrieb nach Ausfall eines Gerätes bis zu einer "Reparierung" durch ein Masterdevice (siehe Kapitel 6.7) sichergestellt werden.

Um f Knotenausfälle zu tolerieren, würde es ausreichen, neue Geräte – anstatt an s Geräte – an insgesamt $s + f$ Geräte zu imprinten. Das Resultat wäre ein $s + f$-fach zusammenhängendes Netz; es existieren demnach zwischen zwei beliebigen Geräten immer mindestens $s+f$ Pfade. Da ein Gerät nur auf jeweils einem Pfad vorkommen kann, bedeutet dies, dass pro defektem Gerät höchstens ein Pfad weniger existieren kann. Letzendlich sind somit selbst bei f Ausfällen immer noch mindestens s Pfade zwischen beliebigen Knoten vorhanden.

Diese weiteren f Verbindungen lassen sich zwar ebenfalls über einen physischen Kontakt herstellen, jedoch führt dies – besonders bei einer angestrebten hohen Fehlerresistenz – zu einem teils erheblichen Mehraufwand für den Benutzer.

Um das physische Imprinten von neuen Geräten an f weitere Geräte zu vermeiden, wird nun ein Verfahren vorgestellt, mittels dessen die Konstante f ohne weiterem Aufwand bei der Einbringung frei gewählt werden kann: Direkt nach dem Imprinten eines Gerätes hat dessen Schlüsselspeicher insgesamt s Einträge. Nun reagiert es solange auf jedes Discovery-Paket mit einer Pfadsuche, bis insgesamt $s + f = z$ Verbindungen erstellt sind. Währenddessen signalisiert es dem Benutzer, dass es gerade diese notwendigen Verbindungen aufbaut und währenddessen keine weiteren Geräte in das Netzwerk eingebracht werden dürfen. Hierbei ist anzumerken, dass zwischen dem letzten Imprinten und dem Vorhandensein der ersten zusätzlich erstellten Verbindung ein Knotenausfall von einem der

imprinteten Partner zu einer Netztopologie führt, welche verhindert, dass das neue Gerät weitere Verbindungen erstellt und ein wiederholtes Einbringen in das Netzwerk erfordert.

Die neuen Verbindungen werden im Schlüsselspeicher mit dem Typ "Pseudoimprintet" hinterlegt. Dies bedeutet, dass diese Verbindungen zwar über einen Verbindungsaufbau erzeugt wurden (und damit eine Kompromitierung bei Übernahme des Netzes möglich wäre), sie jedoch den gleichen Richtlinien wie imprintete Verbindungen unterliegen: Sie dürfen nicht (zufällig) gelöscht werden, da dadurch die Integrität des Netzes zerstört werden kann.

Eine weitere Eigenschaft dieses Vorgehens ist die mögliche Zusicherung einer Verbindungserstellung, selbst bei aktiven Versuchen von getamperten Geräten, dies zu verhindern: Werden insgesamt $z = s + f + a$ Verbindungen erzeugt, so existieren – selbst wenn f Geräte ausgefallen sind und a Angreifer den Verbindungsaufbau behindern – noch s korrekte Pfade ohne Angreifer. Durch eine Wahl von anderen Pfaden bei fehlgeschlagenen Verbindungsaufbauten lässt sich somit eine Angreiferresistenz erreichen.

6.6.2 Abmelden existierender Geräte

In diesem Kapitel wird das kontrollierte Entfernen von Geräten behandelt, ohne auf Eigenschaften wie einen z-fachen Zusammenhalt, wie sie im vorherigen Kapitel vorgestellt wurde, zu bestehen.

Zuerst wird die Art einer imprinteten Verbindung betrachtet: Wird ein Gerät mit einem anderen physisch verbunden, so bildet die entstehende neue (logische) Verbindung eine gerichtete Kante aus. Ein Gerät, welches ordnungsgemäss imprintet ist, besitzt demnach nach dem Einfügen in das Netzwerk s ausgehende Kanten. Da dies bei den ersten s Geräten im Netzwerk (auf Grund fehlender Partner beim Einbringen) nicht möglich ist, handelt es sich bei deren ersten Kanten untereinander um bidirektionale Verbindungen.

Will ein Gerät aus dem Netzwerk ausscheiden, so sendet es an alle Geräte, die sich bei ihm (pseudo-) imprintet haben (siehe Abbildung 6.10), eine Nachricht über ihren Austrittswunsch mit einer Liste von Geräten, an welchen dieser sich selbst (pseudo-) imprinted hat. Bei bidirektionalen Verbindungen wäre die korrespondierende Geräte-ID demnach einerseits in der Liste der selbst imprinteten Geräte, andererseits wird dieses Gerät auch benachrichtigt.

Empfängt ein Gerät nun eine solche Liste, so sucht es darin nach einem Gerät, zu dem es selber nicht imprintet ist. Existiert dies nicht, so ist das Netzwerk derzeit kleiner $s + 2$ Geräte und bereits vollvernetzt. Hat es jedoch ein solches gefunden, so baut es – solange das ausscheidende Gerät noch da ist und einen zusätzlichen Pfad zwischen den beiden Geräten darstellt – eine (pseudoimprintete) Verbindung zu diesem auf und meldet dem beendenden Gerät, dass es alle Vorkehrungen getroffen hat. Dieses kann sich – aus Sicht dieses einen Gerätes – nun beenden.

Zweck dieses Vorgehens ist, dass nach Austritt eines Gerätes eine Netzstruktur vorliegt, die auch vorliegen würde, wenn dieses Gerät nie existiert hätte: Alle später hinzugefügten – und in das nun entfernte Geräte imprinteten – Geräte sind nun nicht an dieses, sondern statt dessen an ein bereits *vorher* existierendes Gerät imprintet.

Während dieses Verfahren einen sicheren Austritt von Geräten aus dem Netzwerk ermöglicht, bestehen weiterhin folgende Probleme:

Nicht erreichbare Knoten Ist ein Gerät in der Zeit, in welcher das ausscheidende Gerät

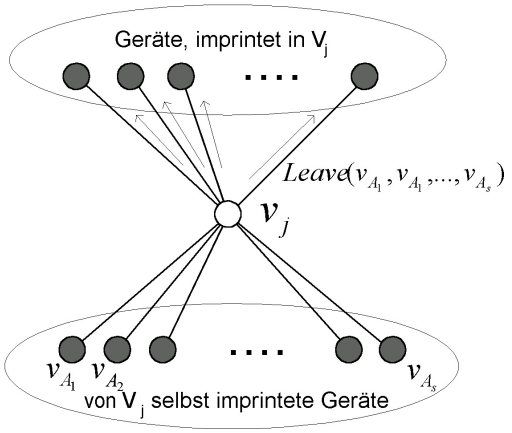

Abbildung 6.10: Abmelden eines Gerätes im Netzwerk

dies mitteilt, nicht verfügbar (z.B. ein mobiles Gerät), so ist für dieses Gerät ein geordnetes Ausscheiden nicht möglich

Nicht sofort mögliche Linkerzeugung Wähend zwar immer mindestens s Pfade zwischen beliebigen Geräten existieren, bedeutet dies – bei Verwendung probabilistischer Pfadsuchen, wie in Kapitel 6.4 beschrieben – nicht ebenfalls die sichere Findung von verwendbaren Pfaden. Aus diesem Grund kann es eine beliebige Zeit dauern, bis alle notwendigen Verbindungen ordnungsgemäss etabliert wurden und das Gerät sich ausschalten darf. Wird ein $z > s$ facher Zusammenhalt verwendet, so können bei einem Ausschalten jedoch immer noch maximal $z - s$ Verbindungen auf den Geräten vorgemerkt, aber noch nicht realisiert sein, ohne das Netzwerk zu gefährden.

Gleichzeitige Abschaltung mehrerer Geräte Empfängt ein Gerät, kurz nachdem es selber die Listen zum eigenen Austritt verschickt hat, die Austrittsankündigung von einem der in der Liste befindlichen Geräte, so kann dies zu Inkonsistenzen führen, wenn das zweite (ältere) Gerät sich letztendlich beendet, bevor alle Empfänger der Austrittsliste des ersten Gerätes ihre Verbindungen nachgeführt haben.

Abrupte Abschaltung bzw. Zerstörung Dieses Verfahren erfordert von Geräten, dass sie – zwischen dem Wissen ihrer Abschaltung und der tatsächlichen Abschaltung – eine gewisse (und teilweise auch längere) Zeit noch aktiv im Netzwerk verbleiben. Während dies bei einem benutzergeführten "Herunterfahren" oder einem gemessenen Batterieende noch möglich ist, ist diese Anforderung oftmals nicht möglich. Aus diesem Grund wird im nächsten Kapitel ein Verfahren zur nachträglichen Reparatur von Netzwerken vorgestellt.

6.6.3 Erkennung von Knotenausfällen

Dieses Verfahren arbeitet ähnlich dem im letzten Kapitel vorgestellten, jedoch muss hier die Liste der imprinteten Geräte verteilt eingesammelt werden. Es wird ein $z > s$ ver-

langt, da sonst bereits bei dem Ausfall eines einzigen Gerätes eine Verbindung zwischen bestimmten Geräten nicht mehr möglich wird.

Erkennt ein Gerät A – zum Beispiel mittels mehrerer nicht beantworteter Ping-Nachrichten – den Ausfall eines anderen Gerätes X, so meldet es dies über einen verschlüsselten Broadcast an das Netzwerk. Nun antworten alle Geräte, an die sich X imprintet hat (also alle Geräte, welche von X eine eingehende oder bidirektionale Verbindung besitzen), mit einer Broadcastmeldung.

Der weitere Ablauf ist der gleiche wie im vorherigen Kapitel, nur mit den folgenden Einschränkungen:

- Nicht alle notwendigen Geräte empfangen die Anfrage. Dies bedeutet, dass – sollte keine nützliche Antwort empfangen werden – die Anfrage von A ggf. mehrfach gesendet werden muss.

- Ein Angreifer, der einen Knoten – zu dem sich X imprintet hat – übernommen hat, verweigert eine Antwort. Da es in einem funktionierenden Netzwerk weniger als s Angreifer gibt (denn sonst gilt das Netzwerk als übernommen), reicht es hierbei aus, $z > 2 \cdot s$ zu wählen, um Angriffe dieser Art ohne Schaden zu überstehen.

- Ein Angreifer, der einen beliebigen Knoten übernommen hat, sendet ohne Grund seine eigene ID. Um diesem Angriff vorzubeugen, kann es sinvoll sein, anstatt zu nur einem einzigen der sich meldenden Geräten, zu allen eine Verbindung aufzubauen. Dies führt dann zu einem ggf. zu dicht vernetzten Netzwerk, als es eigentlich notwendig wäre.

- Ein Angreifer, der einen beliebigen Knoten übernommen hat, sendet ohne Grund fremde IDs. Wird der obrige Ansatz verwendet, so könnte ein Angreifer durch Senden aller IDs des Netzwerkes den fragenden Knoten mit so vielen Geräte-IDs "überschwemmen", so dass er entweder zur Kommunikation notwendige andere Verbindungen verwerfen muss oder die zur Netzrestauration benötigten Verbindungen nicht mehr erstellen kann. Dem kann vorgebeugt werden, indem bei einem folgenden Verbindungsaufbau der Grund desselben mitübermittelt wird. Ein korrekt funktionierendes Zielgerät kann dadurch den Verbindungsaufbau abbrechen und ggf. eine Meldung über einen internen Angreifer aussenden.

- Ein Angreifer, der einen beliebigen Knoten übernommen hat, sendet ohne Grund unbekannte IDs. Dies stellt einen möglichen DoS Angriff dar, denn die einzige Möglichkeit, die Existenz eines Gerätes zu prüfen, ist das Finden eines Pfades zu diesem. Während es bei einem lokal nachgebildeten Subgraphen ohne weitere Nachrichten möglich ist, die Erreichbarkeit eines Gerätes über s Pfade (was die Existenz impliziert, jedoch nicht die Rückrichtung) zu prüfen, sind bei einem 2-Hop-Protokoll bis zu m Nachrichten nötig. Hierdurch kann ein Angreifer bei einem ausgefallenen Knoten (und damit durchschnittlich z darin imprinteten Geräten) durch Aussenden eines einzigen Paketes das Verschicken von $z \cdot m$ Paketen auslösen. Eine Abschwächung dieses Angriffes wäre, dass Knoten, welche den Antwortenden wie den Fragenden kennen, dies bestätigen und damit eine solche Bestätigung im Sinne eines Leumundes explizit verlangt wird. Dieses Verfahren würde jedoch dazu führen,

dass die Netzintegrität zwar mit einer sehr hohen Wahrscheinlichkeit erhalten bleibt, jedoch nicht zugesichert werden kann.

Ebenso wie im letzten Kapitel ist hier auch das Ausfallen mehrerer Geräte gleichzeitig ein Problem. Da Informationen über Verbindungen *nur* auf den jeweiligen Endpunkten gespeichert sind, ist es bei Ausfall von zwei Geräten nicht mehr möglich, festzustellen, ob zwischen diesen Geräten eine Verbindung vorhanden war. Eine mögliche Lösung wäre das Verteilen dieser Informationen auf weiteren Knoten, welche dies im Falle eines Ausfalles melden oder selbst geeignete Aktionen zur Netzwerkmodifikation initiieren.

6.6.4 Sichere Heilung von Knotenausfällen

Dieses Kapitel umreisst das in [26] vorgestellte letzte Verfahren zum Akzeptieren und autonomen Heilen von bis zu c gleichzeitigen Knotenausfällen bei Vorhandensein von bis zu d kompromittierten Geräten.

Hierfür wird für jedes Gerät eine Liste aller ausgehenden (pseudo-)imprinteten Kanten auf jeweils den z ($z = s + c + d$) Geräten gehalten, die Teil dieser Liste sind.

Erkennt ein Gerät den Ausfall eines anderen Gerätes, zu dem es eine ausgehende Kante besitzt, so muss dieses den Ausfall – wie auch in Kapitel 6.6.2 motiviert – heilen, indem es seine Verbindung mit dem ausgefallenen Gerät durch eine neue ersetzt, zu dem das ausgefallene selbst imprintet war. Diese Informationen erhält es, in dem (über eine Gruppennachricht) alle Besitzer der replizierten Kantenmenge aufgefordert werden, dem anfragenden Gerät authentifiziert diese Liste zukommen zu lassen. Selbst wenn hierbei $(c-1)$ weitere Geräte ausgefallen sind und d Angreifer falsche Listen senden, so empfängt das anfragende Gerät immer noch s korrekte (und gleiche) Listen, welche mittels einer Mehrheitsentscheidung ausgewählt werden. Der weitere Heilungsprozess erfolgt wie in Kapitel 6.6.2 beschrieben mit der Eigenschaft, dass – wenn der korrespondierende Zielknoten ebenfalls aufgefallen ist – dieses Verfahren rekursiv angewendet werden kann.

Eine genaue Analyse des Verfahrens ist in [26] enthalten.

6.7 Geführte Netzerhaltung mittels Masterdevice

Sollte keine oder eine nur probabilistische Methode zur autonomen Netzerhaltung gewählt worden sein, so kann es im laufenden Betrieb mit Geräteausfällen zu einer Verringerung des Netzzusammenhaltes kommen. Das Masterdevice kann – so es verwendet wird – in diesem Fall auf Grund des absoluten Vertrauens der anderen Geräte (selbst bei einem Netzwerkzusammenhalt kleiner s) die Netzwerkstruktur wieder bereinigen.

Hierfür ermittelt es zuerst über Abfragen den aktuellen Netzaufbau, berechnet danach die notwendigen weiteren Verbindungen (da das Masterdevice nicht den Ressourcenbeschränkungen unterliegt, ist dies möglich) und veranlasst eine Verbindungserstellung zwischen den notwendigen Geräten gemäss des im Kapitel 6.3.2 beschriebenen Verfahren.

6.8 Gruppenkommunikation

Ein gewisser Anteil der Kommunikation erfolgt mittels Broadcasts. Darunter fallen unter anderem die Discovery-Meldungen der einzelnen Knoten, etwaige Statusmeldungen, Mel-

dungen bei der autonomen Netzerhaltung, Meldungen höherer Schichten von geringerer Sicherheitsrelevanz oder Nachrichten von Geräten mit zu geringem Schlüsselspeicher, um nur über Verbindungen zu kommunizieren.

Diese Gruppennachrichten können auf Grund der Verwendung symmetrischer Kryptographie nicht authentifiziert (im Sinne, dass die Urheberschaft einem einzelnen Gerät zuzuordnen ist) abgesendet werden.

Einem aussenstehenden, passiven Angreifer darf nicht die Möglichkeit gegeben werden, all diese Nachrichten mitzulesen oder sogar gefälschte Nachrichten zu senden. Ein solcher Angreifer hat keinerlei Geräte (mit teilweise erheblichen Aufwand) kompromittiert und analysiert rein den verschlüsselten Datenstrom. Daher wird – wie auch bei [2] verwendet – ein gruppenweiter Schlüssel verwendet, mittels dessen nur Mitglieder im Netzwerk Gruppennachrichten senden und empfangen können.

Verschlüsselt wird hierbei mittels eines einzigen gruppenweiten Schlüssels, welcher – im Falle einer Änderung – über die sicheren Verbindungen mittels eines Wellenalgorithmus in $O(n)$ an die Geräte übertragen wird. Dadurch wird ausgeschlossen, dass Geräte, welche einmal Teil des Netzwerkes waren, aber (z.B. wegen eines Ausschlusses) aus dem Netzwerk wieder entfernt wurden, weiterhin Gruppennachrichten lesen können.

Schlüssel werden hierbei entweder gesteuert von dem Masterdevice, einem beliebig bestimmten "spezialisierten" Gerät oder autonom von allen Geräten ausgegeben. Hierbei wird jeder Schlüssel mit einem zusätzlichen Versionsfeld versehen. Dieses Feld wird bei der Erstellung eines neuen Schlüssels inkrementiert und im Netzwerk wird jeweils der Schlüssel mit der höchsten Version verwendet. Dies wird erreicht, indem neu generierte Gruppenschlüssel an bekannte Geräte geschickt werden. Bei Empfang eines solchen Schlüssels mit gleicher Version wird der Schlüssel verwendet, welcher von dem Knoten mit einer höheren ID ausgegeben wurde. Bei einer neueren Version wird der Übertragene von da an verwendet. Änderungen werden danach entsprechend dem Wellenalgorithmus im Netz propagiert. Die Generierung eines neuen Schlüssels erfolgt zufällig in einem vorgegebenen Zeit- und Nachrichtenintervall.

6.9 Revocationmechanismen

Während Geräte, welche sich vollständig in Benutzerhand befinden, sich ohne weitere Probleme mit den in den Kapiteln zur Netzerhaltung beschriebenen Verfahren entfernen lassen, ist dies bei Geräten, welche kompromitiert wurden, nicht der Fall. Bei diesen ist davon auszugehen, dass sämtliche Schlüssel nicht nur auf dem Gerät, sondern auch in Händen eines Angreifers sind. Demnach ist weder eine Mitarbeit beim Austritt noch ein Auslaufen von Pings auf den erzeugten Verbindungen zu erwarten.

Da gerade die Kompromittierung explizit als möglich eingestuft wird, müssen in diesem Fall geeignete Maßnahmen getroffen werden. Neben dem Neuinitialisieren des gesamten Netzwerkes werden dabei im Folgenden zwei Möglichkeiten vorgestellt. Die erste Möglichkeit beschreibt den sofortigen Ausschluss eines Gerätes mittels eines Masterdevices. Die zweite Möglichkeit zeigt das autonome Entfernen mittels der s imprintenden Geräte auf.

6.9.1 Revocation mittels Masterdevice

Dieses Verfahren nutzt – wie auch bei der Netzerhaltung (Kapitel 6.7) – das vollständige Vertrauen der einzelnen Geräte in das Masterdevice. Der Ausschluss erfolgt dabei in zwei Schritten. Erstens wird das Gerät aus der Unicast-Kommunikation entfernt; dies entspricht dem Entfernen der Verbindungen. Zweitens wird es aus der Grupenkommunikation ausgeschlossen.

Erzwungene Partitionierung des Graphen

Um einzelne Geräte auszuschliessen, müssen in diesem Ansatz die kompromittierte Knoten von dem restlichen Netzwerk vollständig getrennt werden, d.h. dass keinerlei Verbindungen mehr zwischen dem weiter bestehenden Netzwerk und den kompromittierten Geräten liegen. Da jedoch während der Ausschlussprozedur einzelne Geräte nicht erreichbar sein und deren Verbindungen dadurch auch nicht überprüft werden können, kann die Menge der ausgeschlossenen Geräte größer der der kompromitierten Geräte sein.

Der Algorithmus arbeitet in vier Schritten:

- Stoppen sämtlicher Verbindungsaufbauten bis zum Ende des Entfernungsprozesses.
- Finden aller nicht erreichbaren oder kompromittierten Geräte
- Entfernen *aller* Verbindungen bei den restlichen Geräten zu dieser Menge
- Nachträgliches Hinzufügen von pseudoimprinteten Kanten, um die Netzwerkeigenschaften durch das Entfernen nicht zu gefährden.

Mobile Geräte, welche zu diesem Zeitpunkt nicht erreichbar waren, sind danach ebenso ausgeschlossen wie die kompromittierten und müssen nachträglich wieder in das Netzwerk – zum Beispiel durch ein Neuinitialisieren und Imprinten – eingebracht werden. Da die Existenz von Verbindungen aber nur bei den Kommunikationspartnern bekannt ist und den kompromittierten Geräten in dieser Hinsicht nicht vertraut werden kann, ist ein solches Vorgehen notwendig.

Ausschluss aus Gruppenkommunikation

Nachdem sämtliche Verbindungen zwischen weiterbestehenden und zu entfernenden Geräten gelöscht wurden, ist das Ausschliessen der alten Geräte aus der Gruppenkommunikation dahingehend trivial, dass nur ein neuer Schlüssel erzeugt werden und über das Netzwerk verteilt werden muss. Dies geschieht ohne Änderung gemäß des in Kapitel 6.8 beschriebenen Verfahrens.

6.9.2 Autonome Revocation

Um Geräte autonom aus dem Netzwerk auszuschließen, bedarf es zweier Techniken: Der Erkennung von sich fehlverhaltenden Geräten sowie dem darauf folgenden gemeinsamen Ausschluss dieses Gerätes. Für zweiteren Fall wird hier eine Möglichkeit vorgestellt, mittels derer die s Geräte, welche das Gerät in das Netzwerk durch Imprinten eingeführt haben, dieses auch wieder gemeinsam ausschliessen können. Dies bedeutet jedoch, dass bei Ausfall

der imprintenden Geräte ein Ausschluss nicht mehr möglich ist. Wird eine höhere Resistenz bezüglich Ausfällen verlangt, so müssen zusätzlich Ausschlussinformationen so verteilt werden, dass – selbst unter Beisein weiterer kompromittierter Geräte – sichergestellt wird, dass kein Gerät mehr wie eine einzige Ausschlussinformation bezüglich eines einzigen Gerätes erhält. Ware dies der Fall, so könnten in einem Netzwerk, in dem *weniger* als s Geräte kompromittiert wurden, korrekt funktionierende Geräte ausgeschlossen und somit von einem Angreiffer das Netz mehr oder weniger zerstört werden.

Erkennung von kompromittierten Geräten

Die Erkennung kompromittierter Geräte ist nicht einfach zu lösen, da nur Daten, die für das überprüfende Gerät selbst bestimmt sind, entschlüsselt werden können und auf Grund der fehlenden Nicht-abstreitbarkeit ("Non repudiation") Beweise eines Fehlverhaltens nicht weiterleitbar sind.

Während bei Geräten mit komplexen Eingabemöglichkeiten (wie zum Beispiel bei einer Fernbedienung) auch eine Meldung des Benutzers über auszuschliessende Geräte denkbar ist, ist dies bei den hauptsächlich im Bereich Heimautomatisierung verwendeten Geräten (Lampen, Schalter, Sensoren) nicht möglich.

Um sich fehlverhaltende Geräte zu erkennen, bestehen unter anderem folgende Möglichkeiten:

- Anzeige durch Benutzer mittels manueller Eingabe an vorhandener Peripherie
- Anzeige durch Benutzer mittels Eingabe an eingesteckter zusätzlicher Peripherie
- Meldung höherer Schichten (beispielsweise bei Angriffen auf das Rechtemanagement)
- Feststellung ungewöhnlicher Kommunikationsmuster (Verbindungsaufbaufrequenz, Anfragenanzahl, ...)
- Schlüsselmodifikationen bei Verbindungsaufbauten über bestimmte Geräte.
- Verweigerung der Mitarbeit
- Fehlerhafte Protokolle

Durch geeignete Auswahl von Erkennungsmassnahmen lässt sich zu jedem Gerät die Wahrscheinlichkeit einer Kompromittierung ermitteln und zwei Schwellwerte festlegen:

- Ein imprintetes Gerät gilt als möglicherweise kompromittiert, das heisst, bei gleicher Feststellung der $s - 1$ anderen zuständigen Geräten wird es ausgeschlossen.
- Ein Gerät gilt als höchstwahrscheinlich kompromittiert, das heisst, es wird lokal gegenüber höheren Schichten ausgeschlossen.

Ausschluss von Geräten durch imprintende Geräte

Es wird hier eine Möglichkeit vorgestellt, wie eine gemeinschaftliche Revocation mit folgenden zusätzlichen Kosten erreicht werden kann:

- Jedes Gerät erhält – neben seiner ID – noch eine weitere Kennnummer (im weiteren Revocation-ID genannt) mit der gleichen Länge eines Schlüssels (z.B. 16 Byte). Diese muss in der Schlüsseldatenbank zusätzlich abgespeichert werden.
- Geräte, welches ein anderes Imprintet haben, müssen ausserdem ein Geheimnis mit der gleichen Länge wie die eines Schlüssels und der Revocation-ID sowie eine Variable mit dem Wertebereich $1..s$ abspeichern.
- Während einer Pfadsuche bzw. des Schlüsselaufbaus muss sichergestellt werden, dass neben der gleichen ID auch die Revocation-ID auf allen Pfaden übereinstimmt.
- Das Imprinten benötigt an Stelle von $O(s)$ Nachrichten nun $O(s^2)$ Nachrichten.
- Bei Empfang eines Revocationpaketes müssen insgesamt s Secure Hashes berechnet werden.

Das Verfahren funktioniert nach folgendem Schema: Ein nicht imprintetes Gerät besitzt ursprünglich die Revocation-ID 0. Bei einem "Antrag" zum Imprinten übermittelt das neue Gerät N seine bisherige Revocation-ID $R_{N_{j-1}}$ sowie eine Liste $\{I_1, I_2, ...I_{j-i}\}$ aller bisher imprintenden Geräte. Beim Empfang dieser Liste generiert nun das gerade imprintende Gerät I_j ein Geheimnis S_i und ermittelt eine neue Revocation-ID $R_{N_j} = SHA(R_{N_{j-1}} \otimes S_j)$. Danach schickt es an alle vorherigen imprintenden Geräte $\{I_1, I_2, ...I_{j-i}\}$ ein Paket mit den Werten $R_{N_{j-1}}$ und R_{N_j}, wodurch diese ihre Schlüsseldatenbank anpassen können. Dieses Paket wird ausserdem über die sichere Verbindung an N geschickt, welches ebenfalls seine Werte anpasst. Das imprintende Gerät speichert daraufhin neben dem gemeinsamen Schlüssel und R_{N_j} noch zusätzlich das Geheimnis S_i und seine Position beim Imprint-Prozess j in der Schlüsseldatenbank ab und führt das Imprinten – wie in Kapitel 6.2 beschrieben – fort.

Am Ende dieses Vorgangs besitzen alle s Geräte, an die sich N imprintet hat, die gleiche Revocation-ID und es können dadurch neue Verbindungen erzeugt wurden. Des Weiteren besitzt N eine Revocation-ID, welche mittels einer Kette von Einwegfunktionen berechenbar ist. Im Falle von $s = 4$ zum Beispiel:

$R_{N_4} = SHA(SHA(SHA(SHA(S_1) \otimes S_2) \otimes S_3) \otimes S_4)$

Sollten nun alle imprintenden Geräte den Ausschluss bestimmen, so tauschen sie zuerst ihre Daten aus und überprüfen die Korrektheit. Danach verschickt (mindestens eines der Geräte) periodisch ein Paket mit den s Geheimnissen $s_{1..s}$ sowie der Geräte-ID von N.

Hierbei ist anzumerken, dass durch den Ausschluss wiederum die Protokolle bei der Erkennung von Knotenausfällen für die autonome Lebendighaltung (siehe Kapitel 6.6.3) zur Anwendung kommen müssen, da ein aktives Ausscheiden eines kompromittierten Gerätes aus Sicherheitsgründen nicht möglich sein kann.

Kapitel 7

Prototypische Implementierung

Im Rahmen dieser Diplomarbeit wurde eine prototypische Implementierung für einen Teil der in den vorherigen Kapiteln behandelten Verfahren erstellt. Im Folgenden werden zunächst die Ziele eines solchen Vorgehens analysiert und daraufhin die einzelnen Teilschritte erläutert.

7.1 Ziele der Implementierung

Für eine Implementierung der hier gesammelten Ergebnisse sprechen eine Vielzahl von Gründen.

Überprüfung der Funktionsweise von Algorithmen Gerade bei der Entwicklung weiterer Verfahren ist das Testen an einer vorhandenen Emulationsumgebung hilfreich, um eventuelle Fehler oder Probleme aufzufinden.

Statistische Messungen Messungen zur Belegung von Vermutungen oder Berechnungen ermöglichen eine spätere Evaluation (siehe Kap. 8) und eine Entscheidungsgrundlage neben theoretischen Erwägungen. Hierzu müssen neben der eigentlichen Implementierung noch zusätzliche Verfahren zum Sammeln dieser statistischen Information realisiert werden (siehe Kapitel 7.3.4 und 7.4).

Optimierung von Parametern Während in dieser Diplomarbeit hauptsächlich die grundlegenden Verfahren vorgestellt wurden, ist ein weiterer Teil bei der Realisierung das Setzen und Optimieren von Parametern (siehe hierzu Kapitel 7.6. Innerhalb dieses Projektes wurde dies durch Setzen aller Parameter innerhalb einer Konfigurationsdatei realisiert.

Plattformunabhängigkeit Da es keine spezielle Zielplattform gibt (es eignen sich zum Beispiel die TECO SmartIts wie die Berkeley Motes, aber auch die Entwicklung eines einzelständigen Microcontrollers zur Realisierung aller Sicherheitsaspekte bietet sich an), ist bei der Implementierung auf eine möglichst plattformunabhängige Entwicklung zu achten. Des weiteren soll eine Emulation auf PC-Basis unter Unix wie Windows möglich sein. Dies wird unter anderem durch die Verwendung von ANSI C realisiert. Des weiteren wurden keine plattformabhängige Datentypen sowie möglichst keine speziellen Befehle eines Microcontrollertyps verwendet.

Verwendung verschiedener Verfahren Gerade bei der Analyse verschiedener Verfahren (z.B. bei der Pfadsuche) muss es möglich sein, Codeteile austauschen zu können, ohne eine komplett andere Implementierung verwenden zu müssen. Aus diesem Grund empfahl sich ein modularer Aufbau der einzelne Algorithmen.

Optimale Codegrösse Für die Verwendung auf realen Microcontrollern mit starken Ressourcenbeschränkungen ist eine Minimierung der Codegrösse dringend notwendig. Hierfür werden einzelne Komponenten durch den Präprozessor ausgeblendet, um eine möglichst genau für den Einsatzzweck zugeschnittene Codegrösse zu erhalten.

Praktischer Einsatz Das letztendliche Ziel der Implementierung ist die Verwendung der hier behandelten Prinzipien in einem real existierenden Szenario. Da während dieser Diplomarbeit die Hardware noch nicht verfügbar war, ist nichtsdestotrotz das Vorhandensein von Software, welche mittels geringer Modifikationen an eine spezielle Hardware anzubinden ist, eine notwendige Vorarbeit.

7.2 Entwicklung einer Emulationsumgebung

Um die Sicherheitsschicht auf einem PC zu verwenden, musste zuerst eine Kommunikationsschicht erstellt werden, mittels derer verschiedene Prozesse (ein Prozess entspricht einem emulierten Gerät) miteinander kommunizieren können. Designkriterien waren:

- Mehrere Prozesse sollen sich auf einem einzigen Computer befinden können, um eine Grosszahl an Geräten ohne der Notwendigkeit dementsprechend vieler Computer zu emulieren.
- Prozesse müssen über mehrere Rechner verteilt werden können, um existierende Emulationsplattformen, wie das Network Emulation Testbed (NET) [18], verwenden zu können sowie um hohe Lasten zu verteilen.
- Sollten Technologien wie das NET nicht verwendet werden, so soll es möglich sein, den Datenverkehr zentral zu konfigurieren bzw. zu manipulieren (Latenzzeiten, Paketverluste, ...).

Ausgehend von diesen Anforderungen wurde folgendes Design verwendet. Für die einzelnen "Geräte" existiert eine Kommunikationsschicht mit simplen Primitiven zum Senden und Empfangen von Paketen. Diese Schicht ist – zur Messung der effektiven Codegrösse der Sicherheitsschicht – mittels eines Präprozessorbefehls zu reinen Funktionsrümpfen minimierbar. Intern agiert die Kommunikationsschicht als Client für den zentralen (und ein Netzwerk repräsentierenden) Server.

Der Server implementiert einen UDP-Server, welcher Pakete entgegennimmt und an alle verfügbaren Clients weiterreicht. Er implementiert hiermit die Semantik eines Broadcastmediums. Es wurde eine Kommunikation über UDP gewählt (eine TCP-Implementierung existiert ebenfalls), da diese der tatsächlichen Gegebenheit einer später auf Microcontrollerbasis existierenden Funkkommunikationsschicht am nächsten kommt. Da sämtliche Informationen über einen zentralen Server fliessen, lassen sich hierbei Laufzeiten, komplette und selektive Paketverluste mit einfachen Mitteln realisieren. Es ist anzumerken, dass –

da UDP nicht verbindungsbasiert ist – der Server Pakete an sämtliche Ports weiterleitet, von denen er seit seiner Initialisierung Pakete erhalten hat.

Um die zusätzliche Implementierung eines "Sicheren Kanals", wie er für das Imprinten verwendet wird, aber ansonsten denselben Beschränkungen eines Kommunikationskanals unterliegt, zu vermeiden, werden für die Emulation auch Imprint-Informationen über diesen Server verteilt.

7.3 Erstellung einer Administrationssoftware

Nachdem eine Kommunikationsschicht vorhanden ist, können einzelne Prozesse zwar miteinander kommunizieren, jedoch fehlt eine Koordination bzw. Administration durch den Benutzer. Einzelne Geräte müssen zuerst an s andere Geräte imprintet werden. Des Weiteren muss es möglich sein, das Verhalten des Netzes – abgesehen von Debug-Meldungen der einzelnen Prozesse auf einer Konsole – zu überwachen und analysieren. Dieses Kapitel zeigt den Aufbau einer hierfür erstellten Administrationssoftware.

7.3.1 Visualisierung der im Netz befindlichen Geräte

Für die Emulation wurden zufällig generierte Geräte-IDs verwendet. Daraus folgt die Notwendigkeit des Wissens über die sich derzeit im Netz befindlichen Geräte, um zum Beispiel einzelne Geräte anzusprechen.

Die Administrationssoftware stellt – für den Server – einen normalen Client dar, dass heisst, er empfängt – ebenso wie alle Geräte – die versendeten Pakete. Um eine Analyse der Pakete zu ermöglichen, werden innerhalb der Emulation die Pakete nicht tatsächlich verschlüsselt, sondern der jeweils verwendete Schlüssel zusätzlich zu den Daten verschickt. Auf diesem Weg ist es einem Aussenstehenden (wie der Administrationssoftware) einerseits möglich, die Pakete zu analysieren, andererseits kann die Sicherheitsschicht validieren, ob sie fähig wäre, das jeweilige Paket auch zu dekodieren und kann das Paket im negativen Fall verwerfen.

Durch den Empfang sämtlicher Pakete durch die Administrationssoftware können die versendeten Discovery-Nachrichten verwendet und dadurch eine (mittels Soft State) aktuelle Liste aller vorhandenen Geräte – bzw. Prozesse – generiert werden.

7.3.2 Einbringen und Imprinten neuer Geräte

Ist eine Liste aller Geräte vorhanden, so können – gemäss des in Kapitel 4.1 beschriebenen Verfahrens imprintete Verbindungen erstellt werden. Um ein eigenes User Interface auf jedem Gerät zu vermeiden, kann dies mittels einem speziellen Paket (speziell im Sinne eines besonderen Pakettypes, siehe Kapitel 5.4) geschehen, welches ein einzelnes Gerät zum Imprinten an ein anderes auffordert.

Um ein einfaches und schnelles Hinzufügen von Geräten zu ermöglichen, kann mittels der Administrationssoftware zusätzlich ein neuer Prozess gestartet werden. Hierfür wird eine Aufforderung an den Server gesendet, welcher daraufhin einen neuen Prozess startet. Ebenfalls ist das Entfernen von Geräten auf diese Weise möglich, um beispielsweise Ausfälle zu simulieren.

Letztendlich liess sich mit den beschriebenen Verfahren der automatische Aufbau eines Netzes durch ein nacheinander erfolgendes Erstellen eines Gerätes mit darauf folgendem Imprinten an entsprechend vielen ($min(s, n)$) bereits existierenden Geräten realisieren.

Neben diesen Methoden zum Netzaufbau stehen ferner Möglichkeiten der oberen Schichten in der Administrationssoftware zur Verfügung. So können – an beliebigen Geräten – Verbindungsaufbauten zu einem anderen Gerät initiiert und damit Pfadfindungsprotokolle analysiert werden.

7.3.3 Visualisierung der Netztopologie

Gerade für die Überprüfung folgender Eigenschaften ist eine Visualisierung der Netzwerkstruktur (also der vorhandenen Verbindungen) unabdingbar:

- Lifeness des Netzwerkes
- Korrektheit der Pfadsuche
- Verhalten bei Knotenausfällen

Da eine Infrastruktur fehlt, wurde eine Darstellung – wie in Abbildung 7.1 gezeigt – gewählt.

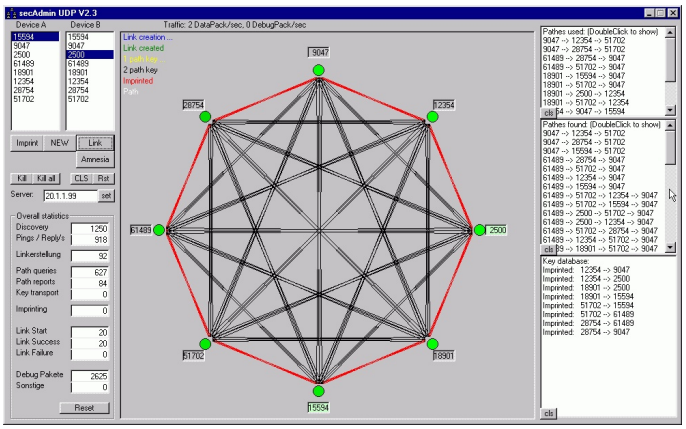

Abbildung 7.1: Simulationsoberflaeche

In ihr sind sichere Verbindungen zwischen zwei Geräten durch eine Doppelverbindung dargestellt: Je eine Kante, wenn auf einem der beiden Partner ein Schlüssel für diese Verbindung existiert. Auf diese Weise lässt sich die Existenz von Fehlern beim Verbindungsaufbau effektiv feststellen.

Farblich sind hierbei folgende Eigenschaften bei einer Verbindung dargestellt:

- Eine Verbindung befindet sich im Aufbau. Das heisst, es sind bereits Schlüssel auf einem bis $s - 1$ Pfaden ausgetauscht worden.
- Eine sichere Verbindung ist soeben vollständig aufgebaut worden, es kann also darüber kommuniziert werden.

- Eine Verbindung ist vom Typ Erzeugt.
- Eine Verbindung ist vom Typ Imprintet.
- Eine Verbindung ist vom Typ Pseudoimprintet.
- Soeben wurde ein Schlüssel auf einem Pfad über diese Verbindung verschickt.

Um diese Informationen zu erhalten, ist das Administrationsprogramm neben der Analyse des Datenverkehrs auf weitere Informationen angewiesen. Hierzu ist innerhalb der Sicherheitsschicht eine zusätzliche Möglichkeit zum Senden von Nachrichten vorgesehen, welche diese Informationen anstelle des Servers direkt an das Administrationsprogramm schickt. Hierdurch wird verhindert, dass diese zusätzlichen Pakete in einem etwaigen Traffic Shaper zu einer Fehlberechnung von Verzögerung und Verwurfsquote führen.

7.3.4 Sammeln von Statistikinformationen

Neben der Visualisierung der Netzwerktopologie ist weiterhin die Messung von statistischen Informationen enthalten. Sinnvolle Metriken für die Qualität eines Protokolls werden in Kapitel 8.2 erörtert.

Innerhalb dieser Oberfläche angezeigte Informationen enthalten unter anderem:

- Anzahl der Discovery Nachrichten
- Anzahl der Ping Nachrichten
- Anzahl der Nachrichten zur Linkerstellung
- Anzahl der Anfragen zur Pfadsuche
- Anzahl und Informationen von gefundenen Pfaden
- Anzahl von Paketen mit Schlüsselinformationen
- Anzahl der Verbindungen, welche mittels Imprinten erstellt wurden
- Gestartete Pfadsuchen
- Erfolgreiche Pfadsuchen (s knotendisjunkte Pfade gefunden)
- Nicht erfolgreiche Pfadsuchen
- Sonstige Pakete

Des Weiteren werden Meldungen der einzelnen Geräte angezeigt (zum Beispiel in Bezug auf den Schlüsselspeicher), um nicht die Kontrolle von n Konsolen zu benötigen.

Um den Netzwerkverkehr zu analysieren, ist eine herkömmliche Verschlüsselung auf Basis des gemeinsamen Geheimnis nicht möglich, da die Simulationsumgebung dabei nicht über die Fähigkeit des Entschlüsselns verfügen kann. Aus diesem Grund wurden die kryptographischen Funktionen innerhalb der Simulation durch Funktionen ersetzt, welche an die eigentliche Nachricht den verwendeten Schlüssel anhängen. Auf diese Weise ist die so behandelte Nachricht ebenso für die Administrationsoberfläche lesbar und kann von den Kommunikationspartner auf korrekte Schlüssel überprüft werden.

7.4 Erstellung einer Plattform für automatisierte Tests

Während das Administrationsprogramm zum benutzergeführten Testen der Sicherheitsschicht dient, sind gerade für das Messen vieler statistischer Informationen zur Entscheidung über die Qualität von Systemparametern andere Anforderungen vorhanden. Es wird vielmehr das Langzeittesten mit ständigem Hinzufügen von Geräten und häufigen Neuinitialisieren benötigt. Ferner müssen Gerätekonfigurationen von Test zu Test geändert werden. Dieser Abschnitt beschäftigt sich mit der Entwicklung eines Werkzeuges zum automatisierten Testen mit Hilfe von vorher definierten Testcases.

7.4.1 Erstellung von Testcases

Ein Testcase beschreibt den gesamten zu verwendenden Testablauf. Hierbei enthält er neben der Anzahl der insgesamt vorhandenen Tests und der Häufigkeit, wie oft jeder Test insgesamt ausgeführt wird (um Messabweichungen auf Grund von nichtdeterministischen Sicherheitsregeln auszuschliessen) Informationen zu den einzelnen Tests.

Ein Test beinhaltet neben Informationen zu den Parametern der Sicherheitsschicht (siehe Kapitel 7.6) unter anderem folgende – für den Ablauf des Tests notwendige – Informationen:

- Anzahl der Geräte, die in das Netzwerk eingebracht werden sollen.
- Anzahl der Geräte, an die ein neues Gerät imprintet wird.
- Pause zwischen dem Einbringen von zwei Geräten.
- Pause nach dem Einbringen aller Geräte.
- Anzahl der danach erfolgenden Messungen.
- Pause zwischen zwei Messungen.

Nach der Erstellung der Konfigurationsdatei lässt sich der Test mittels eines reinen Konsolenprogrammes unter Windows wie Unix – und somit zum Beispiel auch auf dem Network Emulation Testbed – durchführen.

7.4.2 Erstellen und Konfiguration virtueller Geräte

Da – im Gegensatz zum Administratiosprogramm – die einzelnen Parameter der Sicherheitsschicht bei den automatisierten Tests während des Kompilierens derselben nicht feststehen, muss eine Möglichkeit zum nachträglichen Konfigurieren der Geräte geschaffen werden. Hierzu wurde ein Weg gewählt, bei dem eine – abhängig von einem Präprozessorbefehl und somit eine Hardwareverwendung nicht beeinflussende – Konfigurationsdatei beim Start eines Gerätes geladen werden kann und konstante Definitionen in diesem Fall durch Variablen ersetzt werden. Das Testprogramm erstellt hierfür vor dem Beginn der Geräte-Erzeugung eine jeweilige Konfigurationsdatei.

Auch kann es bei sehr vielen Geräten (beispielsweise 256 Geräten) notwendig sein, dass diese nicht nur auf einem einzigen Computer erzeugt werden, sondern verteilt auf einer Liste von verfügbaren Computern. Dies wird über eine weitere Konfigurationsdatei

gelöst, in der sämtliche Adressen von verfügbaren Rechnern aufgelistet sind. Auf diesen Rechnern muss sich – um die Geräte starten zu können – eine laufende Instanz eines Servers befinden, der in diesem Fall jedoch nur für die lokale Erstellung und nicht für die Kommunikation verwendet wird.

7.4.3 Durchführung von Messungen

Während des Einbringens und danach in vorgegebenen Intervallen werden statistische Informatonen über das Netzwerk ausgegeben. Sämtliche Messungen werden hierbei in einer Datei zum späteren Import in einem Tabellenkalkulationsprogramm abgelegt, in welchem eine spätere Analyse der Messungen stattfinden kann. Um eine möglichst programmunabhängige Ausgabe zu liefern, wurde das Ausgabeformat CSV gewählt.

Nach Beendigung der Messungen werden automatisch wieder sämtliche Geräte entfernt, so dass ggf. ein folgender Messvorgang unbeeinflusst bleibt.

7.5 Verwendete Algorithmen

In der prototypischen Implementierung wurden aus Platzgründen nicht sämtliche hier behandelten Algorithmen verwendet. Basisalgorithmen zum Einsatz sind:

- Imprinten zweier Geräte
- Periodisches Aussenden von Discovery-Nachrichten
- Zwei-Hop Protokoll zum Verbindungsaufbau (Siehe Kap. 6.4.1) inklusive Lifeness

Hinzu kommen optionale Erweiterungen mit geringer Codegrösse:

- Erkennung von fehlerhaften Verbindungen und Knotenausfällen (Entfernung des jeweiligen Schlüssels)
- Priorisierungen von Verbindungen und daraus resultierend unterschiedliche Verwurfswahrscheinlichkeiten

Die letzte Klasse der implementierten Protokolle stellen Verfahren dar, die zwar eine hohe Codegrösse benötigen (oder in einer anderen Form ressourcenintensiv sind), die jedoch zur Überprüfung von Realisierbarkeit und Performance umgesetzt wurden:

- Pfadsuche mittels Ringsuche
- Pfadsuche mittels lokaler Subgraphbildung
- Versenden von Verbindungsinformationen in Discovery-Nachrichten
- Selbstheilungsversuche bei Erkennung von Knotenausfällen (nach Kapitel 6.6.3)
- Selbstheilung bei mehreren Knotenausfällen (nach Kapitel 6.6.4)

s	Der "Sicherheitsfaktor", wie in 4.3 beschrieben.
z	Anzahl der initial erzeugten (pseudo-)imprinteten Verbindungen.
$drop_{seldom}$	Schlüsselzahl, ab welcher Verbindungen zufällig verworfen werden können.
m_1	Maximale Schlüsselzahl, bis zu welcher zufällig neue Verbindungen erzeugt werden.
m_2	Maximale Schlüsselzahl, bis zu welcher Verbindungsaufbauwünsche angenommen werden.
$drop_{normal}$	Schlüsselzahl, ab welcher Verbindungen mit einer höheren Wahrscheinlichkeit verworfen werden.
m_3	Maximal verfügbarer Schlüsselspeicher.

Tabelle 7.1: Parameter für den Verbindungsaufbau

7.6 Resultierende Parameter

Nach Implementierung des Verbindungsaufbaus waren auf Grund der Ressourcenbeschränkungen des Schlüsselspeichers mehr Parameter benötigt als im ursprünglichen, theoretischen Protokoll. Eine Liste der notwendigen Parameter bezüglich des Verbindungsaufbaus ist in Tabelle 7.1 enthalten. Die Reihenfolge der Parameter ist so zu verstehen, dass ein Parameter kleiner oder gleich den darunter stehenden sein muss.

Hinzu kommen, so eine Pfadsuche verwendet wird, noch die maximale Länge der Pfade (dies entspricht der maximalen Tiefe bei Verwendung der Ringsuche) sowie die Maximalzahl aller lokal gehaltenen Pfade, die zur Findung eines Knotendisjunkten s-Tupels bei dem suchenden Gerät gespeichert werden müssen.

Werden in Broadcastpaketen Verbindungsinformationen mitgeschickt und somit ein lokaler Subgraph bei den Empfängern erzeugt, so sind folgende zusätzliche Parameter nötig:

- Grösse des Subgraphs (also z.B. Anzahl der gespeicherten Knoten)
- Grösse der Discoverynachrichten (also Maximalzahl der bekannt gemachten Verbindungen)
- Art der Verbindungen (z.B. nur imprintete Verbindungen, oder nur Verbindungen mit geringer Verwurfswahrscheinlichkeit)

Allgemein kommen noch Werte für Timeouts und Aktionswahrscheinlichkeiten hinzu, welche – bezüglich der erwarteten Netzwerkeigenschaften wie Grösse und Geschwindigkeit sowie bezüglich Geräteeigenschaften wie Prozessortakt - optimiert werden müssen. Die wichtigsten Parameter hierbei sind:

- Wahrscheinlichkeit zum Senden einer Discovery-Nachricht. Zu niedrige Werte verlangsamen die Lifeness, zu hohe erzeugen unnötige Netzlast.
- Wahrscheinlichkeit einer Verbindungserzeugung bei Erhalt einer Discoverynachricht. Zu niedrige Werte verlangsamen ebenfalls die Lifeness, während zu hohe Werte bei

Beschreibung	Code Modul [Byte]	Code Gesamt [Byte]	Globale Variablen [Byte]
Basisprotokoll	4611	4611	147
Erkennung von fehlerhaften Verbindungen und Knotenausfällen	282	4893	147
Priorisieren von Verbindungen	54	4947	147
Heilungsversuch bei Knotenausfällen	599	5546	162
Pfadfindung mittels Ringsuche	726	6272	3634
Pfadfindung mittels lokaler Suche	1760	7306	10085
Pfadfindung mittels lokaler Suche (Nur imprintete Pfade)	1800	7346	10085

Tabelle 7.2: Codegrössen

grossen Netzen zu einem Überfluten des Senders der Discovery-Nachricht mit Verbindungsanfragen und somit zu ungültigen und austimenden Schlüsseleinträgen führen.

- Verwurfswahrscheinlichkeiten, welche entweder – wenn sie zu gering gewählt wurden – zur Verhinderung neuer Verbindungsaufbauten führen oder – wenn sie zu hoch gewählt wurden – zu einer schlechten Speicherausnutzung und somit geringeren Performanz.

7.7 Resultierende Codegrössen

In diesem Kapitel werden diverse Codegrössen, wie sie bei der Übersetzung für einen Atmel ATMega16 mittels des Compilers AVR-GCC 3.2 entstanden sind, beschrieben.

Die wichtigsten Konfigurationen hierfür sind in Tabelle 7.2 zusammengefasst. Darin wird das Protokoll nach und nach erweitert. Die letzten drei Einträge zeigen jeweils eine Version mit allen anderen Optionen und einer Pfadsuche an Stelle des Zwei Hop Protokolls. Die extreme Vergrösserung der benötigten globalen Variablen ist durch das Anlegen einer Wissensdatenbank für empfangene Pfade sowie für den erstellten Subgraph notwendig. Diese Werte zeigen die fehlende Eignung dieser Algorithmen für einen ressourcenbeschränkten Microcontroller (ohne zusätzlichem Speicher), sind jedoch der Vollständigkeit halber aufgeführt. Hierbei sei angemerkt, dass die Pfadsuchealgorithmen bezüglich Speicherbedarf nicht optimiert wurden und der angegebene RAM-Bedarf bei den letzten drei Angaben rein von den festgelegten Parametern von Subnetzgrösse und Verbindungsanzahl abhängt.

Kapitel 8

Evaluation

Innerhalb der Evaluation werden die entwickelten Protokolle analysiert und Erkenntnisse aus den praktischen Emulationen aufgezeigt.

8.1 Theoretische Performance von Zwei-Hop-Protokollen

Da das Abwenden von der Findung aller möglichen Pfade hin zu einem probabilistischen Protokoll einen Einschnitt in die Verfügbarkeit von Verbindungen darstellt und somit die Eigenschaften des Netzwerkes – wie in [25] beschrieben – verändert, wird in diesem Kapitel die theoretische Leistungsfähigkeit eines Zwei-Hop-Protokolls betrachtet. Da in Kapitel 6.4.1 bereits gezeigt wurde, dass es in einem auf diesem Protokoll aufbauendem Netzwerk (welches nicht vollvernetzt ist) immer mindestens eine Verbindung existiert, welche erstellbar ist, wird hier nur noch die Wahrscheinlichkeit eines Verbindungsaufbaus zwischen beliebigen Geräten in einem zufällig vernetzten Netzwerk betrachtet.

Parameter, welche bei der Berechnung der Wahrscheinlichkeit eines Verbindungsaufbaus benötigt werden, sind:

n Die Gesamtgrösse des Netzwerkes.

s Der gewählte "Security level".

m Die Anzahl der Schlüssel bzw. der Verbindungen eines Gerätes.

Da Netze mit $n \leq m$ vollvernetzt sind, und somit $P_{Link} = 1$ gilt, werden im Folgenden Netze mit einer Grösse von $n > m$ angenommen.

Die Wahrscheinlickeit, dass ein Gerät einen Schlüssel zu einem beliebigen anderen besitzt, ist:
$P_{Known} = (\frac{m}{n-1})$: m Schlüssel für $n-1$ verbleibende Geräte im Netzwerk.

Ebenso gilt, dass ein Gerät, welches bereits einen Schlüssel zu einem dritten Gerät besitzt, mit folgender Wahrscheinlichkeit einen Schlüssel zu einem zweiten besitzt:
$P_{Proxy} = (\frac{m-1}{n-2})$: $m-1$ noch mögliche Schlüssel bei $n-2$ verbleibenden Geräten im Netzwerk.

Dementsprechend ist die Wahrscheinlichkeit, dass ein für ein drittes Gerät bekanntes Gerät keinen Schlüssel zu einem zweiten besitzt:
$P_{\overline{Proxy}} = 1 - P_{Proxy} = (1 - \frac{m-1}{n-2})$.

Die Wahrscheinlichkeit, dass bei einem Gerät genau i Zwei-Hop-Pfade zu einem anderen Gerät existieren, also genau i Partner das Zielgerät kennen (und somit $m-i$ Partner keinen Schlüssel zum Ziel besitzen), ist:
$P_{iPath} = \frac{n!}{(n-i)!\cdot i!} \cdot (1 - \frac{m-1}{n-2})^{m-i} \cdot (\frac{m-1}{n-2})^{i}$.

Letztendlich gilt ein Schlüssel genau dann als erstellbar, wenn zwischen den beiden Geräten mindestens s Pfade existieren. Die Gesamtformel lautet hiermit:
$P_{Link} = \sum_{i=s}^{m} \left[\frac{n!}{(n-i)!\cdot i!} \cdot (1 - \frac{m-1}{n-2})^{m-i} \cdot (\frac{m-1}{n-2})^{i}\right]$.

Abbildung 8.1 zeigt links für ein Netzwerk mit $s = 3$ die Wahrscheinlichkeit eines Verbindungsaufbaus für Schlüsselspeichergrössen von 10, 50, 100 und 150. Rechts ist für $m = 100$ eine Kurvenschar für verschiedene s-Werte dargestellt.

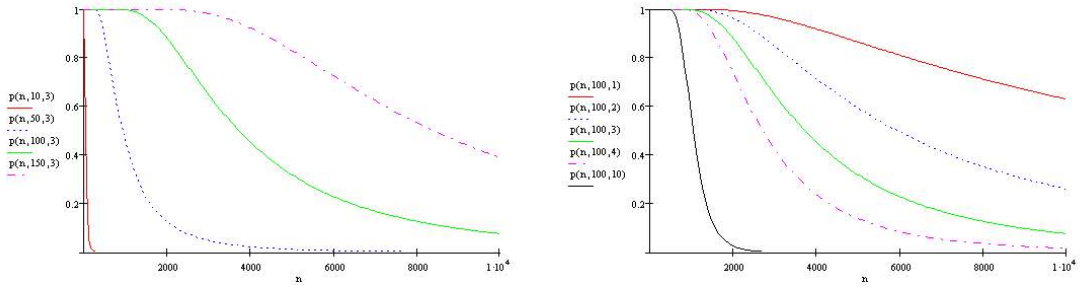

Abbildung 8.1: Erfolgswahrscheinlichkeit eines Verbindungsaufbaus

Man erkennt, dass bei einer Schlüsselspeichergrösse von 100 und eine Netzgrösse von 2000 noch eine Wahrscheinlichkeit von 88% existiert, ein Wert, der in einem sich ständig ändernden Netzwerk kein Problem darstellt. Bei einer Netzgrösse von 10000 ist für eine gleiche Wahrscheinlichkeit Speicher für durchschnittlich 223 Schlüsseln notwendig.

Aus dem rechten Diagramm lässt sich ableiten, dass – wenn man ein Szenario mit 4000 Geräten betracht, welche jeweils 100 Schlüssel speichern können – ein Security level kleiner drei gewählt werden muss, will man Aufbauwahrscheinlichkeiten von über 70% erreichen.

8.2 Messungen einzelner Protokolle

Neben diesen theoretischen Erwägungen wurden mittels der Testplattform (siehe Kapitel 7.4) Messungen durchgeführt, um diese Ergebnisse zu belegen. Abbildung 8.2 zeigt hierbei eine resultierende Messung für verschiedene s-Werte. Der Schlüsselspeicher betrug $m_2 = 22$ Schlüssel, wobei Schlüssel ab einer Menge von $drop_{normal} = 18$ verworfen wurden.

Bei den Messungen und Simulationen hat sich herausgestellt, dass – neben den bereits betrachteten Parametern – auch die Abstimmung folgender Variablen eine entscheidende Rolle spielt:

- Die Gesamtgrösse m_2 des Schlüsselspeichers, innerhalb welchem Verbindungswünsche angenommen werden.
- Die Schlüsselmenge $drop_{normal}$, ab welcher Schlüssel verworfen werden.

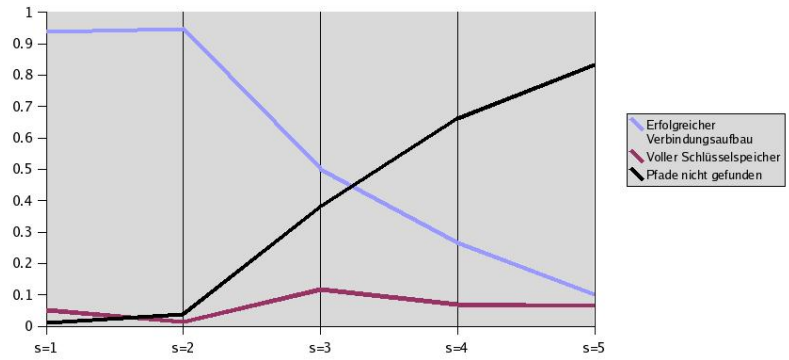

Abbildung 8.2: Messungen bei verschiedenen s

- Die Schlüsselmenge m_1, bis zu welcher neue Schlüssel zufällig erstellt werden.
- Die Wahrscheinlichkeit eines zufälligen Schlüsselaufbaus.

Sind diese Parameter nicht korrekt zueinander abgestimmt, so kommt ein weiterer Fehlerfall zum Zuge: Die Ablehnung eines Schlüsselaufbaus, da (auch wenn genügend Pfade gefunden wurden) der Schlüsselspeicher des Zielgerätes keinen freien Platz enthält.

Ein einfaches Beispiel hierfür wäre das Setzen der Aufbauwahrscheinlichkeit auf eins. Nach dem Senden einer Discoverynachricht werden danach durchschnittlich $n - drop_{normal}$ Verbindungswünsche auf dieses Gerät registriert, von denen jedoch nur durchschnittlich $(m_2 - drop_{normal})$ Schlüssel erzeugt werden. Die Mehrzahl aller Anforderungen schlägt demnach fehl. Dies hat im ersten Moment nur eine Auswirkung auf das für die Lifeness zuständige Protokoll, in welchem ein nicht geglückter Verbindungswunsch keinerlei Nachteile bewirkt. Dennoch führt dies dazu, dass auch in der Zeit nach der Discovery Nachricht, bis eine (zufällige) Verbindung wieder gelöscht wird, keine (von oberen Schichten initiierte) Verbindung zu diesem Gerät glücken kann. Des Weiteren führt dieses massive Aufbauen von Verbindungen zu einer Reduzierung der tatsächlichen Schlüsselmenge, da der Schlüsselspeicher neben den tatsächlich vorhandenen Schlüsseln auch Schlüssel speichern muss, welche sich noch im Aufbau befinden. Abbildung 8.3 zeigt auf der linken Seite die Wahrscheinlichkeiten geglückter Verbindungsaufbauten, Fehlschläge auf Grund zu voller Schlüsselspeicher auf Seite des Empfängers und Fehlschläge auf Grund von fehlenden Pfaden für verschiedene Aufbauwahrscheinlichkeiten. Dieser Test entstand in einer Simulation mit 64 Geräten, einem splitting faktor s von 2 und einem Schlüsselspeicher für 22 Geräte, wobei ab 20 Schlüsseln verworfen wird. Das rechte Diagramm veranschaulicht dabei die tatsächlich vorhandenen (vollständigen und sich im Aufbau befindlichen) Schlüssel pro Gerät.

8.3 Emulation auf dem Network Emulation Testbed

Um möglichst grosse Testszenarien durchführen zu können, wurde die Testplattform so entwickelt, dass sie (wie in Kapitel 7.4.2 beschrieben) innerhalb eines Clusters oder einer vernetzten Rechnermenge verteilt gestartet werden kann. Eine Ausführung der Tests im

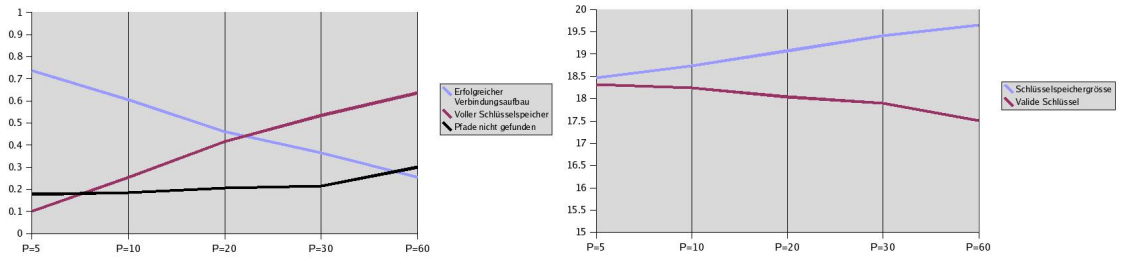

Abbildung 8.3: Messungen bei verschiedenen Verbindungsaufbauwahrscheinlichkeiten

Network Emulation Testbed [18] der Universität Stuttgart bringt der Evaluation hierbei folgende Vorteile:

Messung von Netzwerken mit grosser Gerätezahl Auf Grund der Verteilung der Rechenlasten ist es bei Verwendung des NET möglich, auch Netzwerke im Bereich mehrerer tausend Geräte zu simulieren. Dies ist besonders bei Aufgaben wie der Anpassung der Parameter und der Pfadsuchalgorithmen von Interesse.

Modellierung der Kommunikationsschicht Der grösste Vorteil des NET ist die Möglichkeit einer Modellierung der Kommunikationskanäle und somit ein realistisches Emulieren von bis zu 64 Geräten.

Da sich während der Evaluation Kommunikationsprotokoll wie Zielplattform noch in der Entwicklung befanden, konnten diese Tests nicht durchgeführt werden. Eine Simulation von vielen Geräten wurde durch den Wegfall von Echtzeitanforderungen, wie sie bei einer realen Emulation der Kommunikationsschicht vorhanden wären, vereinfacht.

Ferner ist, da von leistungsschwachen 8 Bit Mikrocontrollern ausgegangen wird, eine Simulation von grösseren Netzwerken mit mehreren hundert Geräten selbst auf einzelnen Rechnern möglich.

8.4 Einbindung der Zugriffskontrollschicht

Um eine möglichst weitgehende Simulation der angestrebten Visionen zu ermöglichen, wurde die von Jochen Noller entwickelte Zugriffskontrollschicht [19] an die Sicherheitsschicht erfolgreich angebunden. Informationen zu den daraus entstandenen Anforderungen an diese Schicht sind hierbei seiner Arbeit zu entnehmen.

Für die Anbindung standen unter anderem zwei Arten von Service Access Points zur Diskussion:

- Jeweils Funktionen zum Überprüfen und Erzeugen von neuen Schlüsseln, sowie eine Funktion zum Senden, sollte ein Schlüssel vorhanden sein.
- Eine einzige Funktion, welche – sollte noch kein Schlüssel existieren – eine Pfadsuche initiiert und danach das übergebene Paket sendet.

Um einen möglichst simplen und kleinen Service Access Point zu ermöglichen, welcher vor allem nicht von der verwendeten Sicherheitsschicht abhängig ist, wurde die zweite

Methode realisiert. Dies ermöglicht ein einfacheres Umstellen von Anwendungen, welche vorher einen ähnlichen Syntax ohne Sicherheitsschicht verwendeten.

Nachteil dieses Ansatzes ist: Sollen während der Pfadsuche (diese kann bis zu mehreren Sekunden betragen) oben liegende Schichten nicht blockiert werden, so muss dies durch zusätzliche Programmlogik bewerkstelligt werden. Neue Pakete dürfen in diesem Fall erst dann eingereicht werden, wenn die Sicherheitsschicht (durch das Setzen eines Flags) signalisiert hat, dass die Bearbeitung des alten Paketes (mit Erfolg bzw. mit einem Fehler) beendet ist.

Wird eine solche Semantik nicht benötigt, so ist die Verwendung eines blockierenden Aufrufes simpel zu realisieren und auch in bestehende Software einzubringen.

8.5 Mögliche Angriffe

In diesem Kapitel werden Angriffe und deren Folgen analysiert, wie sie bei einer Verwendung der hier vorgestellten Sicherheitsschicht auftreten können.

Nach einer Definition der möglichen Angriffsformen werden dabei folgende Ziele für einen Angreiffer betrachtet: Die Kommunikation an sich, die Verbindungserstellung, die Gruppenkommunikation und die Selbstheilung.

8.5.1 Arten von Angreifern

Angriffe lassen sich innerhalb von zwei Dimensionen betrachten: Der Kompromitierung des Netzes sowie der Aktionen des Angreifers.

Die Kompromittierung des Netzes lässt sich ihrerseits in drei Bereiche einteilen:

1. Es ist kein einziges Gerät übernommen oder ausgelesen worden. Dem Angreifer sind somit anfangs keinerlei Geheimnisse bekannt.

2. Es wurden zwischen 0 und $(s-1)$ Geräte übernommen. Dem Angreifer sind sämtliche Schlüssel der übernommenen Geräte und insbesonders auch der Schlüssel für die Gruppenkommunikation bekannt.

3. Es wurden mindestens s Geräte übernommen.

Die Aktionen eines Angreifers lassen sich ebenfalls in drei Bereiche unterteilen:

1. Er analysiert nur den Datenstrom, verhält sich sonst (bei einer gleichzeitigen Übernahme von Geräten) gemäss der Protokolle.

2. Er verweigert teilweise die Mitarbeit.

3. Er verhält sich beliebig, sendet teilweise fehlerhafte Informationen.

Um die Angriffe im eonzelnen zu beschreiben, wird folgender Syntax verwendet: $A(k, a)$ definiert einen Angreiffer, welcher das Netz nach Definition Nummer k kompromittiert hat und sich nach Typ a verhält. k und a können hierbei auch einen $\cdot$ enthalten, wenn alle Typen angesprochen werden. $A(2, 1)$ stellt zum Beispiel einen Angreifer dar, der zwischen einem und $(s-1)$ Geräte übernommen hat und empfangene Nachrichten nur analysiert, um weitere Informationen zu erhalten.

Angriffe des Types $A(3,*)$ werden von dem Netzwerk auf Grund dessen Eigenschaften nicht toleriert. Ein Beispiel für eine daraus resultierende Übernahme wird in Kapitel 8.5.6 erläutert.

8.5.2 Angriffe auf die Kommunikation

Angriffe auf die Kommunikation stellen Angriffe auf Schichten unterhalb der Sicherheitsschicht – insbesondere auf den Funkkanal – dar. Zu ihnen gehören Techniken wie das

- Analysieren der übertragenen Pakete
- Überfluten des Funkmediums ("jamming")
- Einspielen von verfälschten Paketen
- Einspielen von vorher abgefangenen Paketen ("Replay attacks")

Bei der Analyse wird hierbei rein von Angreifern des Types $A(1,*)$ ausgegangen, da – sollte der verwendete Schlüssel bereits durch eine Kompromittierung bekannt sein – ein solcher Angriff in später analysierte Kategorien fällt.

Um die Kryptoanalyse der empfangenen Pakete zu verhindern, wurden in Kapitel 6.5 eine zeit- oder nachrichtenzahlabhängige Änderung der verwendeten Schlüssel empfohlen. Durch ein entsprechend kurz gewähltes Intervall einer Schlüsseländerung lasst sich dieser Angriff damit verhindern.

Da direkte Angriffe auf das Funkmedium nicht Gegenstand dieser Arbeit sind, sei hier nur auf Techniken wie "Frequency Hopping" oder "Spread Spectrum" verwiesen, welche ein Stören des gesamten Funkkanals zu verhindern versuchen.

Die dritte Art der hier betrachteten Angriffe auf die Kommunikation stellt das Einspielen von Paketen dar. Da – ohne Kenntniss des verwendeten Schlüssels – das Erstellen der korrekten Prüfsumme nicht möglich ist, lässt sich die Wahrscheinlichkeit von durch Angreifer eingebrachte Pakete in diesem Fall durch geeignet grosse Wahl der Prüfsumme minimieren.

Das Wiederholen von empfangenen Nachrichten ist jeweils für die Kommunikation zwischen zwei Geräten und für die Gruppenkommunikation getrennt zu behandeln. Da bei der authentifizierten Kommunikation zwischen zwei Geräten die Frische einer Nachricht (mit Hilfe eines Paketzählers) realisiert wird, ist ein wiederholtes Einspielen eines solchen Paketes sinnlos, es wird bei dem Empfang von der Sicherheitsschicht verworfen. Anders sieht dies bei der Gruppenkommunikation aus, bei der die Eigenschaft der Frische nicht garantiert wird. Hier sind rein Nachrichten zu verwenden, bei denen ein mehrfacher Empfang innerhalb kurzer Zeit keinerlei Schaden für das Netzwerk anrichten kann. Da innerhalb eines anzugebenden Intervalls auch der Schlüssel für die Gruppenkommunikation geändert wird, ist eine maximale Zeit vorgegeben, in der eine gesendete Gruppennachricht wiederholt auftauchen kann. Auf diese Weise kann ein Angreifer zum Beispiel nicht tagsüber den Wert eines sicherheitsunkritischen Temperatursensors erfolgreich aussenden, den er nachts aufgezeichnet hat.

8.5.3 Angriffe auf die Linkerstellung

Angriffe auf die Linkerstellung lassen sich – gemäss des Angreiffertyps – nach ihrer Intention unterteilen. Da $A(1,*)$ bereits im vorherigen Kapitel und $A(3,*)$ in Kapitel 8.5.6 behandelt werden, sind hier (wie auch in den nächsten beiden Kapiteln) nur $A(2,*)$, also Angreifer mit übernommenen Geräten, zu betrachten.

Da bei $A(2,1)$ nur die Schlüsselanteile, welche über sie selbst geleitet werden, gelesen werden können, ist es einem solchen Angreifer nicht möglich, weiteres Wissen über das Netz zu erlangen.

$A(2,2)$ hat zusätzlich die Möglichkeit, Schlüsselanteile nicht weiterzuleiten. Wurde das Netz – wie in [26] empfohlen – mit $z = s + d + f$ Verbindungen aufgebaut, so stellt dieser Angriff nur ein Verzögern eines Verbindungsaufbaus dar, da von dem initiierenden Knoten eine andere Pfadmenge gewählt werden kann, in welcher sich kein Angreifer befindet. Wurde das Netz – wie im ursprünglichen Ansatz – nur mit s Verbindungen erstellt, so kann ein Angreifer den Aufbau von neuen Verbindungen so zwar verhindern, jedoch keinerlei weitere Informationen erhalten oder Verbindungen übernehmen. Gleiche Überlegungen gelten bei einer Verweigerung der Mitarbeit bei der Pfadsuche.

$A(2,3)$ kann, neben den bisher beschriebenen Aktionen, noch folgende Angriffsversuche unternehmen:

Verändern von Schlüsselanteilen Leitet ein einziger Knoten einen von ihm modifizierten Schlüsselanteil weiter, so wird (nach Zusammensetzen aller Anteile) der Zielknoten letztendlich einen anderen Schlüssel besitzen als der Initiierende. Das Ergebnis ist – wie im Falle einer Weiterleitungsverweigerung – ein fehlerhafter Verbindungsaufbau und damit die Notwendigkeit für die Auswahl anderer Pfade.

Senden falscher Pfadinformationen Sendet ein Angreifer auf einer Pfadsuche grundsätzlich eine Negativmeldung zurück, so ist das Ergebnis wiederum dasselbe wie bei einer Verweigerung der Weiterleitung von Schlüsselanteilen: Pfade, welche über den Angreifer führen, sind nicht verwendbar. Bei ständigen Positivmeldungen entspricht dies dem einschleusen weiterer (fehlerhafter) Pfade. Um den Aufwand bei späteren Schlüsselaufbauten zu minimieren und das Finden von kompromittierten Knoten zu vereinfachen, sei hier auf die Vorteile eines "Pathraters" verwiesen, mittels dessen sich Vertrauenswerte mit der Zeit aggregieren lassen und so eine Intrusion Detection ermöglichen. Nach dem Finden eines kompromittierten Knotens lässt sich dieser entweder ausschliessen, oder bei einer eigenen Pfadsuche ignorieren.

Ständiger Aufbau neuer Verbindungen Bei einem derartigen Angriff auf den Schlüsselspeicher kann ein einzelner Angreifer maximal zu jedem anderen Gerät genau eine Verbindung erstellen. Insgesamt führt dies für alle Angreifer jedoch zu einer Zahl von weniger als s Verbindungen pro Knoten, wodurch der Schlüsselspeicher der einzelnen intakten Geräte nicht beeinflusst werden sollte.

Weiterleiten von nicht existenten Schlüsselanteilen In dieser Angriffsform sendet das kompromittierte Gerät Pakete an das Opfer, welches den ersten Schlüsselanteil eines dritten Gerätes (in diesem Fall muss die Geräte-ID nicht einmal existieren) enthält. Daraufhin muss es einen neuen Eintrag in seinem Schlüsselspeicher erstellen. Um die Möglichkeit eines Überflutens mit beliebigen Schlüsselanteilen zu ver-

hindern, ist – neben einer Erkennung von Angriffen, die in diesem Fall relativ einfach realisierbar ist – folgende Vorkehrung zu treffen: Ein erster Schlüsselanteil wird nur dann angenommen, wenn nicht bereits über diesen Nachbarknoten g weitere erste Schlüsselanteile von Schlüsseln gesendet wurden, die weder erstellt, noch durch einen Timeout gelöscht wurden. Dadurch lässt sich dieser Angriff im schlimmsten Fall auf $((s-1) \cdot g)$ Einträge im Schlüsselspeicher reduzieren. Durch einen erfolgreichen Angriff liesse sich erreichen, dass ein intaktes Gerät keinerlei weitere Verbindungsanfragen von anderen Geräten auf Grund fehlender Ressourcen annehmen kann.

8.5.4 Angriffe bei der Gruppenkommunikation

Während in Kapitel 8.5.2 bereits die Auswirkungen von Replay Attacken betrachtet wurden, wird hier das Einschleusen weiterer Nachrichten betrachtet (also Angreifer vom Typ $A(2,2)$). Da Gruppennachrichten rein für die Gruppe authentifiziert sind und nicht für ein einzelnes Gerät, kann (sobald ein Gerät kompromittiert wurde) der Angreifer beliebige Nachrichten aussenden. Aus diesem Grund sollten sicherheitskritische Informationen nicht per Broadcast versendet werden. Diese wurden bei folgenden Protokollen verwendet:

Senden von Discovery Informationen Ein Angreifer kann hierbei beliebige Geräte-IDs aussenden und damit – insofern ein Protokoll mit Lifeness gewählt wurde – Verbindungsaufbauten von zufällig im Netz befindlichen Geräten veranlassen. Gerade bei einer Pfadfindung mittels Tiefen- oder Ringsuche kann dies zu einem erfolgreichen Angriff führen, da durch ein vom Angreifer ausgesendetes Paket ggf. hunderte von Pfadpaketen generiert werden.

Eine Möglichkeit, diesem Angriff entgegenzuwirken, ist das Entfernen der Discovery-Broadcasts (bzw. deren Ignorierung) und stattdessen die Verwendung eines Protokolls ähnlich der Zwei-Hop Protokolle im Verbindungsaufbau. Hierbei werden die Discovery Nachrichten mit einer Tiefe von zwei durch einen Wellenalgorithmus verschickt. Damit lassen sich Fehlinformationen Quellen zuordnen und damit Angreifer entdecken.

Senden von Revocationpaketen Da diese Pakete keine weiteren Nachrichten auslösen und vollständig lokal verarbeitet werden, sind hier keine dDoS Angriffe möglich. Da die Nachrichten selber noch durch Hashwerte gesichert sind, ist ein Missbrauch dieser Pakete ausgeschlossen.

Initiieren des Selbstheilungsprozessen Wie in Kapitel 6.6.4 beschrieben, führt das Senden einer Anfrage dazu, dass bis zu $(s+c+d)$ andere Geräte versuchen, eine neue Verbindung zu dem vom Angreifer genannten Gerät zu erstellen. Dies führt – wie bei den Discovery Nachrichten – zu einer ggf. sehr hohen Anzahl von Nachrichten pro eingeschleustem Paket. Um die Auswirkungen dieses Angriffes zu minimieren, ist folgende Strategie vorteilhaft: Empfängt ein Gerät eine Aufforderung zur Heilung, so überprüft es zuerst, ob das angegebene Gerät wirklich ausgefallen ist. Des Weiteren hält es die Heilungsinformationen nur eine gewisse Zeit nach einem Knotenausfall vor, so dass letztendlich nur direkt nach einem Knotenausfall ein solcher Angriff möglich ist. Zusätzlich ist das Abarbeiten dieser Aufgaben mit einer gerin-

geren Priorität sinnvoll, um einerseits die benötigte Bandbreite gering zu halten, andererseits dem Gerät Ressourcen für andere Aufgaben zu lassen.

8.5.5 Angriffe bei der Selbstheilung

Neben dem Fälschen der Broadcastnachrichten kann ein Angreifer ebenfalls die Liste der imprinteten Geräte fälschen und an den Anfragenden senden. Da diese Liste jedoch auf eine authentifizierte Weise übertragen wird, kann ein Angreifer vom Typ $A(2,*)$ höchstens $(s-1)$ gefälschte Nachrichten senden. Diese reichen jedoch – da ein Voting verwendet wird – nicht aus, um als valide erachtet zu werden.

Eine genaue Analyse der Angriffsmöglichkeiten bei den einzelnen Nachrichten für dieses Protokoll ist in [26] enthalten.

8.5.6 Übernahme eines Netzes

Um die Folgen einer Übernahme von s Geräten aufzuzeigen, wird in diesem Kapitel ein Beispielszenario vorgestellt, in dem am Ende – abgesehen von den $(z \cdot n)$ (pseudo-)imprinteten Verbindungen – nahezu sämtliche Verbindungen übernommen wurden.

Nachdem die s Geräte übernommen wurden, werden diese ausgelesen und stillgelegt. Die Schlüssel werden dabei auf einem ressourcenstarken Rechner abgelegt, welcher den weiteren Angriff führt. Dieser generiert daraufhin eine grosse Anzahl ($>> m$) weiterer virtueller Geräte, welche ebenso virtuell an die s übernommenen und im Netz befindlichen Geräte imprintet werden. Als Ergebnis wurde das Netzwerk um beliebig viele neue Angreifergeräte erweitert.

Im nächsten Schritt bauen diese Geräte Verbindungen zu allen nicht kompromittierten Knoten auf. Dies führt zu zwei Eigenschaften:

1. Die Angreifergeräte sind sehr stark vernetzt.

2. Die nicht kompromittierten Geräte müssen – auf Grund ihrer begrenzten Ressourcen – ständig Verbindungen verwerfen. Insbesondere kann davon ausgegangen werden, dass jede Verbindung zwischen zwei korrekt arbeitenden Geräten früher oder später verworfen wird, solange ständig neue Verbindungsanforderungen gestellt werden.

Auf Grund der ersten Eigenschaft ist die Wahrscheinlichkeit, dass alle s Pfade über mindestens einen der virtuellen Angreifergeräte führt, sehr hoch. Diese lässt sich noch durch geschickte Delegation der Verbindungen zwischen den Geräten erhöhen. In Verbindung mit der zweiten Eigenschaft werden die notwendigerweise neu aufgebaute Verbindungen zwischen nicht kompromittierten Geräten von dem Angreifer mit einer hohen Wahrscheinlichkeit übernommen.

Ab diesem Punkt kann der Angreifer ebenso Pfade, welche über diese kompromittierten Verbindungen führen, dechiffrieren und zur Übernahme weiterer Verbindungen nutzen. Des Weiteren kann er nun authentifizierte Nachrichten zwischen zwei Geräten einbringen.

Hierbei sei angemerkt, dass bei einem auf Mehrheitsentscheiden oder Voting basierenden Revocationmechanismus die Angreifer zusätzlich die Möglichkeit hätten, beliebige Geräte aus dem Netz zu nehmen.

Eine weitere – von diesem Szenario unabhängige – Methode zum Senden authentifizierter Nachrichten im Namen eines anderen Gerätes ist das Erstellen eines virtuellen Gerätes

mit der gleichen Geräte-ID wie der, in dessen Namen die Nachricht gesendet werden soll und ein Erstellen der Verbindung über die kompromittierten Geräte zum gewünschten Ziel der Nachricht.

Kapitel 9

Ausblick

In diesem Kapitel wird – nach einer Zusammenstellungen weiterer wünschenswerter Arbeiten – rückblickend eine Analyse dieser Arbeit erstellt.

9.1 Mögliche Erweiterungen und Verbesserungen

Einbringen einer Intrusion Detection Das hier vorgestellte Sicherheitsmodell basiert auf der Annahme, dass nur eine maximal vorgegebene Anzahl von Geräten in einem Netzwerk *gleichzeitig* kompromittiert sein darf. Demnach sind Verfahren, welche ein inkorrektes Verhalten einzelner Geräte erkennen und dadurch diese entweder lokal ausschliessen oder gar einen Ausschluss aus dem gesamten Netzwerk veranlassen, vorteilhaft. Als Erkennungsmasstäbe lassen sich unter anderem statistische Messungen über gesendete Pakete, versuchte Verbindungsaufbauten, sowie über eingebrachte Geräte verwenden. Des Weiteren sind zum Beispiel fehlerhaft weitergeleitete Pakete ein klares Indiz für eine Kompromittierung.

Analyse anderer Pfadsuchealgorithmen Die in dieser Arbeit vorgeschlagenen Protokolle sind entweder probabilistisch oder von hoher Komplexität. Ein zukünftiges Ziel ist die Entwicklung von Algorithmen zur Pfadfindung, welche die Eigenschaften des s-fachen Zusammenhaltes aktiv ausnutzen und so zu einer Komplexitätsreduktion beitragen. Beispielsweise lässt sich die Eigenschaft verwenden, dass – betrachtet man rein die durch das Imprinten entstandenen Verbindungen – sich die Richtung der Kanten auf mindestens s Pfaden maximal einmal ändert. Ebenfalls könnte eine Vergabe der Geräte-ID, abhängig von den zum Einbringen verwendeten Geräte, eine lokale Berechnung der Pfade erleichtern.

Tatsächliche Realisierung auf Mikrocontrollern Da bisher die Funktionalität dieser Algorithmen nur in Emulationen nachgewiesen werden konnte, ist ein weiterer Schritt die tatsächliche Herstellung eines Prototyps. Die derzeit angestrebte Zielplattform bei der Entwicklung von Prototypen stellen die TECO Smart-Its dar, welche auf einem PIC18 Mikrocontroller basieren. Aufgrund der ursprünglichen Strategie, die Sicherheitsschicht ebenso wie die Zugriffskontrollschicht auf einem ATMega16L prototypisch zu implementieren, muss jedoch der entwickelte Code vorher für die PIC Plattform portiert werden. Da die Smart-Its zum Abgabezeitpunkt jedoch

noch nicht verfügbar waren, war die Erstellung eines physikalischen Prototypes im Rahmen dieser Arbeit nicht möglich.

9.2 Fazit

Während dieser Diplomarbeit wurde eine Sicherheitsschicht entwickelt, welche den Anforderungen bezüglich Sicherheit und Ressourcenverbrauch in Szenarien wie der Heimautomatisierung Rechnung trägt.

Ausgehend von den dabei von Wacker et al. entstandenen Veröffentlichungen [25, 26] wurde das Prinzip des physikalischen Verbindens an eine vorgegebene Anzahl von Geräten in Form von Protokollen und Emulationen realisiert. Dabei entstand auch ein funktionsfähiger Prototyp auf Softwareebene, welcher für den Einsatz auf einem Atmel ATMega Mikrocontroller optimiert ist.

Parallel zu dieser Arbeit wurde ebenso eine Zugriffskontrollschicht für ressourcenbeschränkte Geräte von Jochen Noller entwickelt [19], welche auf den Service Access Points der Sicherheitsschicht aufbaut. Es ist festzustellen, dass diese Schichten bei Emulationen korrekt miteinander arbeiteten und verfügbare Semantiken erfolgreich genutzt werden konnten. Ebenso wurde bei der kooperativen Entwicklung der Schnittstellen ein Austausch von Anforderungen der korrespondierenden Schichten möglich.

Der primäre nächste Schritt ist die Realisierung der hier vorgestellten Algorithmen in einem grösseren Netzwerk, um grössere Szenarien unter den Echtzeitbeschränkungen der Kommunikationsschicht zu analysieren.

Literaturverzeichnis

[1] Embedded Security. Atmel Corporation. http://www.atmel.com/products/Embedded/.

[2] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenets. In *Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking and computing*, pages 156–163. ACM Press, 2001.

[3] Giuseppe Di Battista, Roberto Tamassia, and Luca Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4):302–340, 1999.

[4] Michael Brown and Donny Cheung. PGP in constrained wireless devices. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[6] I. Chlamtac and A. Faragó. A new approach to the design and analysis of peer-to-peer mobile networks. *Wireless Networks*, 5(1999/3):149–156, 1999.

[7] EIB - European Installation Bus. Webpage. http://www.eiba.com.

[8] ELV - Home Automation. Webpage. http://www.haustechnik.elv.de.

[9] A. Ephremides, J. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. In *Proceedings of the IEEE*, 1987.

[10] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communication Security (CCS-02)*, pages 41–47, November 2002.

[11] Shimon Even and R. Endre Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, December 1975.

[12] Jean-Pierre Hubaux, Levente Buttyan, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 146–155, 2001.

[13] J. Kohl and C. Neuman. The Kerberos network authentication service (v5). IETF RFC 1510, September 1993.

[14] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *In Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume volume 2139 of Lecture Notes in Computer Science, pages 310–331, Santa Barbara, California, 2001. Springer.

[15] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[16] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks, 2000.

[17] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[18] NET - Network Emulation Testbed. Webpage. http://net.informatik.uni-stuttgart.de/.

[19] Jochen Noller. Design, prototypische Implementierung und Evaluierung einer Zugriffskontrollschicht füer ressourcenbeschränkte Geräte, 2003.

[20] National Institute of Standards and Technology. Fips pub 140-1, security requirements for cryptographic modules, Jan 1994.

[21] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[22] Michael Schmidt. Blauzahnlücken: Angriffsmöglichkeiten bei bluetooth. *c't Magazin*, 11:186, June 2003.

[23] Home of the Smart-Its project. Webpage. http://smart-its.teco.edu.

[24] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th International Workshop on Security Protocols*, Lecture Notes in Computer Science, pages 172–194, 1999.

[25] Arno Wacker, Timo Heiber, and Holger Cermann. A key-distribution scheme for wireless home automation networks. to appear in Proceedings of IEEE CCNC 2004, Jan 2004.

[26] Arno Wacker, Timo Heiber, Holger Cermann, and Pedro José Marrón. A faulttolerant key-distribution scheme for wireless ad-hoc networks. to appear in Proceedings of The Second International Conference on Pervasive Computing, 2004.

[27] Ulrich Wurst. Einsatz kryptographischer Verfahren auf stark ressourcenbeschränkten Geräten, 2003.

[28] X10 Home Automation System. Webpage. http://www.x10.com.

[29] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Mobile Computing and Networking*, pages 275–283, 2000.

[30] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[31] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. Technical Report ISE-TR-03-01, George Mason University, March 2003.

[32] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

Erklärung

Hiermit versichere ich, diese Arbeit
selbständig verfaßt und nur die
angegebenen Quellen benutzt zu haben.

(Holger Cermann)