Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Masterthesis

im Studiengang Informatik
mit der Studienrichtung Informationssicherheit

Entwicklung eines Schlüsselmanagement-Konzeptes für ein exemplarisches Smart Home System

von

Thorsten Rick

Matrikelnr.: 9018195

Erstprüfer: Prof. Dr. Karl Jonas

Zweitprüfer: Prof. Markus Ullmann

Eingereicht am: 13.03.2017

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die Masterarbeit selbstständig verfasst und keine anderen als die angegebene Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommen Gedanken als solche kenntlich gemacht habe.

Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde auch nicht veröffentlicht. Ich erkläre mich damit einverstanden, dass die Arbeit mithilfe eins Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft wird.

Unterschrift Sankt Augustin, den

Inhaltsverzeichnis

Eidesstattliche Erklärung ii
Abbildungsverzeichnis v
Abkürzungsverzeichnis vii
1 Einleitung 1
1.1 Motivation 1
1.2 Zielsetzung 1
1.3 Vorgehensweise 2
2 Sicherheit von Heimautomatisierungssystemen 3
2.1 Heimautomatisierung-Model (Target Analysis) 4
2.2 Angriffsvektoren eines Heimautomatisierungsnetzwerks (Attack Analysis) 6
3 Sicherheitskonzept eines Heimautomatisierungsnetzwerk 9
3.1 Funktionale Anforderungen 10
3.2 Nicht-funktionale Anforderungen 11
4 Schlüsselmanagement 13
4.1 Generierung 13
4.2 Trennung 14
4.3 Speicherung 14
4.4 Austausch 14
4.5 Vernichtung 14
5 Bewertungsschema 15
6 State of the Art 17
6.1 Diffie-Hellman Schlüsselaustausch 17
6.2 Needham-Schroeder-Protokoll 18
6.3 Out-of-Band Schlüsselaustausch 19
6.4 Public-Key-Infrastruktur 20
6.4.1 RSA/SHA-1 22
6.4.2 ECC/SHA-2 24
6.5 Hybride Kryptosysteme 28
6.6 Konzepte aus der Forschung 31
6.6.1 Physical Layer Security 32
6.6.2 Key Management in Wireless Sensor Networks 33
6.7 Zusammenfassung der Bewertungen 37
7 Vergleich verschiedener Konzeptideen 38
8 Eigenes Konzept 40
8.1 Netzwerkstruktur 40
8.2 Schlüsselerzeugung 41
8.3 Schlüsseltrennung 41

8.4 Schlüsselspeicherung ... 41
8.5 Schlüsselaustausch ... 41
8.5.1 Netzwerkbeitritt ... 42
8.5.2 Gruppenschlüsselaustausch ... 42
8.5.3 Schlüsselaktualisierung ... 43
8.6 Schlüsselvernichtung ... 44
9 Proof-of-Concept ... 45
9.1 Kommunikations- und Verschlüsselungsprotokoll ... 45
9.1.1 Constrained Application Protocol (CoAP) ... 46
9.1.2 Datagram Transport Layer Security (DTLS) ... 46
9.2 Exemplarisches Smart Home System (Testumgebung) ... 49
9.3 Schlüsselerzeugung im Management Knoten (Android) ... 49
9.4 Schlüsselspeicherung und –abruf mit Android KeyStores ... 50
9.5 Implementierung der Konzeptprotokolle ... 51
9.5.1 Knotenregistrierung via QR-Code ... 51
9.5.2 Erzeugung von Kommunikationsgruppen ... 54
9.5.3 Knoten- oder Gruppenschlüsselaktualisierung ... 55
10 Evaluierung ... 57
11 Zusammenfassung und Ausblick ... 62
12 Literaturverzeichnis ... 63
13 Anhang ... 66
13.1 Applikations Methoden ... 66
13.2 Wireshark ... 71

Abbildungsverzeichnis

Abbildung 1: Hierarchie eines Heimautomatisierungsnetzwerkes 4
Abbildung 2: Geräte, Kontrollanwendungen und Nutzer-Anwendungen 5
Abbildung 3: Zweistufige Hierarchie eines Heimautomatisierungsnetzwerkes 6
Abbildung 4: Klassen von Netzwerkangriffen [2] 7
Abbildung 5: Ergebnisformel des Bewertungsschemas 15
Abbildung 6: Bewertungsschema Layout 16
Abbildung 7: Diffie-Hellman Protokoll 18
Abbildung 8: Needham-Schroeder-Protokoll 19
Abbildung 9: Rollen einer Public-Key-Infrastruktur [15] 21
Abbildung 10: Inhalt eines X.509 Zertifikates 21
Abbildung 11: RSA Verschlüsselungsalgorithmus 22
Abbildung 12: RSA Entschlüsselungsalgorithmus 22
Abbildung 13: RSA Signaturalgorithmus 23
Abbildung 14: RSA/SHA-1 Bewertungsergebnis 24
Abbildung 15: Elliptische Kurven Addition [17] 25
Abbildung 16: Elgamal-Verschlüsselungsalgorithmus 26
Abbildung 17: Elgamal-Entschlüsselungsalgorithmus 26
Abbildung 18: Elgamal Entschlüsselungsgleichung 26
Abbildung 19: ECC/SHA-2 Bewertungsergebnis 28
Abbildung 20: TLS Bewertungsergebnis 31
Abbildung 21: Robust Key Generation from Signal Envelopes Bewertungsergebnis ... 33
Abbildung 22: Key Management in Wireless Sensor Networks Gleichung 34
Abbildung 23: Direkter Beitritt zum Netzwerk 34
Abbildung 24: Indirekter Beitritt zum Netzwerk 35
Abbildung 25: Schlüsselwideruf 35
Abbildung 26: Aktualisierung eines Sitzungsschlüssels 36
Abbildung 27: Key Management in Wireless Sensor Networks Bewertungsergebnis .. 37
Abbildung 28: Bewertungsergebnisse "State-of-the-Art" 37
Abbildung 29: Netzwerkstruktur des Konzeptes 40
Abbildung 30: Beispiel QR-Code 42
Abbildung 31: Gruppenschlüsselaustausch 43
Abbildung 32: Schlüsselaktualisierung 44
Abbildung 33: Protokoll Stack CoAP-DTLS 46
Abbildung 34: DTLS Record Layer 47
Abbildung 35: DTLS Cookie Zusammensetzung 47
Abbildung 36: DTLS DoS Gegenmaßnahmen 48
Abbildung 37: DTLS Handshake Protokoll 49

Abbildung 38: Generierung eines AES Schlüssels (Android) 50
Abbildung 39: Schlüsselspeicherung über einen KeyStore 50
Abbildung 40: Schlüsselzugriff über einen KeyStore 51
Abbildung 41: Beispiel QR-Code eines Knoten 52
Abbildung 42: Screenshot Knoten Activity 52
Abbildung 43: Light Activity Switch (Example DTLS Connection) 53
Abbildung 44: Screenshot Gruppenerzeugung 54
Abbildung 45: Generierung eines zufälligen AES Schlüssels 55
Abbildung 46: Screenshot Schlüsselaktualisierung 56
Abbildung 47: Bewertung des entwickelten Konzeptes 58
Abbildung 48: Unicast Kommunikation zwischen MK und AK 59
Abbildung 49: DTLS „Server Hello Verify“ Cookie 59
Abbildung 50: DTLS zweiter „Client Hello“ 60
Abbildung 51: DTLS "Client Key Exchange" 60
Abbildung 52: Application Data Ver -und Entschlüsselt 61
Abbildung 53: DNS Discovery Service 66
Abbildung 54: DNS Resolve Service 67
Abbildung 55: Generierung eines neuen Gruppenschlüssels 68
Abbildung 56: Verteilung der Gruppenparameter an die Gruppenmitglieder 68
Abbildung 57: Generierung eines Knotenschlüssels und Austausch mit dem Knoten . 69
Abbildung 58: Erzeugung und Speicherung eines QR-Codes via mithilfe von ZXing .. 70
Abbildung 59: DNS Discovery Antwort eines Knoten 71
Abbildung 60: Key Update entschlüsselte Applikationsdaten 71

Abkürzungsverzeichnis

AES ..... *Advanced Encryption Standard*, *Advanced Encryption Standard*
CA ..... *Certificate Authority*
CoAp ..... *Constrained Application Protocol*
DTLS ..... *Datagram Transport Layer Security*
ECC ..... *Elliptic Curve Cryptography*
ECDH ..... *Elliptic Curve Diffie-Hellman*
ECDSA ..... *Elliptic Curve Digital Signature Algorithm*
FG ..... *Feld Geräte*
HAN ..... *Heimautomatisierungsnetzwerk*
HLKK ..... Heizungs-, Lüftungs-, Klima- und Kältetechnik
ICG ..... *Interconnection Geräte*
IDS ..... Intrusion Detection System
IETF ..... *Internet Engineering Task Force*
ITU ..... *International Telecommunication Union*
IV ..... *Initialisierungsvektor*
M2M ..... *Mehrpunkt-zu-Mehrpunkt*
MAC ..... *Message Authentication Code*
MG ..... *Management Geräte*
MS ..... *Master-Secret*
NIST ..... *National Institute of Standards and Technology*
P2M ..... *Punkt-zu-Mehrpunkt*
P2P ..... *Punkt-zu-Punkt*
PKI ..... *Public-Key-Infrastruktur*
PLS ..... *Physical Layer Security*
PMS ..... *Pre-Master-Secret*
PSK ..... *Pre-Shared Key*
QoS ..... *Quality of Service*
RA ..... *Registration Authority*
REST ..... *Representational State Transfer*
RSA ..... Rivest, Shamir und Adleman Algorithmus
SFIR ..... *Secure Fuzzy Information Reconciliator*
SHA ..... *Secure Hash Algorithm*
SPN ..... *Substitutions-Permutations-Netzwerk*
TLS ..... *Transport Layer Security*
TPM ..... *Trusted Platform Module*
URI ..... *Uniform Resource Identifier*
VA ..... *Validation Authority*

1 Einleitung

Der Begriff „Smart Home“ beschreibt ein vernetztes Zuhause, das mithilfe von Sensoren und Aktoren versucht, die Lebens- und Wohnqualität, die Betriebs- und Einbruchsicherheit, sowie die Energieeffizienz zu erhöhen. Zur Umsetzung eines vernetzten Hauses entwickelten verschiedene Hersteller – wie z. B. ZigBee, Z-Wave oder EnOcean – Kommunikationsprotokolle, die einen Datenaustausch zwischen sonst getrennten Systemen ermöglichen. Da innerhalb eines „Smart Home“ Netzwerkes neben unkritischen Informationen – wie z. B. Status der Wohnzimmerbeleuchtung – auch sicherheitskritische Informationen und Befehle – wie z. B. Türpasswörter oder Fensterverriegelungen – kommuniziert werden, müssen diese Netzwerke abgesichert werden.

1.1 Motivation

Eine erste Untersuchung innerhalb eines Masterprojektes [1] zeigte, dass die untersuchten Protokolle von ZigBee, Z-Wave und EnOcean als Sicherheitsmechanismus den AES-128 nutzen. Hierbei wird für den initialen Schlüsselaustausch entweder auf einen vorinstallierten „Masterschlüssel“ gesetzt oder unverschlüsselt kommuniziert. Dabei werden die Masterschlüssel in der Firmware gespeichert, innerhalb des Systemspeichers abgelegt oder müssen vom Anwender manuell installiert werden. Diese Praktiken schaffen entweder potenzielle Sicherheitslücken oder verringern den Anwenderkomfort.

1.2 Zielsetzung

Das Ziel der Arbeit soll die Entwicklung, Realisierung und Evaluierung eines Schlüsselmanagement Konzeptes für ein prototypisches „Smart Home“ System sein. Dabei soll ein besonderer Wert auf eine sichere Erzeugung und Kommunikation der genutzten Schlüssel, sowie ein möglichst hoher Anwenderkomfort gesetzt werden.

Das zu entwickelnde Konzept soll im Hinblick auf die schnelle Entwicklung moderner Mikrokontroller entwickelt werden und setzt somit keine direkten Performanceeinschränkungen an die einzelnen Knoten voraus. Dennoch wird bei der Auswahl und Evaluation der Sicherheitsalgorithmen, die erforderliche Rechenleistung und der benötigte Speicherbedarf mit bewertet.

Zur Entwicklung des Konzeptes sollen etablierte Sicherheitsalgorithmen aus „Smart Home“ Systemen und „Smart Home“ fremden Systemen betrachtet und anhand eines Bewertungsschemas evaluiert werden. Hierzu wird ein Angreifermodell definiert und angeschaut ob und wie die geforderten Sicherheitsziele erreicht werden. Zusätzlich zu bereits etablierten Sicherheitskonzepten sollen auch neue und innovative Schlüsselmanagement und Schlüsselaustausch Konzepte aus der Forschung betrachtet und mit dem erzeugten Schema evaluiert werden. Eine vorläufige Liste mit möglichen Verfahren:

- Etablierte Verfahren.
    - SSL bzw. TLS (DTLS)
    - RSA
    - ECC
    - AES
    - Diffie-Helman
- Konzepte aus der Forschung
    - Secure Communication in Home and Building Automation Systems (Dissertation) [2]
    - An Identitiy-Based Key-Exchange Protocol [3]
    - An efficient identity-based key exchange protocol with KGS forward secrecy for low-power devices [4]

1.3 Vorgehensweise

Zu Beginn der Arbeit werden der allgemeine Sicherheitsbegriff in der Heimautomatisierung, sowie mögliche Angriffsvektoren erläutert. Dabei wird auf die grundlegende Netzwerkstruktur innerhalb eines Heimautomatisierungsnetzwerkes näher eingegangen und die Anwendungsstrukturen beschrieben. Anschließend werden die Anforderungen und Herausforderungen, die ein Sicherheitskonzept in der Heimautomatisierung abdecken soll, identifiziert. Dabei werden die funktionalen und nicht-funktionalen Anforderungen, die später für die Evaluation genutzt werden, erläutert. Nachdem die notwendigen Eigenschaften eines Heimautomatisierungsnetzwerkes und eines Sicherheitskonzept beschrieben wurden, wird anschließend auf das Schlüsselmanagement näher eingegangen. Hierbei werden die für die Arbeit wichtigen Disziplinen betrachtet und die Relevanz für ein Sicherheitskonzept in der Heimautomatisierung erläutert. Basierend auf den gewonnenen Erkenntnissen, wird ein Bewertungsschema konstruiert mit dem im Folgenden verschiedene Schlüsselmanagementkonzepte evaluiert werden. Mithilfe der Bewertungsergebnisse und Erkenntnisse aus der Evaluierung wird jetzt ein eigenes Konzept entwickelt, das anschließend prototypisch in einer Testumgebung umgesetzt wird. Zum Abschluss der Arbeit wird das eigene Konzept mithilfe des Bewertungsschemas evaluiert und die Ergebnisse der Arbeit zusammengefasst.

2 Sicherheit von Heimautomatisierungssystemen

Der Begriff der Sicherheit besitzt je nach Autor oder Betrachtungsposition eine unterschiedliche Definition. Innerhalb der Informatik wird häufig, sollte die allgemeine Systemsicherheit gemeint sein, von IT-Sicherheit gesprochen. Dabei unterteilt sich die IT-Sicherheit in die Funktionssicherheit und die Informationssicherheit.

Die Funktionssicherheit (englisch: Safety) gewährleistet, dass sich ein gegebenes System gemäß der definierten Funktionalität verhält. Damit betrachtet die Funktionssicherheit einen unbeeinflussten Betrieb eines Systems und identifiziert Abweichungen vom definierten Systemkonzept. Die Informationssicherheit (englisch: security) hingegen betrachtet den Schutz der technischen Verarbeitung von Informationen. Hierbei sollen nicht-autorisierte Datenmanipulationen verhindert und die Preisgabe von Informationen unterbunden werden. Da im Rahmen dieser Arbeit die Kommunikation zwischen einzelnen Knoten insbesondere das Schlüsselmanagement relevant ist, wird die Funktionssicherheit nicht näher betrachtet.

Ein modernes Smart Home System besteht aus einer Vielzahl kleinerer Teil-Systeme – wie z. B. einem „Intrusion Detection System“ (IDS), HLKK-Anlagen oder Multimedia-Systemen –. Diese Teil-Systeme lassen sich aus Anwenderperspektive in drei Sicherheitsstufen einteilen. Zum einem sicherheits-irrelevante Systeme, deren Ausfall lediglich einen Komfortverlust für den Anwender bedeutet. Solche Systeme sind im Allgemeinen die Unterhaltungselektronik eines Smart Home. Zum zweiten sicherheits-relevante Systeme, die bei einem Bruch der Sicherungsmaßnahmen neben einem Komfortverlust ihre definierte Funktionalitäten lediglich eingeschränkt zur Verfügung stellen können. Darunter fallen die bereits erwähnten HLKK-Anlagen, die z. B. bei einem Angriff ihre Klimaanlagensteuerung verlieren, jedoch weiterhin die Heizungsanlage steuern können. Zuletzt sicherheits-kritische Systeme, die bei einem Bruch der Sicherungsmaßnahmen neben einem Komfortverlust und Funktionsverlust auch das gewünschte Sicherheitsniveau nicht weiter gewährleisten können. Unter solche Systeme fallen z. B. Zugangskontrollsysteme, die bei einer Kompromittierung der Sicherheitsmaßnahmen keine sicheren Aussagen zu möglichen Eindringlingen treffen können.

Generell führen Angriffe auf Smart Home Systeme zu kleinen bis keinen wirtschaftlichen Schäden für den Besitzer. Da jedoch bei den drei Stufen immer mit einem Komfortverlust für den Anwender zu rechnen ist, werden heutzutage auch bei sicherheits-irrelevanten Systemen Sicherungsmaßnahmen umgesetzt.

Im Rahmen der Informationssicherheit wird ein Angriff (englisch: security attack) von einem Angreifer (englisch: Adversary) über Schwachstellen (englisch: Vulnerabilities) durchgeführt. Dabei kann der Angreifer eine Person oder eine böswillige Software z. B. ein Trojaner oder Wurm sein. Ein Angreifer erhofft sich in der Regel über einen solchen Angriff, Zugriff und Kontrolle über das jeweilige System. Jede Schwachstelle in einem System führt zu einer sogenannten Bedrohung (englisch: Threat), die im Allgemeinen beschreibt wie schwerwiegend und wahrscheinlich ein Angriff über diese Schwachstelle ist. Somit gilt es den eigentlichen Angriff und das Angriffspotenzial einer Bedrohung zu unterscheiden. Einem Systementwickler obliegt nun die Aufgabe die auftretenden Schwachstellen und die dazugehörigen Bedrohungen zu identifizieren und geeignete Gegenmaßnahmen (englisch: countermeasures) bereitzustellen. Dabei unterscheidet man grundlegend zwischen zwei verschiedenen Strategien.

Zum einen lassen sich mögliche Schwachstellen bei der Systementwicklung identifizieren und beseitigen. Diese Strategie auch Angriffs-Vermeidung (englisch: Attack Prevention) genannt, bedeutet in der Regel, dass bereits bei der Systementwicklung sicherheits-kritische Komponenten angemessen abgesichert werden. Die zweite Strategie wird für die Schwachstellen genutzt, die sich nicht mit verhältnismäßigem Aufwand absichern lassen und somit nicht unter Strategie eins fallen. Hierbei wird lediglich eine Angriffs-Detektion (englisch: Attack Detection) angestrebt, bei dem ein sicherheits-kritischer oder abnormaler System-Status identifiziert und gemeldet werden soll.

Im Rahmen der Sicherheitsanalyse eines Systems sollte die erste Strategie immer präferiert werden, da sie einen direkten Schutz vor möglichen Schwachstellen bietet und Bedrohungen eliminiert. Jedoch sind in modernen System auch immer Schwachstellen zu finden, die sich nicht mit akzeptablen Aufwand absichern lassen. Somit wird heutzutage meist eine hybride Strategie, die sich aus Vermeidung und Detektion zusammensetzt, genutzt.

2.1 Heimautomatisierung-Model (Target Analysis)

Nachdem im vorherigen Abschnitt von Angreifern, Angriffen und Teilsystemen gesprochen wurde, wird im nachfolgenden Abschnitt auf den grundlegenden Aufbau eines Heimautomatisierungsnetzwerkes (HAN) näher eingegangen. Hierbei wird eine abstrakte herstellerunabhängige Struktur betrachtet, die zur Identifizierung möglicher Angriffsziele genutzt wird. Wobei das Endziel eines Angriffes auf ein Automationsnetzwerk im Allgemeinen immer ein Zugriff auf die Kontrollfunktionen (englisch: control functionality) des Feld-Level sein wird. Im Allgemeinen basiert ein HAN und die darin bereitgestellten Kontrollfunktionen auf einer dreistufigen Hierarchie (siehe Abbildung 1). Die unterste Ebene der Hierarchie auch Feld-Level genannt, bilden die sogenannten Feldgeräte. Diese Feldgeräte agieren direkt mit der Umwelt und können dabei Werte erfassen – wie z.B. die Temperatur, Luftfeuchtigkeit oder Sonneneinstrahlung –, wobei sie die erfassten Informationen auch in ein geeignetes Format für den Transport innerhalb des Heimautomatisierungssystem umwandeln. Des Weiteren werden Eigenschaften des Systems – wie z. B. Einstellungen oder Positionierungen von Feldgeräten – physikalisch über Kommandos aus dem Automationsnetz kontrolliert. Direkt über dem Feld-Level befindet sich der sogenannte Automations-Level. Innerhalb dieser Ebene der Hierarchie werden die grundlegenden Automationsfunktionalitäten umgesetzt. Hierfür werden Kontrollschleifen und –sequenzen implementiert, die auf den Kontrollfunktionen des Feld-Levels aufsetzten. Somit lassen sich z. B. Beleuchtungsketten auf Basis der Außenhelligkeit steuern. Die oberste Ebene der Hierarchie bildet das sogenannte Management-Level, das für globale Konfigurationen des Netzwerkes und die eigentliche Visualisierung für den Anwender zuständig ist.

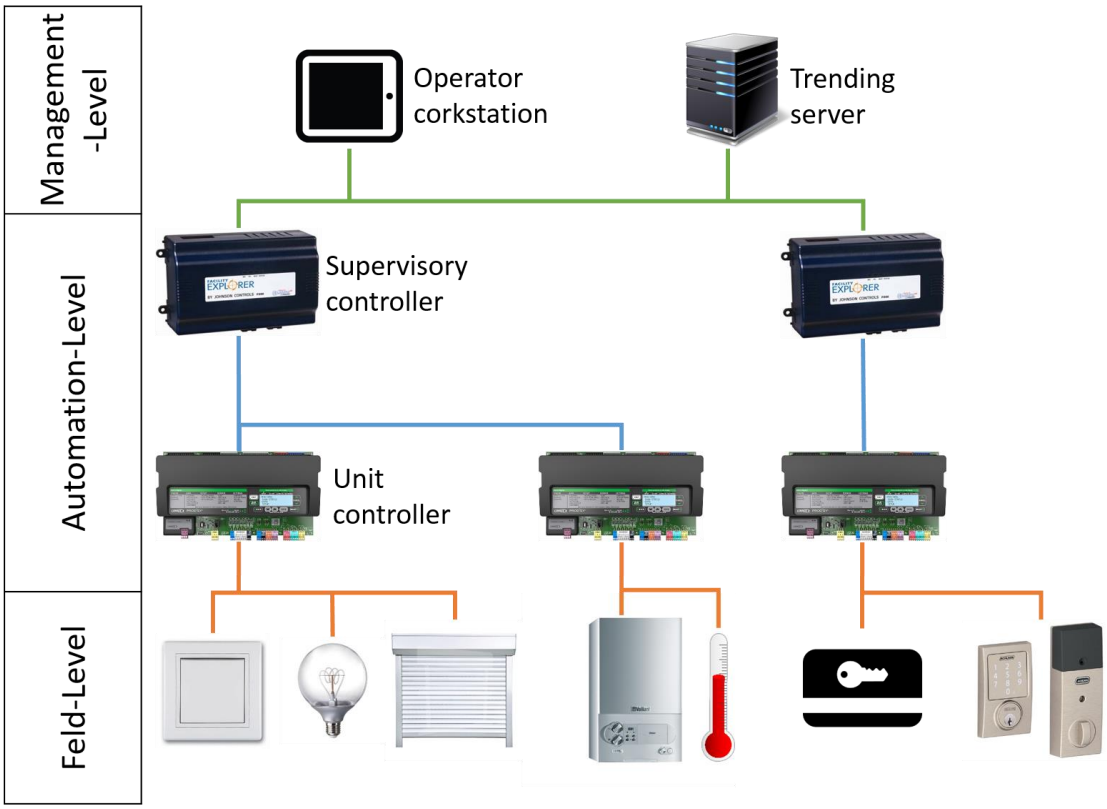

Abbildung 1: Hierarchie eines Heimautomatisierungsnetzwerkes

Diese Netze werden im Allgemeinen als sogenannte verteilte Systeme (englisch: Distributed Systems) implementiert. Hierbei wird eine Kontrollfunktionalität über sogenannte verteilte Kontrollanwendungen realisiert (englisch: Distributed Control Applications). Diese Anwendungen können über eine Vielzahl von Geräten innerhalb des Netzwerkes verteilt sein. Die eigentlichen Funktionalitäten der Kontrollanwendungen, werden über sogenannte Nutzer-Anwendungen (englisch: User Applications) bereitgestellt. Im Gegensatz zu den Kontrollanwendungen, sind Nutzer-Anwendungen immer einem einzelnem Gerät zugeordnet.

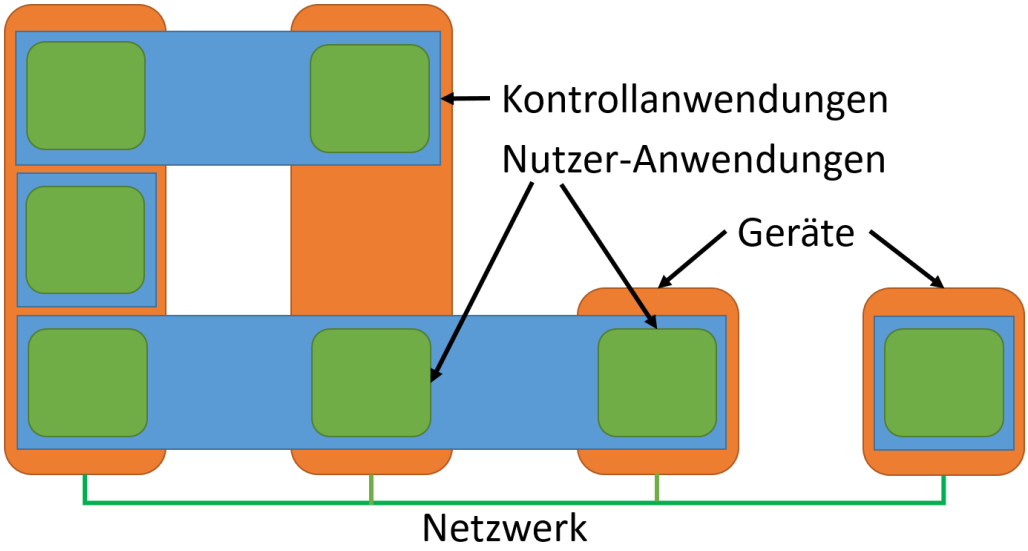

Abbildung 2: Geräte, Kontrollanwendungen und Nutzer-Anwendungen

Abbildung 2 zeigt beispielhaft den Aufbau eines verteilten Systems innerhalb eines Automatisierungsnetzwerkes. Über vier Geräte werden ebenfalls vier Kontrollanwendungen etabliert. Dafür greifen die jeweiligen Kontrollanwendungen über Nutzer-Anwendungen, der einzelnen Geräte, auf Gerätefunktionen zu.

Eine wie in Abbildung 2 gezeigte, verteilte Anwendungsstruktur, benötigt für den Austausch von Daten zwischen den einzelnen Anwendungen ein bestehendes Netzwerk zwischen den Geräten. Hierbei gilt es grundlegend zwischen zwei Kommunikationsarten zu unterscheiden. Zum einen der Austausch von sogenannten Kontrolldaten (englisch: Control Data), die zwischen den einzelnen Nutzer-Anwendungen einer Kontrollanwendung ausgetauscht werden müssen. Typische Kontrolldaten sind aktuelle Sensor- und Aktorwerte – wie z. B. Raumtemperaturen – oder virtuelle Prozessdaten – wie z. B. Regelabweichungen –. Zum Anderen der Austausch von sogenannten Konstruktionsdaten (englisch: Engineering Data), die im Allgemeinen zur Konfiguration der Kontrollanwendungen genutzt werden.

Eine strickte Aufteilung auf Basis der Funktionen und Geräte, wie sie innerhalb der Hierarchie in Abbildung 1 dargestellt wird, findet in modernen Automatisierungsnetz weniger statt. Die steigende Leistungsfähigkeit moderner Mikrokontroller ermöglicht es heutzutage Geräte des Feld-Levels – wie z. B. einem Sensor oder einem Lichtschalter – mit leistungsstarken Netzwerkkontrollern zu wirtschaftlichen Preisen auszustatten. Somit ist es Feld-Geräten möglich neben ihren Hauptfunktionen auch Funktionen aus dem Automations-Level zu übernehmen. Infolge dessen verschwimmen in modernen Automatisierungsnetzen die verschiedenen Level der Hierarchie immer stärker. So wird heutzutage eher eine zweischichtige Hierarchie praktiziert. Innerhalb dieser Übernehmen die Geräte des Management- und Feld-Levels, die Funktionen des Automations-Levels.

Somit entsteht die in Abbildung 3 dargestellte zweistufige Hierarchie, die aus Feld-Netzwerken und einem Kontrollnetzwerk besteht. Ein Feld-Netzwerk setzt sich aus einer be-

liebigen Anzahl von Feld-Geräten, die über ein Transportmedium miteinander kommunizieren können, zusammen. Diese Feld-Netzwerke sind wiederrum über ein sogenanntes Backbone-Netzwerk, das „Interconnection“ Geräte als Gateways nutzt, miteinander verbunden. Innerhalb des Backbone-Netzwerks befinden sich auch die Management-Geräte, die einen Zugriff auf das gesamte Automatisierungsnetzwerk benötigen. Auf Basis der unten gezeigten Hierarchie können drei verschiedenen Geräteklassen identifiziert werden:

- Feld-Geräte (Sensoren, Aktoren und Kontroller) agieren direkt mit der Umgebung, erfassen Umgebungsdaten und führen Kontrollkommandos aus.
- Interconnection Geräte (Router und Gateways) dienen als Schnittstelle zwischen den verschiedenen Netzwerkebenen. Einerseits intern zwischen dem Feld-Level und dem Management-Level, andererseits zwischen dem Automatisierungsnetzwerk und externen Netzwerken wie z. B. dem Internet.
- Management-Geräte ermöglichen Konfigurationen, Wartungsarbeiten und andere Nutzeraktionen.

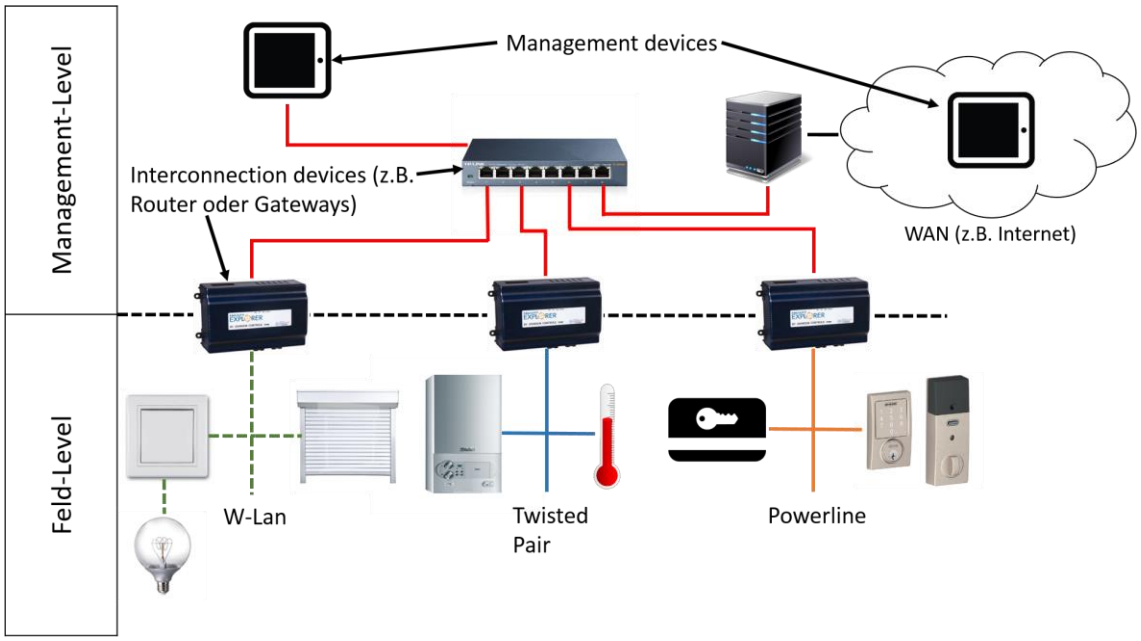

Abbildung 3: Zweistufige Hierarchie eines Heimautomatisierungsnetzwerkes

2.2 Angriffsvektoren eines Heimautomatisierungsnetzwerks (Attack Analysis)

Basierend auf der erläuterten Hierarchie und den definierten Geräteklassen, wird zum Abschluss des Kapitels näher auf mögliche Angriffsvektoren innerhalb eines HAN eingegangen. Hierzu werden zu Beginn einige Angriffsziele identifiziert und kurz erläutert:

- Feld-Netzwerk: Angreifer mit direktem Zugriff auf ein Feld-Netzwerk, besitzen unter Umständen direkten Zugriff auf unverschlüsselte Kommunikation innerhalb des Netzwerkes. Somit könnten die übertragenden Kontroll- und Konstruktionsdaten gelesen, verändert oder blockiert werden.
- Management-Netzwerk: Vergleichbar mit dem Feld-Netzwerk, da jedoch im Backbone-Netzwerk die Kommunikation aller Feld-Netzwerke zusammenläuft, erhält ein Angreifer somit eine Sicht auf das gesamte Automatisierungsnetzwerk.
- Feld-Geräte (FG): Ein direkter Zugriff auf ein Feld-Gerät, könnte es einem Angreifer ermöglichen, über die verteilten Kontrollanwendungen direkten Einfluss auf die Funktionalitäten zu nehmen. Somit könnten Konfigurationen oder Kontrolldaten direkt auf dem Feld-Geräten manipuliert werden.

- Interconnection Geräte (ICG): Dieser Gerätetyp dient im Allgemeinen der Verbindung zweier Netzwerke. Dadurch erhält ein Angreifer, sollte er einen direkten Zugriff auf ein solches Interconnection Gerät erlangen, Einsicht in die gesamte Kommuniaktion zum und vom Interconnection Gerät. Bei einer verschlüsselten Kommunikation hingegen, erhält der Angreifer lediglich Einsicht in den unverschlüsselten Nachrichtenteil. So entstehen die gleichen Angriffsszenarien wie schon im Feld-Netzwerk.
- Management-Geräte (MG): Die zentrale Rolle der Management-Geräte, die zur Konfiguration und Verwaltung des Automatiosierungsnetzwerkes genutzt werden, macht sie besonders interessant für Angriffe. Sollte ein Angreifer Zugriff auf ein Management-Geräte erhalten oder dieses Gerät sogar durch ein korrumpiertes Gerät ersetzen, könnte er die erhaltenden Zugriffsrechte missbrauchen. Damit wäre es einem Angreifer ohne weitere Hindernisse möglich, die vorher genannten Angriffe durchzuführen.

Generell kann in Automatisierungsnetzwerken zwischen Netzwerk-Angriffen und Geräte-Angriffen unterschieden werden. In den meisten Fällen – wie z. B. einem Funknetzwerk – lassen sich Netzwerkangriffe einfacher vorbereiten und durchführen. Da ein Angreifer keinen direkten physischen Zugriff auf das Netzwerk benötigt, sondern mithilfe geeigneter Sende und Empfangshardware aus der Ferne arbeiten kann. Nach [5] lassen sich Netzwerkangriffe in die Klassen Abfangen (englisch: Interception), Unterbrechung (englisch: Interruption), Modifikation (englisch: Modification) und Fälschung (englisch: Fabrication) unterteilen (siehe Abbildung 4). Eine „Interception Attack“ bezeichnet im Allgemeinen den Versuch eines Angreifers Kommunikation zu belauschen oder abzufangen und so einen unautorisierten Zugriff auf Daten zu erhalten. Die zweite Angriffsklasse (Interruption) beinhaltet Angriffe, die einen vollständigen Kommunikationsabbruch hervorrufen sollen. Somit finden sich hier die sogenannten Denial-of-Service Angriffe. Unter Modifikation befinden sich Angriffe, bei denen der Angreifer versucht, Daten während des Transports über das Netzwerk zu manipulieren. Innerhalb der letzten Angriffsklasse (Fabrication) befinden sich Angriffe, bei denen ein Angreifer versucht gefälschte oder bösartige Daten in das Netzwerk einzuspeisen.

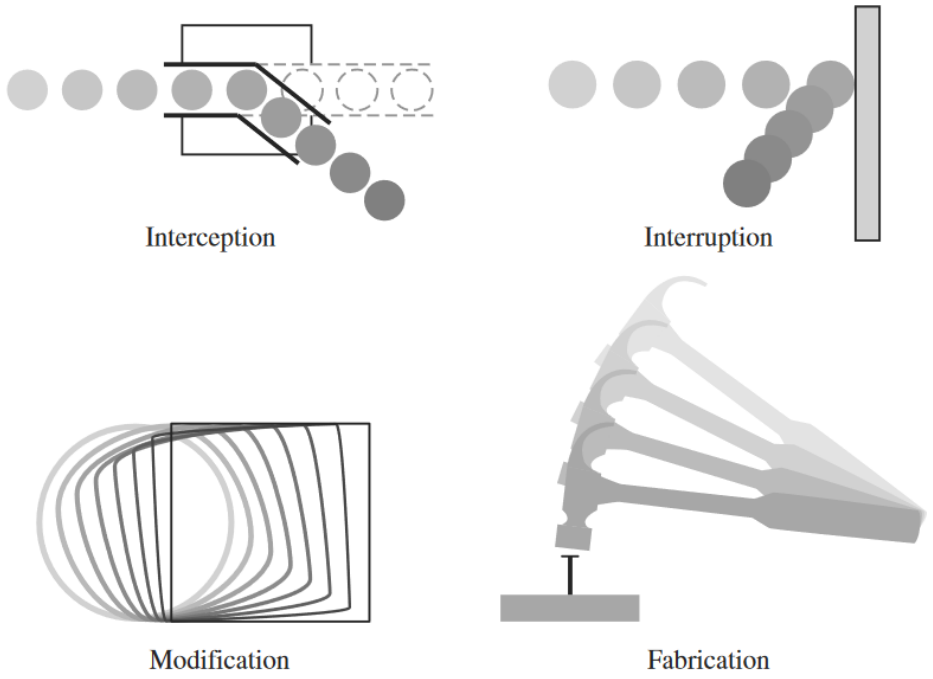

Abbildung 4: Klassen von Netzwerkangriffen [2]

Bei direkten Angriffen auf das Netzwerk, spielen innerhalb des Automatisierungsszenarios besonders der Man-In-The-Middle-Angriff und der sogenannte Denial-of-Service-Angriff eine wichtige Rolle. Dabei umfasst der Man-In-The-Middle-Angriff neben Netzwerk-Sniffing auch Replay-Angriffe und Datenmanipulationen. Bei den sogenannten Replay-Angriffen, werden von einem Angreifer Datenpakete abgefangen und zu einem späteren Zeitpunkt wieder in das Netzwerk eingeschleust. Womit sich ein Angreifer z. B. unberechtigten Zugriff auf Netzwerkfunktionalitäten erhofft. Wohingegen der Denial-of-Service-Angriff, das gezielte Unterbinden legitimer Kommunikation bezeichnet.

3 Sicherheitskonzept eines Heimautomatisierungsnetzwerk

Nachdem im vorherigen Kapitel der allgemeine Sicherheitsbegriff in der Heimautomatisierung sowie mögliche Angriffsvektoren erläutert wurden, werden in den nachfolgenden Abschnitten die besonderen Anforderungen und Herrausforderungen eines Sicherheitskonzeptes für ein HAN betrachtet. Auf Grundlage dieser Erkenntnisse wird im Kapitel 4 ein Bewertungsschema entwickelt, dass zur Evaluierung der verschiedenen „State-of-the-Art“ Protokolle und Konzepte in Kapitel 6 genutzt wird. Bevor im folgenden Aussagen über mögliche Anforderungen und Herrausforderungen getroffen werden können, wird vorher auf die verschiedenen Kommunikationstypen innerhalb eines HANs eingegangen.

Innerhalb eines Automationsnetzwerkes können eine Vielzahl von Kommunikationsgeräten (englisch: Communication Entity) über Nutzer-Anwendungen Kontroll- und Konstruktionsdaten miteinander austauschen (siehe Abbildung 2). Hierzu formen diese Kommunikationsgeräte eine sogenannte Kommunikationsbeziehung (englisch: Communication Relationship). Eine solche Beziehung wird mithilfe eines Kommunikationsmodels (englisch: Communication Model) charakterisiert. Dabei kann anhand der betroffenen Kommunikationsgeräte zwischen einer Punkt-zu-Punkt (P2P), Punkt-zu-Mehrpunkt (P2M) oder Mehrpunkt-zu-Mehrpunkt (M2M) Verbindung unterschieden werden. Das wohl verbreiteste Model, das auf einer P2P Verbindung beruht, ist das sogenannte Client/Server Modell. Hierbei stellt ein Dienstnehmer (Client) eine Anfrage (englisch: Request) an einen Dienstgeber (Server). Dieser verarbeitet die Anfrage und antwortet dem Client mit einer entsprechenden Antwort (englisch: Response) oder bestätigt lediglich das ausgeführte Kommando (englisch: Acknowledgment).

Die Wohl bekanntesten Modelle, die auf einer P2M oder M2M Verbindung beruhen, sind das Producer/Consumer und das Publisher/Subscriber Modell. Bei einem Producer/Consumer Modell, stellen eine beliebige Anzahl von sogenannten Produzenten Daten öffentlich zu Verfügung. Die sogenannten Konsumenten können diese Daten empfangen und entscheiden ob sie ein Interesse an den Daten haben oder sie verwerfen wollen. Innerhalb eines solchen Modelles können sogenannte Kommunikationsgruppen entstehen (englisch: Communication Group). Eine derartige Gruppe setzt sich aus Sendern und Empfänger zusammen, die ein Interesse an den gleichen Daten besitzen. Da bei diesem Modell einer Kommunikationsgruppe jedes Mitglied selber über Beitritt und Austritt entscheidet, da keine zentrale Managementinstanz existiert, wird auch von einer losen Gruppenmitgliedschaft (englisch: Loose Group Membership) gesprochen.

Das Publisher/Subscriber Modell hingegen, das in den grundlegenden Kommunikationsstrukturen dem Producer/Consumer Modell gleicht, nutzt eine striktere Beitrittspolitik. Innerhalb des Publisher/Subscriber Modells muss jedes Gruppenmitglied einer Gruppe explizit beitreten bzw. austreten (englisch: Strict Group Membership). Hierbei sendet ein Gerät, das einen Beitritt wünscht, eine sogenannte „Subscription“ an einen Gruppenkoordinator (englisch: Group Coordinator). Dieser Koordinator entscheidet drüber, ob das anfragende Kommunikationsgerät der Gruppe beitreten darf. Erst nach einem erfolgreichem Beitritt ist es einem Gerät möglich innerhalb der Kommunikationsgruppe zu kommunizieren. Durch diese strikte Beitrittspolitik eignet sich das Publisher/Subscriber Modell im Gegensatz zum Producer/Consumer Modell auch für sicherheits-kritische Systeme. Da ohne eine Bestätigung des Koordinator keine neuen Mitglieder beitreten können.

Die vorgestellten Kommunikationsmodelle erläutern lediglich abstrakte Regeln einer Kommunikationsbeziehung. Wie diese Regeln schlussendlich umgesetzt werden, definieren die Modelle nicht. Somit benötigen die oben genannten Modelle ein Kommunikationsdienst, der einen Datentransport zwischen einzelnen Geräten ermöglicht. Hierbei wird im Allgemeinen zwischen Unicast, Broadcast und Multicast unterschieden. Als Unicast wird eine Kommunikation zwischen einem einzelnen Sender und einem einzelnen Empfänger bezeichnet. Broad- und Multicast hingegen bezeichnen eine Kommunikation zwischen einem einzelnen Sender und einer Vielzahl von Empfängern. Dabei wird bei

einer Broadcast Kommunikation die Nachricht ohne Selektion an alle Netzwerkteilnehmer gesendet, wohingegen bei einer Multicast Kommunikation die Nachricht lediglich an eine ausgewählte Gruppe gesendet wird.

Theoretisch lassen sich P2M oder M2M Verbindungen über eine Vielzahl von Unicast Verbindungen konstruieren. Dies würde im Falle des Publisher/Subscriber Modelles dazu führen, das ein Publisher zu jedem einzelnen Gruppenmitglied (Subscriber) eine dedizierte Unicast Verbindung aufbauen muss. Eine solche Umsetzung ist insbesondere in großen Gruppen ineffizient. Neben der Verwendung vieler Unicast Verbindungen, könnte auch eine einzelne Broadcast Nachricht genutzt werden. Diese müsste im Falle des Publisher/Subscriber Modelles noch zusätzlich z. B. über Verschlüsselungsalgorithmen so codiert werden, das lediglich Gruppenmitglieder sie lesen können. Jedoch gilt auch eine Broadcast Kommunikation in Netzwerken, die über viele kleinere Gruppen verfügen können als ineffizient. Da potenziell jeder Netzwerkteilnehmer das Paket verarbeiten muss. Deswegen streben moderne Heimautomatisierungssysteme eine Multicast Kommunikation, die eine direkte Adressierung einzelner Gruppenmitglieder ermöglicht an.

3.1 Funktionale Anforderungen

Auf Basis der erläuterten Heimautomatisierungsstruktur und den identifizierten Kommunikationstypen, werden im nachfolgenden Kapitel die für diese Arbeit wichtigen funktionalen Anforderungen für ein sicheres Schlüsselmanagement und eine darauf basierend sichere Kommunikation definiert.

Damit innerhalb eines HANs Gerät über sogenannte sichere Kanäle (englisch: Secure Channels) kommunizieren können, müssen einige grundlegende Sicherheitseigenschaften erfüllt sein. Diese Eigenschaften werden in der Informationssicherheit Sicherheitsziele genannt. Seit der Definition der ersten Sicherheitsziele vor rund 20 Jahren wurden immer weitere Sicherheitsziele anerkannt sowie bestehende weiter aufgespalten. Basierend auf [6] [7] [8] [9] [10] [11] [12] wurden die folgenden Sicherheitsziele für den Rahmen der Arbeit ausgewählt:

- ***Vertraulichkeit (Confidentiality):*** Vertraulichkeit gilt als gewährleistet, sollten lediglich berechtigte Instanzen einen Zugriff auf die Daten innerhalb des Systems besitzen. Somit müssen Maßnahmen getroffen werden, die es unberechtigten Instanzen untersagt die gespeicherten und versendeten Daten zu lesen. Die gängigen Maßnahmen zur Erfüllung der Vertraulichkeit sind Zugriffsberechtigungen für die gespeicherten Daten und Verschlüsselungsalgorithmen für den Datentransport.
- ***Integrität (Integrity):*** Integrität bezeichnen zum einen die Datenintegrität und zum anderen die Systemintegrität. Dabei gewährleistet die Integrität das Daten weder bei der Verarbeitung vom System noch bei der Übertragung beschädigt, verändert oder von unbefugten Dritten manipuliert wurden. Zur Gewährleistung dieser Eigenschaften müssen Techniken – wie z. B. kryptographisch sichere Hashfunktionen – genutzt werden um diese veränderten Daten zu erkennen.
- **Verfügbarkeit (Availability):** Zusammen mit der Vertraulichkeit und der Integrität, bildet die Verfügbarkeit die sogenannte CIA Triade. Die CIA Triade stellt eines der grundlegenden Sicherheitsmodelle in der Informationssicherheit dar. Dabei gilt die Verfügbarkeit als gewährleistet, sollten die verarbeiteten Daten und angebotenen Dienste innerhalb des HANs jederzeit und in der erwarteten Qualität zur Verfügung stehen.
- **Authentizität (Authenticity):** Die Authentizität teilt sich in die Nachrichtenauthentizität und Teilnehmerauthentizität auf. Dabei bezeichnet die Nachrichtenauthentizität, das Sichergestellt werden kann, dass empfangene Nachrichten aus einer angegebenen Quelle stammen. Zum anderen muss auch die Teilnehmerauthentizität gewährleistet sein, wobei gewährleistet werden muss, dass die vorgegebene Identität eines Kommunikationspartners korrekt ist.

- **Aktualität (Freshness):** Neben einem sicheren und unveränderten Transport der Daten, stellt insbesondere die Aktualität im Automationsbereich ein wichtiges Sicherheitsziel dar. Dabei muss gewährleistet werden, dass einmal versendete Daten nicht zu einem späteren Zeitpunkt wieder unbemerkt eingespielt werden können (Replay-Schutz).

Die vorgestellten Sicherheitsziele stellen lediglich die primären Sicherheitseigenschaften dieser Arbeit dar. Dabei gilt es zu beachten das die Auswahl geeigneter Sicherheitsziele stark vom jeweiligen Anwendungsszenario abhängen. So könnte in einem HAN, das Nutzeridentitäten mit einzelnen Geräten koppelt, die Anonymität ein weiteres wichtiges Sicherheitsziel darstellen. Anonymität gilt als gewährleistet, sollte ein Angreifer auf Basis der Kommuniaktion nicht auf die Identitäten der einzelnen Kommunikationspartner schließen können. Des Weiteren ist es denkbar, die vorgestellten Ziele auf Basis eines ausgewählten Dienstes unterschiedlich zu priorisieren. So könnte bei einem sicherheitsrelevanten Dienst eventuell die Vertraulichkeit weniger wichtig gewertet werden, als ein integerer und authentischer Datentransport und somit auf eine Kommunikationsverschlüsselung verzichtet werden.

3.2 Nicht-funktionale Anforderungen

Die Anforderungen an ein Sicherheitssystem werden neben den funktionalen Anforderungen, die beschreiben was ein System leisten soll, insbesondere von den Eigenschaften der jeweiligen Einsatzumgebung definiert. Zur Spezifizierung dieser Eigenschaften werden sogenannte nicht-funktionale Anforderungen, die beschreiben wie gut ein System etwas leisten soll, genutzt. Beispielsweise sind die bekannten Verfahren aus der IT-Security, wie z. B. IPsec oder Kerberus, auf den Einsatz in modernen Computernetzwerken zugeschnitten. Daraus folgt das im Allgemeinen standarisierte Verfahren aus der IT-Security, nicht ohne Anpassungen, innerhalb der Heimautomatisierungsdomäne genutzt oder lediglich mit Einschränkungen verwendet werden können. Auf Basis der Eigenschaften eines HANs, des vorangegangenen Praxisprojektes [1] und einer Dissertation [2], wurden die folgenden nicht-funktionalen Anforderungen für die Heimautomatisierungsdomäne identifiziert und erläutert:

- ***Low-power Embedded Devices:*** Ein Erfolgsgrund von HAN liegt in der Nutzung von sogenannten „Low-power Embedded Devices“. Diese Geräte lassen sich mit relativ geringen Kosten anschaffen und aufgrund ihrer geringen Baugröße beinnahe überall platzieren. So werden z. B. die in Kapitel 2.1 genannten Feld- und Interconnection Geräte meist als „Low-Power Embedded Devices“ umgesetzt. Diese Geräte besitzen dabei häufig unter einem MB Speicher, geringe Prozessorleistung und eine limitierte Stromzufuhr [1]. Als Stromlieferant dient meist eine Batterie oder zuteilen auch lediglich sogenannte „Energy Harvesting“ Verfahren. Daraus folgt das bei der Nutzung eines Sicherheitskonzeptes und den darin enthaltenen Kryptografischen Verfahren, besonders auf die zur Verfügung stehenden Ressourcen geachtet werden muss. Hierbei sollte immer eine Balance zwischen dem gewünschten Sicherheitsgrad und den zur Verfügung stehenden Ressourcen gefunden werden („Good Enough Security“).
- ***Skalierbarkeit:*** Im Gegensatz zu vielen bekannten Diensten aus der IT, die lediglich eine geringe Anzahl von Kommunikationspartner innerhalb einer einzelnen Kommuniaktionsbeziehung besitzen, sind in einem HAN Beziehungen mit mehreren hunderten Kommunikationspartnern denkbar wie z. B. eine Beleuchtungssteuerung eines Mehrfamilienhauses. So kann ein beispielhaftes Netzwerk in der Heimautomatisierungsdomäne aus einige wenige MG, mehreren ICG und tausenden FG bestehen. Somit muss bei der Entwicklung eines Sicherheitskonzeptes für die Heimautomatisierung, die Skalierbarkeit des Verfahrens besonders betrachtet werden.
- ***Quality of Service (QoS) Parameter:*** QoS besitzt im Rahmen der IT keine allgemeingültige Umsetzung und Definition. Jedoch werden im Allgemeinen Para-

meter aus Anwendersicht definiert, die eine Evaluierung von Diensten und Software ermöglichen. Dabei unterscheiden sich die Anforderungen an einen Heimautomatisierungsdienst zuteilen signifikant von den Anforderungen an z. B. einen Internetdienst. Ein besonders wichtiger Faktor in einem HAN ist die Echtzeitkommunikation, die in vielen anderen Anwendungsszenarien weniger relevant ist. Denn bereits wenige Sekunden können (z. B. bei einer Beleuchtungssteuerung) als unangenehm empfunden werden. Weitere wichtige Parameter sind die Zuverlässigkeit, Reihenfolge und Paketverluste der Kommunikation. Diese Parameter sind insbesondere wichtig, da eine Wiederholung der Kommunikation wieder Strom aus der limitierten Quelle kostet.

- ***Nicht vertrauenswürdige Netzwerke:*** Heutzutage wird Heimautomatisierung zur Reduzierung der Kosten und Steigerung des Anwenderkomforts, meist mithilfe von Funktechnologien umgesetzt. Somit kann eine physikalische Sicherheit der Kommunikationswege nicht gewährleisten werden. Des Weiteren werden die Heimnetzwerke meist noch über ein Gateway mit dem Internet verbunden. Somit muss mit einem zeitweiligen unberechtigten Zugriff zum Netzwerk gerechnet werden.
- ***Hohe Lebensdauer:*** Ein installiertes und konfiguriertes HAN sollte Jahre im besten Fall sogar Jahrzehnte vollständig operable bleiben. Dabei spielen vor allem die Verlässlichkeit und die Fehlertoleranz des Netzwerkes und der angebotenen Dienste eine wichtige Rolle. Es muss gewährleistet sein, dass die Dienste im Rahmen des QoS kommunizieren. Des Weiteren müssen Fehler bei der Verarbeitung und der Kommunikation von Daten behandelt werden.
- ***Änderbarkeit/Wartbarkeit:*** Damit Konfigurationen durchgeführt, neue Fehlererkennungen hinzugefügt oder lediglich Sicherheitsparameter aktualisiert werden können, müssen Verfahren implementiert werden die Änderung und Wartungen ermöglichen. Da diese Verfahren wiederrum ein mögliches Angriffsziel darstellen, müssen sie abgesichert werden.

4 Schlüsselmanagement

Moderne Kryptosysteme nutzen Schlüssel, Passwörter oder Codes zur Parametrisierung ihrer Algorithmen. Dabei steht die Komplexität und Größe der Schlüssel in direkter Relation zur Sicherheit des gesamten Systems. So lässt sich z. B. ein noch so gut gesicherter Safe mit einfachen Mitteln öffnen, sollte einem Einbrecher der verwendete Schlüssel bekannt sein. Analog hierzu kann ein fehlendes oder schlechtes Schlüsselmanagent die Sicherheit eines Systems kompromittieren.

Im Rahmen moderner Computersysteme wird zwischen drei verschiedenen Klassen von kryptografischen Algorithmen unterschieden:

- ***Kryptografische Hash-Funktionen:*** Im Allgemeinen benötigen Hash-Funktionen kein Schlüsselmaterial. Denn Hash-Funktionen werden als sogenannte Einwegfunktionen (englisch: One-Way Function) bezeichnet, die einen kleinen Hash-Wert aus einem größeren Eingabewert erzeugen. Dabei stellt die Rückrechnung eines Eingabewertes aus einem Hash-Wert ein nicht effizient lösbares mathematisches Problem dar. Im Rahmen der IT-Security werden Hash-Funktionen jedoch auch zur Sicherung der Quellen- und Nachrichtenintegrität genutzt. Hierzu werden spezielle Hash-Funktionen, die zusätzlich kollisionsresistent sein sollen, verwendet. Kollisionsresistenz bedeutet, dass sich im besten Fall keine zwei voneinander verschiedenen Eingabewerte finden, die zu einem gleichen Hash-Wert führen. Diese speziellen Hash-Funktionen werden dann zur Generierung eines sogenannten „Message Authentication Code“ (MAC) mithilfe eines Schlüssels genutzt. Des Weiteren werden Hash-Funktion im Rahmen eines Schlüsselaustausch Algorithmus zur Ableitung eines gemeinsamen Schlüssels oder zur Generierung deterministischer Zufallszahlen genutzt.
- ***Symmetrische Kryptoalgorithmen:*** Symmetrische Kryptoalgorithmen nutzen einen sogenannten symmetrischen Schlüssel, der jeder berechtigten Entität bekannt sein muss. Dabei wird der gleiche Schlüssel zum einen für die kryptografische Funktion und zum anderen für die jeweilige inverse Funktion genutzt. Diese Klasse von Algorithmen wird meist zur Sicherung der Vertraulichkeit der Kommuniaktion genutzt.
- ***Asymmetrische Kryptoalgorithmen:*** Im Gegensatz zu einem symmetrischen Algorithmus nutzen asymmetrische Algorithmen ein sogenanntes Schlüsselpaar. Meistens als „Public“ und „Private“ Schlüssel bezeichnet, besitzt jede Entität ihr eigenes Paar. Dabei darf der „Public“ Schlüssel jeder anderen Entität bekannt sein, der „Private“ Schlüssel hingegen lediglich der Entität mit dem zugehörigen „Public“ Schlüssel. Anders als die symmetrischen Algorithmen werden die asymmetrischen Algorithmen meistens für die Erzeugung von digitalen Signaturen oder zum Austausch von kryptografischem Schlüsselmaterial, das für die Generierung eines symmetrischen Schlüssels genutzt werden kann, verwendet.

Unter dem Begriff des Schlüsselmanagements wurden die Disziplinen und Herrausforderungen zusammengefasst, die bei der Nutzung von Schlüssel in der IT-Security entstehen. Für den Rahmen dieser Arbeit wurden die Generierung, die Trennung, die Speicherung, der Austausch und die Vernichtung von Schlüssel ausgewählt. In den nachfolgenden Abschnitten werden diese Teilaufgaben auf Basis der NIST [13] und BSI Empfehlungen [14] näher erläutert.

4.1 Generierung

Die Generierung oder Erzeugung von Schlüsselmaterial sollte in sicherer Umgebung und mithilfe geeigneter Generatoren durchgeführt werden. Dabei sollten die Generatoren kontrolliert, statistisch gleichverteilte Zufallsfolgen über den gesamten Schlüsselraum erzeugen. So kann z. B. eine Rauschquelle einen zufälligen Bitstream erzeugen, der mithilfe geeigneter Algorithmen zu einem Schlüssel weiterverarbeitet werden kann. Abschließend sollten die erzeugten Schlüssel mithilfe verschiedener Testverfahren auf

ihre Güte überprüft werden. Die Generierung von Schlüsseln kann sowohl direkt am Einsatzort oder zentral an einem sogenannten „Trust Center“ durchgeführt werden. In der Regel müssen bei der Erzeugung am Einsatzort Abstriche bei der Sicherheit der Umgebung gemacht werden. Bei einer zentralen Erzeugung hingegen muss sichergestellt werden, dass die Schlüssel die Empfänger authentisch und integer erreichen. Neben der Erzeugung von Schlüssel mithilfe von integrierten Zufallsgeneratoren, kann der benötigte Zufall, sollte keine integrierte Zufallsquelle vorhanden sein, auch über Benutzereingaben erhalten werden. Dafür werden Anwender zur Eingabe eines möglichst langen und gut gewählten Passworts oder zur Eingabe einer möglichst zufälligen Eingabefolge aufgefordert.

4.2 Trennung

Im Rahmen eines guten Schlüsselmanagements sollten Schlüssel lediglich für einen ausgewählten Einsatzzweck verwendet werden. Somit sollten z. B. die Schlüssel, mit denen die Kommunikation verschlüsselt wird, verschieden zu den Schlüsseln sein die für die Erzeugung von Signaturen genutzt werden. Somit besitzt ein System eine Vielzahl von verschiedenen Schlüsseln, das NIST [13, p. 44] definiert 19 verschiedene Schlüsseltypen innerhalb ihrer Schlüsselmanagement Richtlinien.

4.3 Speicherung

Schlüssel sollten niemals im Klartext, sonder immer auch eine Art verschlüsselt im System gespeichert werden. Viele moderne Mikrocontroller mit integrierten Kryptomodulen bieten speziell abgesicherte Speicherbereiche innerhalb des Chips an, die gegen einen unberechtigten Zugriff geschützt sind. Sollte kein ausreichender Zugriffsschutz auf dem Chip vorhanden sein, sollten keine Schlüssel innerhalb des Mikrocontrollers gespeichert werden. Neben einer Speicherung direkt im Mikrocontroller, kann auch ein sogenanntes „Trusted Platform Module“ (TPM) genutzt werden. Ein TPM stellt ein dediziertes Kryptomodul dar, das neben einem Zufallsgenerator und speziellen Schlüsselgeneratoren, auch einen abgesicherten Speicher für Schlüssel besitzt.

4.4 Austausch

Kryptografische Verfahren funktionieren lediglich, sollten die jeweiligen Kommuniaktionspartner aufeinander abgestimmte Schlüssel besitzen. Neben einem manuellen Transport der Schlüssel durch z. B. eine Nutzereingabe, können Schlüssel auch über einen Kommunikationsweg – wie z. B. Kabelverbindungen oder Funkverbindungen – ausgehandelt bzw. ausgetauscht werden. Dieser Austausch sollte vertraulich, authentisch und integer erfolgen. Dafür wird heutzutage die sogenannte hybride Kryptografie genutzt. Grundlegend stellt die hybride Kryptografie eine Mischung aus symmetrischen und asymmetrischen Verfahren dar. Mithilfe asymmetrischer Verfahren lässt sich sicher Schlüsselmaterial austauschen, das dann dafür genutzt werden kann einen geteilten symmetrischen Schlüssel abzuleiten. Somit kombinieren hybride Verfahren die Sicherheit der asymmetrischen Kryptografie mit der Geschwindigkeit der symmetrischen Kryptografie.

4.5 Vernichtung

Nicht mehr gebrauchte Schlüssel oder Schlüssel, deren Nutzungsdauer abgelaufen ist, sollten auf sichere Art gelöscht werden. Etablierte Verfahren nutzen mehrfaches Löschen oder Überschreiben der betroffenen Speicherzellen.

5 Bewertungsschema

Zur Evaluierung der in Kapitel 6 erläuterten Konzepte und Auswahl geeigneter Methoden für die Entwicklung eines eigenen Schlüsselmanagementkonzeptes, wird nachfolgend ein Bewertungsschema beschrieben. Dieses Schema wird auf Grundlage der in den vorangegangenen Kapiteln identifizierten und erläuterten Eigenschaften eines HAN und Anforderungen an ein Sicherheitskonzept entworfen.

Das Schema setzt sich aus den Kategorien Schlüsselmanagement Disziplinen, funktionale Anforderungen (Sicherheitsziele) und nicht funktionale Anforderungen zusammen. Zu Beginn der Bewertung werden die verschiedenen Schlüsselmanagement Disziplinen aus Kapitel 4 betrachtet. Dabei werden die jeweiligen Strategien und Verfahren genannt, die das Bewertungsobjekt verwendet, um die verschiedenen Disziplinen zu erfüllen. Hierbei wird auch auf notwendige Schlüssellängen und den somit benötigten Speicherplatz eingegangen. Nachdem die Schlüsselmanagement Disziplinen überprüft wurden, werden die funktionalen Anforderungen näher betrachtet. Da die funktionellen Anforderungen durch die Sicherheitsziele beschrieben werden, muss ein Bewertungsobjekt diese alle erfüllen, um für das eigene Konzept in Betracht zu kommen. Zum Abschluss der Evaluierung werden die nicht funktionalen Anforderungen (siehe Kapitel 3.2) betrachtet. Diese Anforderungen sollten im optimalen Fall auch alle erfüllbar sein, stellen jedoch kein direktes Ausschlusskriterium dar.

Die Ergebnisse werden zur Steigerung der Übersichtlichkeit auch in das in Abbildung 6 dargestellte Layout eingetragen. Hierbei wird zwischen vollständig erfüllt (✓), nicht erfüllt (✗) und teilweise erfüllt (⊙) unterschieden. Teilweise erfüllt gilt, sollte ein Verfahren lediglich Teilaspekte der definierten Anforderungen erfüllen. Um eine einfache und direkte Vergleichbarkeit zwischen den verschiedenen Konzepten zu ermöglichen, werden die Ergebnisse der einzelnen Kategorien auch als Zahl dargestellt. Dabei errechnet sich das Evaluierungsergebnis einer Kategorie ($\boldsymbol{x}$), durch die unten dargestellte Formel.

$$\boldsymbol{x = 2 * 1 * 2 * 2 * 2}$$

Abbildung 5: Ergebnisformel des Bewertungsschemas

Für jede vollständig erfüllte Bedingung wird eine Zwei hinzugefügt. Für jede teilweise erfüllte Bedingung eine Eins und für eine nicht erfüllte Bedingung eine Null. Somit werden vollständig erfüllte Bedingung stärker gewichtet als teilweise erfüllte und nicht erfüllte resultieren in einer Null als Ergebnis. Beispielsweise ist das beste Ergebnis, das eine Kategorie erhalten kann, $\mathbf{32}$ ($\mathbf{2^5 = 2 * 2 * 2 * 2 * 2}$) und das nächst beste Ergebnis wäre **16** ($\mathbf{2^4 = 2(* 2 * 2 * 1}$).

Bewertungsobjekt			
Schlüsselmanagement Disziplinen:			**32**
Generierung	✓	Schlüssellänge > 1024 Bit	
Trennung	✗		
Speicherung	⊙		
Austausch			
Vernichtung			

Funktionale Anforderungen (Sicherheitsziele)		**32**
Vertraulichkeit	✓	
Integrität	✗	
Verfügbarkeit	⊙	
Authentizität		
Aktualität		
Nicht-funktionale Anforderungen		
Low-power		
Skalierbar		
Hohe Lebens-dauer		
Wartbarkeit		

Abbildung 6: Bewertungsschema Layout

6 State of the Art

Nachdem die grundlegenden Eigenschaften von Heimautomatisierungsnetzwerken erläutert und basierend darauf ein Bewertungsschema definiert wurde, wird im nachfolgenden Kapitel ein genauerer Blick auf den aktuellen Stand der Forschung geworfen. Dafür werden neben verschiedenen etablierten Verfahren und Protokolle aus der Informatik, auch Konzepte aus der Forschung näher betrachtet und evaluiert. Die so gewonnenen Erkenntnisse werden im Anschluss für die Entwicklung eines eigenen Schlüsselmanagement Konzeptes genutzt.

Bereits vor der Entwicklung der ersten Computer wurden kryptografische Verfahren dazu genutzt Informationen vor unerlaubten Lesen zu schützen. Diese recht simplen Techniken mussten damals noch mittels Papier und Stift umgesetzt werden. Doch bereits hier mussten die Schlüssel, die für die Entschlüsselung der Informationen benötigt werden, der jeweiligen Empfangsstelle bekannt sein oder über Transportwege mitgeteilt werden. Die Erzeugung, die Speicherung und der Transport des Schlüssels stellen somit einen kritischen Bestandteil der Kryptografie dar.

Im Rahmen eines Praxisprojektes [1] wurden die populärsten Heimautomatisierungsprotokolle ZigBee, EnOcean und Z-Wave näher analysiert. Dabei wurden die jeweiligen Kommunikationsprotokolle, insbesondere das Schlüsselmanagementkonzept, mithilfe öffentlich verfügbarer Dokumentationen untersucht. Dabei konnte gezeigt werden, das die etablierten Heimautomatisierungsprotokolle zur Sicherung der Vertraulichkeit und Integrität der Daten den AES-128 Algorithmus nutzen. Der AES (Advanced Encryption Standard) benötigt einen unter den Kommunikationspartner geteilten Schlüssel, der für die Ver- und Entschlüsselung genutzt wird. Somit brauchen die untersuchten Standards ein Verfahren, wie die jeweiligen Kommunikationspartner einen gemeinsamen Schlüssel sicher generieren oder empfangen können. Diese Verfahren bestanden bei allen untersuchten Herstellern daraus, dass sogenannte Master- oder Standard Schlüssel bei der Chipproduktion fest eingespeichert oder über unsichere Kommunikationskanäle ausgetauscht wurden. Neben dem klaren Sicherheitsproblem mit unsicher übertragenden Schlüsseln, stellt ein fest eingespeicherter Schlüssel, der über alle produzierten Chips identisch ist, ein weiteres Problem dar.

6.1 Diffie-Hellman Schlüsselaustausch

Der Diffie-Hellman Schlüsselaustausch gilt als das erste asymmetrische Schlüsselaustauschverfahren, das die sichere Erzeugung eines geteilten Schlüssels über eine unsichere Leitung ermöglichte. Dabei wird die Sicherheit des Verfahrens durch das diskrete Logarithmus Problem gewährleistet. Das diskrete Logarithmus Problem bezeichnet den Umstand, das die diskrete Exponentialfunktion eine Einwegfunktion darstellt. Damit ist gemeint das die diskrete Exponentialfunktion ($\boldsymbol{b^x \; mod \; m}$) auch für große Exponenten effizient berechnet werden kann, die Umkehrfunktion, der diskrete Logarithmus, jedoch nicht.

Zu Beginn des Verfahrens einigen sich die beiden Kommunikationspartner (Alice und Bob) auf eine zyklische Gruppe primer Ordnung **p** und einen Generator dieser Gruppe **g.** Die Aushandlung dieser Parameter muss nicht geheim erfolgen und kann somit auch über einen unsicheren Kanal erfolgen. Im Anschluss an die Aushandlung der Parameter, wählen die Kommunikationspartner jeweils zufällig eine Zahl aus der Menge **{1, …, p-1}.** Diese beiden Zahlen **a** und **b** stellen den privaten Schlüsselanteil der Kommunikationsinstanzen dar und müssen daher geheim gehalten werden. Jetzt werden wie in Abbildung 7 zu sehen ist mithilfe der geheimen Zufallszahlen und den öffentlichen Werten die Parameter **A** und **B** berechnet. Diese Parameter werden wieder über den unsicheren Kanal ausgetauscht und für die Berechnung des endgültigen Schlüssels **K** genutzt.

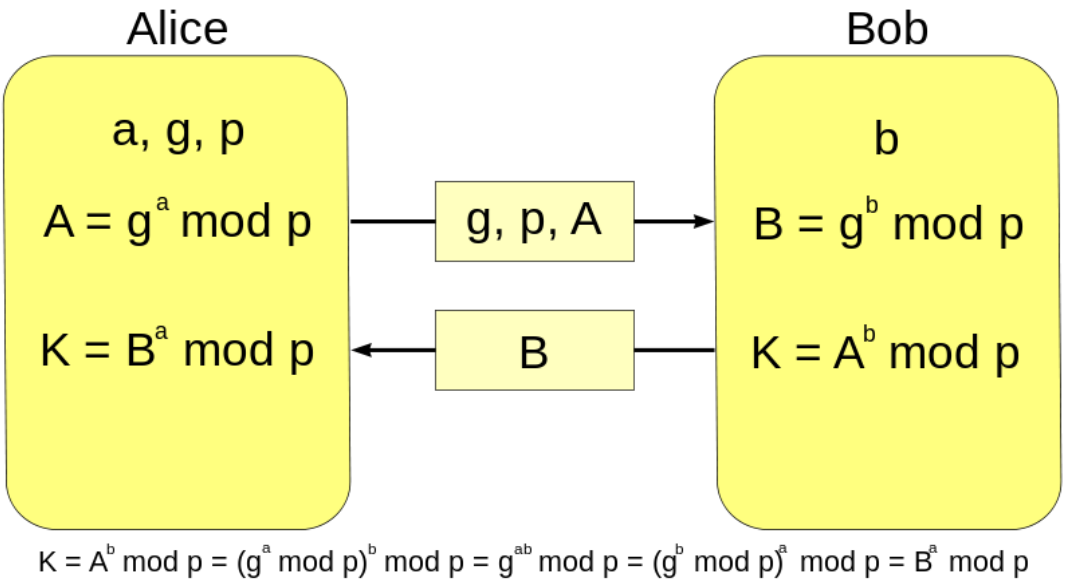

Abbildung 7: Diffie-Hellman Protokoll

Anonyme Schlüsselaustauschverfahren wie Diffie-Hellman besitzen keine Authentifizierung der Kommunikationspartner, wodurch ein „Man-in-the-Middle“ Angriff ermöglicht wird. Sollten die gesendeten Parameter nicht integritätsgeschützt sein, ist es einem Angreifer möglich einen Schlüsselaustausch zwischen ihm und den beiden Kommunikationspartnern zu initiieren. Hierzu muss er lediglich eine eigene Zufallszahl berechnen und diese für die Berechnung an Alice und Bob senden. Danach besitzt der Angreifer einen Schlüssel für die Kommunikation mit Alice und einen anderen Schlüssel für die Kommunikation mit Bob. Somit ist es ihm jetzt möglich eine direkte Kommunikation zwischen den beiden Partner zu simulieren und gleichzeitig die gesamte Kommunikation zu belauschen.

Im Laufe der Zeit wurden eine Reihe von Verfahren und Protokollen entwickelt, die einen authentifizierten Schlüsselaustausch ermöglichen. Die nachfolgenden Kapitel erläutern die verbreitesten Strategien und Protokoll für einen sicheren Schlüsselaustausch.

6.2 Needham-Schroeder-Protokoll

Das Needham-Schroeder-Protokoll ermöglicht einen Schlüsselaustausch inklusive einer Authentifizierung innerhalb eines dezentralen Netzwerkes. Somit besitzt jedes Netzwerk neben den Kommunikationspartnern auch eine sogenannte vertrauenswürdige Instanz. Das Protokoll gibt es in einer symmetrischen und asymmetrischen Ausführung. Die symmetrische Version des Protokolls setzt bereits verteilte symmetrische Schlüssel zwischen den Kommunikationspartnern und der vertrauenswürdigen Instanz voraus. Da ein solches Schlüsselmanagement-Verfahren im vorausgelaufenen Praxisprojekt als unzureichend bewertet wurde, wird im Nachfolgenden lediglich die asymmetrische Variante des Protokolls betrachtet.

Ein Netzwerk besitzt eine zentrale und vertrauenswürdige Instanz (**T**). Des Weiteren besitzt es mindestens die beiden Kommunikationsinstanzen Alice (**A**) und Bob (**B**). Alle Netzwerkteilnehmer besitzen ein Schlüsselpaar bestehend aus einem privaten und einem öffentlichen Schlüssel. **T** kennt von jedem Netzwerkteilnehmer den öffentlichen Schlüssel und jedem Netzwerkteilnehmer ist Wiederrum der öffentliche Schlüssel von **T** bekannt. Sollte **A** jetzt eine verschlüsselte Nachricht an **B** senden wollen, benötigt **A** den öffentlichen Schlüssel von **B**. Da der direkte Transport des Schlüssels über einen „Man-in-the-Middle“ manipuliert werden könnte, wird dieser über **T** authentifiziert. Dafür sendet **A** eine Nachricht mit seiner eigenen Identität und der gewünschten Zielidentität unverschlüsselt an **T**. **T** antwortet **A** mit der Identität von **B** und dem zugehörigen öffentlichen Schlüssel ($\mathbf{P_B}$). Zur Sicherung der Integrität signiert **T** das Paket mit seinem privaten Schlüssel. Jetzt kann **A** seine Identität und eine generierte Zufallszahl ($\mathbf{N_A}$) mit dem

Schlüssel $\mathbf{P_B}$ verschlüsseln und an **B** senden. Lediglich **B** kann das empfangene Paket mit seinem privaten Schlüssel entschlüsseln. Da **B** anhand der empfangenen Informationen die Authentizität von **A** nicht gewährleisten kann und eventuell den öffentlichen Schlüssel von **A** nicht kennt, erfragt **B** den Schlüssel $\mathbf{P_A}$ von **T**. Jetzt sendet **B** die Zufallszahl $\mathbf{N_A}$ und eine selbsterzeugte Zufallszahl ($\mathbf{N_B}$) mit dem Schlüssel $\mathbf{P_A}$ verschlüsselt an **A**. Im letzten Protokollschritt sendet **A** die erhaltene und entschlüsselte Zufallszahl $\mathbf{N_B}$ mit dem Schlüssel $\mathbf{P_B}$ verschlüsselt zurück an **B**. Die letzten beiden Schritte stellen die Aktualität der Verbindung und die Authentizität der beiden Kommunikationspartner sicher. Das Verfahren ist sicher, solange einem Angreifer kein privater Schlüssel in die Hände fällt. Denn sollte ein Angreifer ein Schlüsselpaar von **A**, **B** oder sogar **T** erbeuten, kann er die jeweilige Rolle übernehmen.

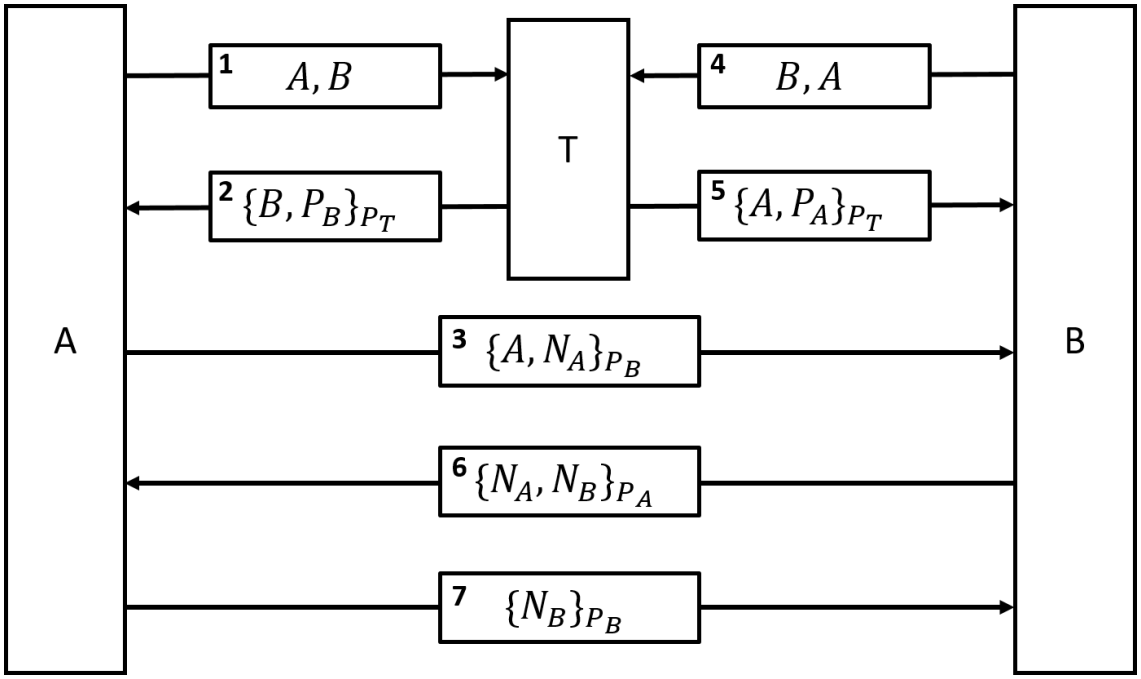

Abbildung 8: Needham-Schroeder-Protokoll

6.3 Out-of-Band Schlüsselaustausch

Neben einem Schlüsselaustausch über dasselbe Medium über das auch der eigentliche Datentransport durchgeführt wird, können Schlüssel auch über andere Kommunikationskanäle und Verfahren ausgetauscht werden. Im Laufe der Zeit haben sich so verschiedenen sogenannte „Out-of-Band“ Konzepte entwickelt. Paradebeispiele für ein sogenanntes „Out-of-Band“ Konzept sind die Zwei-Faktor-Authentifizierung und moderne Online-Banking Lösungen. So wird z. B. bei einer Zwei-Faktor-Authentifizierung für einen Identitätsnachweis eines Nutzers eine Kombination zweier unabhängigen Faktoren genutzt. Hierbei können z. B. das Wissen eines Passwortes mit dem Besitz eines Token kombiniert werden. Ein solches Token kann z. B. ein USB-Stick oder ein RFID Transceiver sein. Beim Online-Banking hingegen werden Sicherheitscodes per SMS übermittelt, die Wiederrum für die Anmeldung an der Online-Plattform genutzt werden. In beiden Fällen basiert die Sicherheitsannahme darauf, das ein möglicher Angreifer entweder nicht alle Kanäle abhören kann oder nicht im Besitz des benötigten Token ist. Ein weiteres Konzept findet sich bei der Verteilung von Netzwerkschlüsseln – z. B. W-Lan Schlüssel – mithilfe von QR-Codes. QR-Codes sind zweidimensionale Codes, die sich mit geeigneten Lesegeräten und heutzutage auch mittels Smartphone Applikationen auslesen lassen. Aufgrund einer automatischen Fehlerkorrektur innerhalb des Codes ist das Verfahren äußerst robust. Möchte ein Nutzer seinen Netzwerkschlüssel möglicht einfach

verteilen, kann er aus dem Schlüssel einen QR-Code erzeugen und diesen an berechtigte Teilnehmer verteilen. Eine besondere Umsetzung dieses Konzeptes, wird mithilfe von sogenannten RFID-Transponder ermöglicht. Ein RFID Transponder ist eine kleine integrierte Schaltung mit Antenne, die als Informationsspeicher genutzt werden kann. Dabei wird zwischen aktiven und passiven Transponder unterschieden, wobei aktive Transponder eine eigene Stromversorgung besitzen und passive lediglich über die induzierte Lesespannung, die ein RFID-Leser erzeugt, betrieben werden. So kann ein erzeugter Schlüssel innerhalb eines RFID-Transponders gespeichert werden und von entsprechenden Lesegeräten ausgelesen werden.

6.4 Public-Key-Infrastruktur

Eine Public-Key-Infrastruktur (PKI) ist ein asymmetrisches Kryptosystem, das eine authentische Schlüsselverteilung über sogenannte digitale Zertifikate ermöglicht. Dafür müssen innerhalb der PKI eine Reihe von Rollen, Regeln und Verfahren zum Erzeugen, verwalten, verteilen, nutzen und widerrufen von digitalen Zertifikaten bereitgestellt werden. Abbildung 9 zeigt die verschiedenen Rollen und Abläufe einer PKI. Kern einer jeden PKI ist die sogenannte Zertifizierungsstelle (englisch: „Certificate Authority“) kurz CA. Aufgabe der CA ist die Ausgabe und Verifizierung digitaler Zertifikate. Ein digitales Zertifikat ist ein Datensatz, der grundlegend aus einem öffentlichen Schlüssel, Informationen zum Besitzer und einer digitalen Signatur, die das Zertifikat verifiziert, besteht. Dabei wird die digitale Signatur von einer CA mithilfe ihres privaten Schlüssel generiert. Somit können Dritte die Signatur mit dem öffentlichen Schlüssel der CA überprüfen. Damit diese Art der Verifizierung funktionieren kann, müssen die Teilnehmer der CA und der Echtheit ihres öffentlichen Schlüssels vertrauen. Solche CA´s werden auch Root-CA genannt. Damit ein digitales Zertifikat von der CA signiert werden kann, muss dieses zuallererst erzeugt werden. Diesen Prozess bezeichnet man auch als registrieren. Dabei sendet ein Anwender die benötigten Informationen – z. B. Name, Adresse und Verwendungszweck – an eine CA. Diese überprüft die Angaben auf ihre Richtigkeit und fügt den entsprechenden Schlüssel in das Zertifikat ein. Dabei kann das benötigte Schlüsselpaar direkt von der CA erzeugt und der private Schlüssel dem Anwender z. B. per Post zugesendet werden oder das Schlüsselpaar wird vom Nutzer selber generiert und lediglich der öffentliche Schlüssel wird übermittelt. Bei diesem Verfahren muss die CA die Authentizität des Schlüssels jedoch noch gewährleisten. Zur Entlastung einer CA kann der gesamte Registrierungsprozess in eine eigenen Instanz, die sogenannte Registrierungsstellen (englisch: „Registration Authority“) kurz RA, ausgelagert werden. Möchte jetzt z. B. ein Kunde einem Online-Shop – wie in Abbildung 9 zu sehen ist – eine verschlüsselte Nachricht senden, so benötigt der Shop zum Entschlüsseln den öffentlichen Schlüssel des Kunden. Anstatt diesen einfach direkt an den Shop zu senden, sendet der Kunde sein von der CA signiertes Zertifikat. Der Shop benötigt jetzt noch eine Möglichkeit die Authentizität des Zertifikates zu gewährleisten, dafür kann er die Signatur des Zertifikates mithilfe des öffentlichen Schlüssels der CA überprüfen.

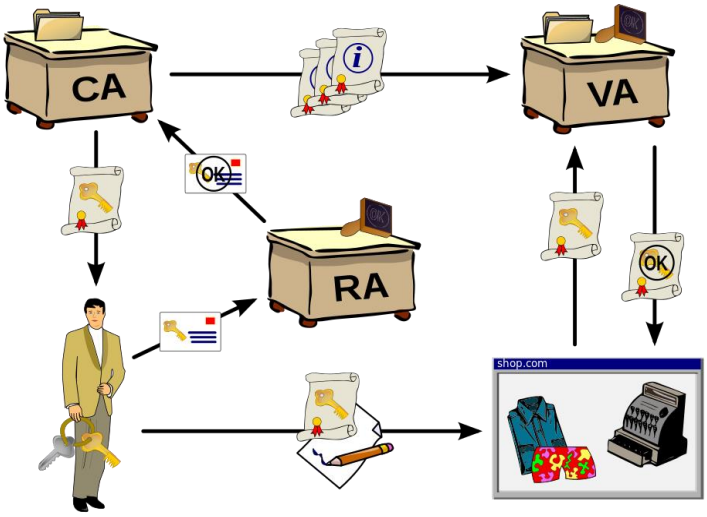

Abbildung 9: Rollen einer Public-Key-Infrastruktur [15]

Die heutzutage am häufigsten verwendete Zertifikatsstruktur wird durch den X.509 Standard, der Internationalen Fernmeldeunion (ITU), definiert. Abbildung 10**Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt den grundlegenden Inhalt eines solchen X.509 Zertifikates. Neben den allgemeinen Informationen zum Empfänger und dem öffentlichen Schlüssel, beinhaltet das Zertifikat noch Informationen zu den genutzten Algorithmen zur Schlüsselgenerierung und Signierung. Die Informationen zum Zertifikatsausteller können von Dritten dazu genutzt werden, den öffentlichen Schlüssel der CA abzufragen oder aus einer Liste mit gespeicherten CA Schlüsseln auszuwählen. Ein Zertifikat besitzt immer eine limitierte Gültigkeit. Jedoch kann es auch vorkommen das Zertifikate vor ihrem eigentlichen Ablaufdatum für ungültig erklärt werden müssen. Dafür werden sogenannte Zertifikatssperrlisten veröffentlicht, die anhand der Seriennummer ungültige Zertifikate auflistet. Neben diesen Listen können aber auch sogenannte Validierungsdienste (englisch: „Validation Authority) kurz VA implementiert werden, die eine direkte Überprüfung einzelner Zertifikate über das Internet ermöglichen.

Inhalt	**Erläuterung**
Versionsnummer	Beschreibt das verwendete Zertifikatformat
Seriennummer	Eindeutiger Identifikator
Algorithmen-ID	Identifikator für den verwendeten Signaturalgorithmus
Zertifikatsausteller	Name der ausstellenden Instanz
Gültigkeitsdauer	Angabe eines Gültigkeitszeitraumes
Zertifikatinhaber	Eindeutiger Name des Benutzers
Zertifikatinhaber -Schlüsselinformationen	Public-Key-Algorithmus Public-Key des Zertifikatinhaber
Zertifikat-Signatur	Signatur des Zertifikats

Abbildung 10: Inhalt eines X.509 Zertifikates

6.4.1 RSA/SHA-1

Die Kombination des Rivest, Shamir und Adleman (RSA) Algorithmus und dem „Secure Hash Algorithm“ (SHA) Version 1 ist die verbreitetste Kombination für die Umsetzung einer PKI. RSA ist ein asymmetrischer Kryptoalgorithmus, der sowohl zum Verschlüsseln als auch zum Signieren von Daten genutzt werden kann. Unter der Abkürzung SHA veröffentlich und standarisiert das „National Institute of Standards and Technology“ (NIST) seit 1993 kryptografische Hashfunktionen.

Erzeugung des Schlüsselpaares

Das Schlüsselpaar innerhalb des RSA Algorithmus besteht aus zwei Zahlenpaaren. Dabei setzt sich der private Schlüssel aus $(\boldsymbol{d}, \boldsymbol{N})$ zusammen und der öffentliche Schlüssel aus $(\boldsymbol{e}, \boldsymbol{N})$. $\boldsymbol{N}$ ist innerhalb eines Schlüsselpaares gleich und wird RSA-Modul genannt, $\boldsymbol{e}$ ist der sogenannte Verschlüsselungsexponent und $\boldsymbol{d}$ der Entschlüsselungsexponenten. Die folgenden Schritte erläutern die Erzeugung der einzelnen Zahlen:

1. Es werden zufällig und stochastisch zwei unabhängige Primzahlen $\boldsymbol{p} \neq \boldsymbol{q}$ gewählt. Die gewählten Primzahlen sollten innerhalb des Rahmens $\mathbf{0,1} < |\mathbf{log_2}\, \boldsymbol{p} - \mathbf{log_2}\, \boldsymbol{q}| < \mathbf{30}$ liegen.
2. Berechne das RSA-Modul $\boldsymbol{N} = \boldsymbol{p} * \boldsymbol{q}$.
3. Basierend darauf wird mit der Eulersche φ-Funktion $\boldsymbol{\varphi}(\boldsymbol{N}) = (\boldsymbol{p} - \mathbf{1}) * (\boldsymbol{q} - \mathbf{1})$ berechnet.
4. Jetzt wird eine zu $\boldsymbol{\varphi}(\boldsymbol{N})$ teilerfremde Zahl $\boldsymbol{e}$ gewählt, für die gilt $\mathbf{1} < \boldsymbol{e} < \boldsymbol{\varphi}(\boldsymbol{N})$.
5. Berechne $\boldsymbol{d}$ als multiplikatives von $\boldsymbol{e}$ bezüglich $\boldsymbol{\varphi}(\boldsymbol{N})$, sodass die Kongruenz $\boldsymbol{e} * \boldsymbol{d} \equiv \mathbf{1}\ \boldsymbol{mod}\ \boldsymbol{\varphi}(\boldsymbol{N})$ gilt.

Nachdem $\boldsymbol{N}$, $\boldsymbol{e}$ und $\boldsymbol{d}$ berechnet wurden, werden die Primzahlen $\boldsymbol{p}$ und $\boldsymbol{q}$, sowie $\boldsymbol{\varphi}(\boldsymbol{N})$ nicht weiter benötigt.

Verschlüsseln von Nachrichten

Eine Nachricht $\boldsymbol{m}$ wird mit der folgenden Formel zum Geheimtext $\boldsymbol{c}$ verschlüsselt. Dabei muss $\boldsymbol{m}$ kleiner als der RSA-Modul sein.

$$\boldsymbol{c} \equiv \boldsymbol{m}^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

Abbildung 11: RSA Verschlüsselungsalgorithmus

Entschlüsseln von Nachrichten

Eine Chiffrat $\boldsymbol{c}$ lässt sich mithilfe einer ähnlichen Formel wieder entschlüsseln. Hier wird der asymmetrische Charakter der Funktionen erkennbar. Denn Daten, die mit dem öffentlichen Schlüssel verschlüsselt wurden, lassen sich lediglich mit dem privaten Schlüssel wieder entschlüsseln.

$$\boldsymbol{m} \equiv \boldsymbol{c}^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

Abbildung 12: RSA Entschlüsselungsalgorithmus

Signieren von Nachrichten

Zum Signieren von Nachrichten wird die gezeigte Verschlüsselungsfunktion mit dem privaten Schlüssel durchgeführt. Damit können dritte mithilfe des öffentlichen Schlüssels die Signatur entschlüsseln und somit die Nachrichtenauthentizität der Nachricht bestätigen. Da der öffentliche Schlüssel jedem zugänglich gemacht wird, kann theoretisch jeder, der die übertragende Signatur abfängt den Nachrichteninhalt lesen. Um dieses Problem zu umgehen, werden kryptografische Hash-Funktionen verwendet, um den Nachrichteninhalt zu verbergen.

$$\boldsymbol{s \equiv (H(m))^{e} \; mod \; N}$$

Abbildung 13: RSA Signaturalgorithmus

Bewertung des Konzepts

Nachfolgend wird das in Kapitel 5 beschriebene Bewertungsschema auf die Kombination des RSA Kryptoalgorithmus mit einer SHA-1 Hashfunktion angewendet. Die Schlüsselgenerierung innerhalb einer PKI kann – wie bereits in Kapitel 6.3 beschrieben wurde – sowohl vom Anwender als auch von der CA durchgeführt werden. Bei einer Generierung durch die CA würde das Schlüsselaustausch Problem nicht gelöst, sondern lediglich verlagert werden. Denn das erzeugte Schlüsselpaar muss jetzt irgendwie zum Anwender transportiert werden. Die größte Problematik beim RSA Kryptosystem sind die benötigten Schlüssellängen. So gelten im allgemeinen Schlüssel kleiner 2048 Bit nicht zukunftssicher und sollen in den nächsten Jahren unzureichend werden für ein sicheres Kryptosystem. Des Weiteren gestaltet sich die Schlüsseltrennung bei einer PKI ebenfalls komplizierter als in einem symmetrischen Verfahren. Bei der symmetrischen Kryptografie können einfach mehrere Schlüssel für verschiedenen Zwecke abgespeichert werden. Bei einer PKI müssen für diesen Fall verschiedenen Zertifikate angefordert und von der CA erzeugt werden. Somit steigt der benötigte Speicher und Kommunikationsaufwand erheblich an. Insbesondere im Rahmen eines HAN wo Systemressourcen limitiert sind und viele Geräte auf Batteriebasis betrieben werden, reduziert ein hoher Kommuniaktionsaufwand die aktive Laufzeit der Hardware. Der große Vorteil eines asymmetrischen Systems wie einer PKI liegt im einfachen Austausch von Schlüsseln. Da die öffentlichen Schlüssel einzelner Kommuniaktionspartner über unsichere Kanäle übertragen werden können, ohne die Sicherheit des Systems zu korrumpieren. Die Vernichtung von Schlüsseln wird im Rahmen einer PKI, wie in Kapitel 6.3 erläutert wurde über sogenannte Zertifikatssperrlisten ermöglicht.

Nachfolgend wird das Konzept auf die Erfüllung der Sicherheitsziele hin geprüft. Die Verschlüsselung und das Signieren von Nachrichten, dienen dem vertraulichen und integren Transport. Des Weiteren kann innerhalb des RSA/SHA-1 Konzeptes sowohl die Nachrichtenauthentizität, wie auch die Teilnehmerauthentizität gewährleistet werden. Nachrichtenauthentizität wird über eine Signatur Erzeugung mithilfe des privaten Schlüssels erreicht und Teilnehmerauthentizität über die Zertifikatssignaturen der CA. Die Verfügbarkeit der angebotenen Dienste wird durch das Konzept nicht eingeschränkt. Durch die meist längere Nutzungsdauer eines Schlüsselpaares innerhalb einer PKI und deren Nutzung für jede durchgeführte Operation, gilt die Aktualität lediglich als teilweise erfüllt.

Zum Abschluss der Bewertung werden die nicht-funktionalen Anforderungen betrachtet. Der gesteigerte Kommunikationsaufwand durch die Zertifikatsanforderung, den Zertifikatsaustausch und die Zertifikatsüberprüfung – wie auch die großen Schlüssel (>2000 Bit) – machen ein reines PKI Verfahren für ein HAN eher ungeeignet. Da diese meist geringen Speicher und limitierte Energieressourcen besitzen. Eine Zertifikatsstruktur ermöglicht jedoch eine einfache Skalierbarkeit und Wartbarkeit des Sicherheitssystems. Da mithilfe der Sperrlisten vorhandene Schlüssel aus dem Verkehr gezogen werden und anschließend neue verteilt werden können. Beim QoS müssen Wiederrum durch die großen Schlüssel und die damit langsamere Berechnung Abstriche gemacht werden. Denn wie bereits in Kapitel 3.2 erläutert wurde, stellt die Echtzeitkommunikation eine wichtige Rolle innerhalb eines HAN dar.

RSA/SHA-1			
Schlüsselmanagement Disziplinen:			**8**
Generierung	✓	Schlüssellänge > 2000 Bit	
Trennung	⦿	Nutzung mehrerer Zertifikate	
Speicherung	⦿	Gesteigerter Speicherbedarf durch weniger relevante Zertifikatinformationen	
Austausch	✓		
Vernichtung	✓	Zertifikatssperrlisten	
Funktionale Anforderungen (Sicherheitsziele)			**16**
Vertraulichkeit	✓	Verschlüsseln von Nachrichten	
Integrität	✓	Signieren von Nachrichten (SHA-1)	
Verfügbarkeit	✓		
Authentizität	✓	Signatur und Zertifikate	
Aktualität	⦿		
Nicht-funktionale Anforderungen			
Low-power	✗		
Skalierbar	✓		
Hohe Lebensdauer	✗		
Wartbarkeit	✓		

Abbildung 14: RSA/SHA-1 Bewertungsergebnis

6.4.2 ECC/SHA-2

Neben dem RSA Algorithmus hat sich in den letzten Jahren ein weiteres asymmetrischer Verfahren für die Umsetzung einer PKI verbreitet. Die sogenannte Elliptische Kurven Kryptografie (ECC). Die ECC nutzt Operationen, die auf elliptischen Kurven über endlichen Körpern basieren. Dabei liegt die Sicherheit des Verfahrens im Problem des diskreten Logarithmus in Elliptischen Kurven. Diese Problem ist mathematisch deutlich schwerer als das Problem des diskreten Logarithmus in endlichen Körpern, worauf der RSA Algorithmus, beruht. Dadurch können ECC Systeme mit erheblich kürzeren Schlüsseln, dieselbe Sicherheit wie RSA Systemen mit wesentlich längeren Schlüsseln, erreichen. So bietet nach Angaben des NIST z. B. ein 3072 Bit RSA oder Diffie-Hellman Schlüssel, dieselbe Sicherheit wie ein 128 Bit ECC Schlüssel. Gleichzeitig können alle Kryptografie-Systeme, die auf Basis des diskreten Logarithmus Problem in endlichen Körpern entwickelt sind, mit einfachen Mitteln auf elliptische Kurven umgestellt werden.

Aufgrund dessen gibt es einen „Elliptic Curve Diffie-Hellman“ (ECDH) und einen sogenannten „Elliptic Curve Digital Signature Algorithm“ (ECDSA). Der Nachteil bei der ECC liegt im Berechnungsaufwand. Im Allgemeinen sind die mathematischen Operationen innerhalb der ECC aufwendiger zu berechnen, als bei gleicher Schlüssellänge im RSA. Durch die deutlich kürzeren Schlüsseln können ECC Systeme jedoch abhängig von der Implementierung auch schneller als RSA Systeme sein [16].

Mathematische Grundlage

Eine elliptische Kurve über einem endlichen Körper $\boldsymbol{K}$ ist die Menge von Punkten $(\boldsymbol{x}, \boldsymbol{y})$, die von einer gegebenen Gleichung erzeugt werden. Eine solche Gleichung kann z. B. $\boldsymbol{y^2 = x^3 + Ax + B}$ sein. Zusammen mit einem zusätzlichen Punkt, der sogenannte Punkt 0 (Punkt an Unendlich), bildet eine elliptische Kurve eine additive zyklische Gruppe. Das heißt die Kurve enthält mindestens einen Punkt, mithilfe dessen jeder andere Punkt der Kurve erreicht werden kann. Die für die Kryptografie wichtigen Operationen auf der elliptischen Kurve sind die Addition zweier Punkte $(\boldsymbol{P} + \boldsymbol{Q})$ und die Potenzierung eines Exponenten mit einem Punkt $(\boldsymbol{P^x})$. Wobei die Verknüpfung durch eine wiederholte Addition $(\boldsymbol{P^x = P + P + P})$ berechnet wird. Die Addition zweier Punkte $(\boldsymbol{P} + \boldsymbol{Q})$ lässt sich grafisch darstellten (siehe Abbildung 15). Dafür wird eine Gerade durch die jeweiligen Punkte angelegt, sodass sie die Kurve an genau einer Stelle schneidet. Die Summe der Punkte ist dann durch die Spiegelung des Schnittpunktes dargestellt.

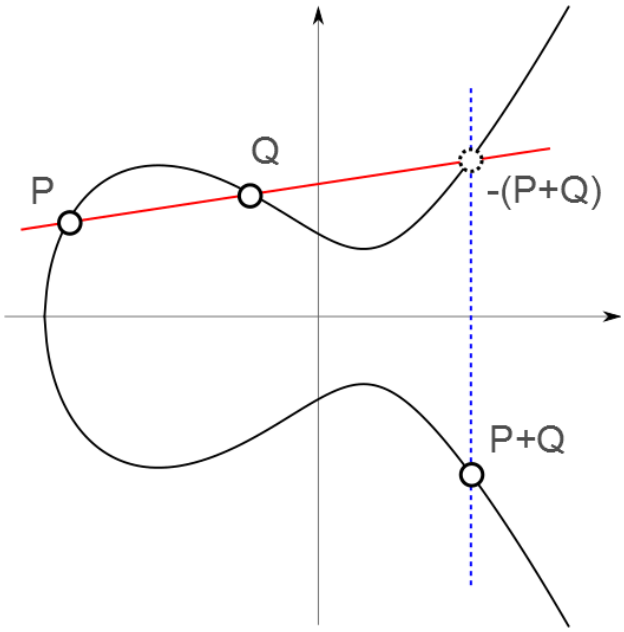

Abbildung 15: Elliptische Kurven Addition [17]

Erzeugung des Schlüsselpaares

Die Grundlage einer PKI auf Basis elliptischer Kurven ist die Einigung der Teilnehmer auf eine ausgewählte elliptische Kurve. Hierbei können Kurven von namhaften Institutionen – wie z.B. dem NIST – bezogen werden oder eigene elliptische Kurven auf Basis einiger Kriterien zufällig berechnet werden. Diese Kriterien sind in Standards wie dem ANSI X9.62 oder dem IEEE P1363 definiert. Das wichtigste Kriterium dabei ist die Menge von Punkten auf der Kurve. In der Regel soll diese mehr als $\boldsymbol{2^{160}}$ Punkte enthalten. Zu Beginn wählt oder erzeugt die CA eine sichere elliptische Kurve und speichert die sogenannten Kurvenparameter $(\boldsymbol{E}, \boldsymbol{p}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{G}, \boldsymbol{n})$ ab. $\boldsymbol{E}$ steht für die erzeugende Formel der elliptischen Kurve. Die Größe des endlichen Körpers $(\boldsymbol{p})$ auf dem die elliptische Kurve

gebildet wird. $\boldsymbol{A}$ und $\boldsymbol{B}$ sind die Koeffizienten von $\boldsymbol{E}$. $\boldsymbol{G}$ ist ein ausgewählter Generator-Punkt der elliptischen Kurve, dessen Ordnung $\boldsymbol{n}$ eine Primzahl sein muss.

Zur Berechnung eines eigenen Schlüsselpaares benötigen Teilnehmer lediglich die Kurvenparameter. Diese können von der jeweiligen CA oder falls vorhanden einer RA angefordert werden. Ein Schlüsselpaar ($\boldsymbol{x}, \boldsymbol{Q}$) besteht aus einer Zufallszahl $\boldsymbol{x}$ innerhalb des Intervall ($\boldsymbol{1} < \boldsymbol{x} < \boldsymbol{n}$) und einem Kurvenpunkt $\boldsymbol{Q}$, der mithilfe des ausgewählten Kurvenpunktes $\boldsymbol{G}$ ($\boldsymbol{Q} = \boldsymbol{G}^{\boldsymbol{x}}$) berechnet wird.

Verschlüsseln von Nachrichten

Eine direkte Verschlüsselung mithilfe der ECC, wird im Allgemeinen über das sogenannte Elgamal-Verschlüsselungsverfahren umgesetzt. Das Elgamal-Verschlüsselungsverfahren wurde vom Kryptologen Taher Elgamal 1985 veröffentlich und basiert auf der Idee des Diffie-Hellman-Schlüsselaustausches. Nachfolgend werden die einzelnen Schritte erläutert:

1. Der Sender möchte die Nachricht $\boldsymbol{m}$ verschlüsselt versenden.
2. Der Sender berechnet mithilfe des öffentlichen Schlüssels des Empfängers ($\boldsymbol{B} = \boldsymbol{b} * \boldsymbol{G}$), seinem privaten Schlüssel $\boldsymbol{a}$ und der Nachricht das Chiffrat $\boldsymbol{c}$ (siehe Abbildung 16).
3. Der Sender verschickt das Chiffrat zusammen mit seinem öffentlichen Schlüssel ($\boldsymbol{A} = \boldsymbol{a} * \boldsymbol{G}$) an den Empfänger.

$$\boldsymbol{c} \equiv \boldsymbol{B}^{\boldsymbol{a}} * \boldsymbol{m}$$

Abbildung 16: Elgamal-Verschlüsselungsalgorithmus

Entschlüsseln von Nachrichten

Mithilfe der bekannten Informationen kann der Empfänger jetzt das empfangene Chiffrat entschlüsseln. Hierfür nutzt der Empfänger die Formel:

$$\boldsymbol{m} \equiv \boldsymbol{A}^{-\boldsymbol{b}} * \boldsymbol{c}$$

Abbildung 17: Elgamal-Entschlüsselungsalgorithmus

Dies ist möglich da die folgende Gleichung gilt:

$$\boldsymbol{A}^{-\boldsymbol{b}} * \boldsymbol{c} = \boldsymbol{G}^{-\boldsymbol{ba}} * \boldsymbol{B}^{\boldsymbol{a}} * \boldsymbol{m} = \underbrace{\boldsymbol{G}^{-\boldsymbol{ba}} * \boldsymbol{G}^{\boldsymbol{ba}}}_{=\boldsymbol{1}} * \boldsymbol{m} = \boldsymbol{m}$$

Abbildung 18: Elgamal Entschlüsselungsgleichung

Signieren von Nachrichten

Das Signieren von Nachrichten wird mithilfe des „Elliptic Curve Digital Signature Algorithm“ (ECDSA) durchgeführt. Grundlage hierfür ist das sich die einzelnen Partner auf Kurvenparameter ($\boldsymbol{E}, \boldsymbol{p}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{G}, \boldsymbol{n}$) und eine kryptografische Hashfunktion geeinigt haben. Des Weiteren muss der Empfänger im Besitz des öffentlichen Schlüssels des Signierers sein. Möchte ein Sender/Signierer eine Nachricht signieren, führt er die folgenden Schritte durch:

1. Berechne den Hash $\boldsymbol{h} = \boldsymbol{HASH}(\boldsymbol{m})$ und speichere in $\boldsymbol{z}$ die $\boldsymbol{L_n}$ höchstwertigsten Bits von h.

2. Wähle eine zufällige Ganzzahl $\boldsymbol{k}$ aus dem Intervall $[\mathbf{1} < \boldsymbol{k} < \boldsymbol{n} - \mathbf{1}]$.
3. Berechne $\boldsymbol{r} = \boldsymbol{x_1}(\boldsymbol{mod\ n})$ mit $(\boldsymbol{x_1}, \boldsymbol{y_1}) = \boldsymbol{G^k}$.
4. Berechne $\boldsymbol{s} = \boldsymbol{k^{-1}}(\boldsymbol{z} + \boldsymbol{ra})\boldsymbol{x_1}(\boldsymbol{mod\ n})$

Die berechnete Signatur besteht aus dem Tupel $(\boldsymbol{r}, \boldsymbol{s})$. Bei dem Verfahren ist es besonders wichtig das bei jeder Signaturberechnung ein neues zufälliges $\boldsymbol{k}$ gewählt wird. Denn sollten zwei verschiedene Nachrichten mithilfe des gleichen $\boldsymbol{k}$ Wertes erzeugt werden, kann ein Angreifer diesen herausrechnen und daraus den privaten Schlüssel des Signierers ermitteln. Dazu benötigt der Angreifer lediglich zwei Signaturen$(\boldsymbol{r}, \boldsymbol{s})$ und $(\boldsymbol{r}, \boldsymbol{s}')$. Für die Verifizierung einer Signatur benötigt der Empfänger neben der Signatur $(\boldsymbol{r}, \boldsymbol{s})$ lediglich den öffentlichen Schlüssel $\boldsymbol{A}$ des Erzeugers.

1. Berechne den Hash $\boldsymbol{e} = \boldsymbol{HASH}(\boldsymbol{m})$ und speichere in $\boldsymbol{z}$ die $\boldsymbol{L_n}$ höchstwertigsten Bits von h.
2. Berechne $\boldsymbol{w} = \boldsymbol{s^{-1}}(\boldsymbol{mod\ n})$
3. Berechne $\boldsymbol{u_1} = \boldsymbol{zw}\ (\boldsymbol{mod\ n})$ und $\boldsymbol{u_2} = \boldsymbol{rw}\ (\boldsymbol{mod\ n})$
4. Berechne $(\boldsymbol{x_1}, \boldsymbol{y_1}) = \boldsymbol{u_1 G} + \boldsymbol{u_2 A}$ **+** mit $(\boldsymbol{x_1}, \boldsymbol{y_1}) = \boldsymbol{G^k}$.
5. Die Signatur ist gültig wenn $\boldsymbol{r} = \boldsymbol{x_1}\ (\boldsymbol{mod\ n})$

Bewertung des Konzepts

Zum Abschluss des Kapitels wird auch die ECC/SHA-2 Variante einer PKI mithilfe des Bewertungsschemas (siehe Abbildung 19) evaluiert. Durch die festgelegten Strukturen und Abläufe innerhalb einer PKI, unterscheiden sich die vorgestellten Umsetzungen lediglich an einigen Punkten. So müssen wie bereits beim vorherigen Konzept für eine Schlüsseltrennung mehrere Zertifikate angefordert und abgespeichert werden und die Speicherung von Schlüsseln beinhaltet auch weniger relevante Zertifikatinformationen. Des Weiteren fällt die Bewertung der funktionalen Anforderungen gleich aus, da mithilfe ähnlicher Strategien und Verfahren gearbeitet wird. So werden z. B. Daten für die Vertraulichkeit verschlüsselt und für die Integrität signiert. Im Gegensatz zum RSA kommen ECC Systemen jedoch mit erheblich kürzeren Schlüsseln aus, wodurch der benötigte Speicherplatz und Kommunikationsaufwand reduziert werden kann. Insbesondere dieser Umstand macht ECC Systeme für die Heimautomatisierung interessant. Deswegen erfüllt im Gegensatz zum RSA System das ECC System die nicht-funktionale Anforderung „low-power“ teilweise. Einziger Kritikpunkt an ECC Systemen liegt in ihrer langsameren und teureren Berechnung.

ECC/SHA-2		
Schlüsselmanagement Disziplinen:		**8**
Generierung	✓	Schlüssellänge > 128 bit
Trennung	⦿	Nutzung mehrerer Zertifikate
Speicherung	⦿	Gesteigerter Speicherbedarf durch weniger relevante Zertifikatinformationen
Austausch	✓	
Vernichtung	✓	Zertifikatssperrlisten

Funktionale Anforderungen (Sicherheitsziele)			**16**
	Vertraulichkeit	✓	Verschlüsseln von Nachrichten
	Integrität	✓	Signieren von Nachrichten (SHA-1)
	Verfügbarkeit	✓	
	Authentizität	✓	Signatur und Zertifikate
	Aktualität	⦿	
Nicht-funktionale Anforderungen			
	Low-power	⦿	
	Skalierbar	✓	
	Hohe Lebens-dauer	✗	
	Wartbarkeit	✓	

Abbildung 19: ECC/SHA-2 Bewertungsergebnis

6.5 Hybride Kryptosysteme

Als hybride Kryptografie wird die Kombination von asymmetrischer und symmetrischer Kryptografie Verfahren bezeichnet. Diese Kombination verbindet die Stärken der beiden Systeme und reduziert vorhandene Schwächen. Dabei wird zwischen zwei verschiedenen Konstruktionsmodi unterschieden. Zum einen kann der Sender einen sogenannten symmetrischen Sitzungsschlüssel, der mithilfe des asymmetrischen Verfahrens sicher zum Empfänger transportiert wird, generieren oder mithilfe des Diffie-Hellman Schlüsselaustauschs einigen sich die Kommunikationspartner gemeinsam auf einen Schlüssel. Diese Konstruktion eliminiert das Schlüsselverteilungsproblem der symmetrischen Kryptografie und ermöglicht es die schnellere symmetrische Verschlüsselung für den eigentlichen Datentransport zu nutzen. Weit verbreitet sind Kombinationen aus RSA und AES, Diffie-Hellman und AES oder ECC und AES. Nachfolgend wird eines der prominentesten hybriden Kryptografie Systeme erläutert. [18]

Transport Layer Security (TLS) – ehemals Secure Sockets Layer – ist ein hybrides Kryptografie System, das für einen Datentransport im Internet genutzt wird. Dafür unterstützt TLS eine Reihe von sogenannte „Cipher Suites“, die innerhalb des RFC2246 definiert sind und eine Kombination aus vier Algorithmen festlegt. Ein Algorithmus für den Schlüsselaustausch (z. B. RSA, DH oder ECDH), ein Algorithmus zur Authentifizierung (z. B. RSA, DSA oder ECDSA), eine Hashfunktion (z. B. SHA) und ein symmetrischer Verschlüsselungsalgorithmus (z.B. DES oder AES). Nachfolgend wird beispielhaft der Schlüsselaustausch mittels RSA erläutert.

Der Schlüsselaustausch innerhalb des TLS findet im sogenannten „Handshake Protocol“ satt. Dieses wird in vier Phasen unterteil. Phase eins dient der allgemeinen Kontaktaufnahme und der Einigung auf die genutzten Kryptoalgorithmen. Dabei werden auch zwei Zufallszahlen ($\boldsymbol{RN_c}$ & $\boldsymbol{RN_s}$) ausgetauscht, die für die spätere Berechnung des symmetrischen Schlüssels genutzt werden. Innerhalb der zweiten Phase sendet der Server sein Zertifikat an den Client, fordert das Client Zertifikat an und der Client prüft das Server Zertifikat auf Korrektheit. Phase drei beinhaltet die eigentliche Schlüsselerzeugung. Zu

Beginn sendet der Client sein Zertifikat an den Server, der daraufhin die Echtheit prüft. Anschließend generiert der Client eine Zufallszahl, das sogenannte „Pre-Master-Secret“ (**PMS**). Diese Zahl wird mithilfe des öffentlichen Schlüssels des Servers verschlüsselt und an ihn versendet. Daraufhin können jetzt sowohl der Client als auch der Server mithilfe der beiden Zufallszahlen ($\boldsymbol{RN_C}$ & $\boldsymbol{RN_S}$), dem **PMS** und der ausgewählten Hashfunktion ein sogenanntes 48 Byte „Master-Secret“ (**MS**) berechnen. Aus diesem **MS** errechnen beide Kommunikationspartner sechs verschiedene Schlüssel:

1. Ein symmetrischer Schlüssel ($\boldsymbol{K_C}$) für die verschlüsselte Kommunikation vom Client zum Server.
2. Ein symmetrischer Schlüssel ($\boldsymbol{K_S}$) für die verschlüsselte Kommunikation vom Server zum Client.
3. Ein symmetrischer MAC Schlüssel, der für die Integritätssicherung von Clientnachrichten genutzt wird.
4. Ein symmetrischer MAC Schlüssel, der für die Integritätssicherung von Servernachrichten genutzt wird.
5. Wird eine symmetrische Blockchiffre, die immer feste Blockgrößen verschlüsselt, genutzt, so werden noch zwei weitere Initialisierungsvektoren (**IV**) abgeleitet.

In der letzten Phase starten die beiden Kommunikationspartner eine symmetrisch verschlüsselte Kommunikation, die mithilfe der abgeleiteten Schlüssel abgesichert wird und beenden die laufende asymmetrisch abgesicherte Kommunikation.

Ver- und Entschlüsseln von Nachrichten

Nachdem im „Handshake Protocol“ das **MS** ausgetauscht oder ausgehandelt wurde, wird die anschließende Kommunikation mithilfe des in der „Cipher Suite“ festgelegten Verfahrens abgesichert. Nachfolgend wird eines der verbreitetsten und sichersten symmetrischen Verschlüsselungsverfahren innerhalb der TLS „Cipher Suites“ erläutert. Der Advanced Encryption Standard (AES) ist ein symmetrisches Blockchiffre Verfahren, das 128 Bit Nachrichtenblöcke verschlüsselt. Dafür können Schlüssel der Länge 128, 192 oder 256 Bit genutzt werden. AES ist als sogenanntes Substitutions-Permutations-Netzwerk (SPN) konstruiert. Ein SPN ist ein rundenbasiertes Designprinzip für Blockchiffren. Dabei wird eine feste Rundenstruktur mehrfach wiederholt. Die Rundenanzahl liegt zwischen 10 und 14 und ist dabei von der genutzten Schlüssellänge abhängig. Die nachfolgende Liste gibt einen Überblick über die durchgeführten Operationen [19]:

1. Schlüsselexpansion
2. Vorrunde
    a. AddRoundKey (Rundenschlüssel[0])
3. Verschlüsselungsrunden (r = 1 bis R-1)
    a. SubBytes()
    b. ShiftRows()
    c. MixColumns()
    d. AddRoundKey(Rundenschlüssel[r])
4. Schlussrunde
    a. SubBytes()
    b. ShiftRows()
    c. AddRoundKey(Rundenschlüssel[R])

Zu Beginn des Algorithmus wird ein 128 Bit Nachrichtenblock in eine zwei-dimensionale Matrix mit vier Zeilen und vier Spalten geschrieben, sodass jede Zelle jeweils einen Byte enthält. Anschließend werden aus dem genutzten Schlüssel sogenannte **R+1** Rundenschlüssel generiert (Schlüsselexpansion). **R** bezeichnet hierbei die durchgeführten Runden z.B. 10. Grundlegend wird innerhalb der Schlüsselexpansion eine weitere zwei-dimensionale Matrix mit 1 Byte großen Zellen erzeugt. Diese setzt sich aus dem gewählten symmetrischen Schlüssel und **R+1** Rundenschlüsseln zusammen. Die ersten Spalten

der Matrix werden mit dem gewählten Schlüssel (**MS**) gefüllt und anschließend die restlichen Spalten rekursiv erzeugt. Nach der Schlüsselexpansion und am Ende jeder Runde, wird die „AddRoundKey“ Operation durchgeführt. Hier werden der aktuelle Rundenschlüssel und zu Beginn der gewählte symmetrische Schlüssel bitweise mit dem Nachrichtenblock XOR-Verknüpft. Der erste Schritt jeder Runde besteht aus der sogenannte „SubBytes“ Operation. Dabei wird der aktuelle Nachrichtenblock monoalphabetisch mithilfe der sogenannten S-Box verschlüsselt. Dafür wird für jedes Byte in der Matrix, der jeweilige Eintrag in der S-Box gesucht. Nach dieser monoalphabetischen Verschlüsselung werden die einzelnen Zeilen der Nachrichtenblöcke noch anhand der Blocklänge linksverschoben (Zeilenmischung). Anschließend wird noch eine Spaltenmischung (MixColumns) durchgeführt, diese wird durch eine Multiplikation der einzelnen Spalten mit dem irreduziblen Polynoms ($\boldsymbol{x^8 + x^4 + x^3 + x^2 + x}$) erhalten. Die Entschlüsselung innerhalb des AES erfolgt über die gleiche SPN-Struktur wie die Verschlüsselung. Dafür wird die erläuterte SPN-Struktur in umgekehrter Reihenfolge durchgeführt. Für eine detailliertere Erläuterung des AES siehe [20, p. 346].

Signieren von Nachrichten

Das Signieren von Nachrichten zur Gewährleistung der Integrität kann im TLS Protokoll z. B. mit dem DSA (siehe 6.4.1) oder ECDSA (siehe 6.4.2) Verfahren erfolgen.

Bewertung des Konzepts

Nachfolgend wird die hybride Kryptografie mit dem Beispiel TLS evaluiert (siehe Abbildung 20). Im Gegensatz zu einem reinen asymmetrischen Kryptografieverfahren befinden sich in einem hybriden Verfahren zwei verschiedene Schlüsselarten. Zum einen ein asymmetrisches Schlüsselpaar und zum anderen eine Vielzahl von symmetrischen Schlüsseln. Diese besondere Kombination – insbesondere die einmalige Nutzung von Sitzungsschlüsseln – ermöglicht eine effiziente Trennung und Ressourcen schonende Speicherung von Schlüsseln. Da für jede neue Verbindung ein Sitzungsschlüssel ausgehandelt wird und nicht mehr verwendete Schlüssel gelöscht werden können. Somit erfüllt ein hybrides Kryptografieverfahren die Schlüsselmanagement Disziplinen vollständig. Des Weiteren ermöglicht ein hybrides Verfahren durch die Sitzungsschlüssel eine bestmögliche Aktualität und bietet ohne zusätzliche Maßnahmen einen Replay-Schutz. Die verbleibenden funktionalen Anforderungen werden wie bei den bereits evaluierten Verfahren mit Verschlüsselung und Signierung erfüllt. Somit werden auch die funktionalen Anforderungen vollständig erfüllt. Da ein hybrides Kryptosystem einen asymmetrischen Teil besitzt, stellt das „Low-Power“ Kriterium auch hier einen kleinen Stolperstein dar. Da asymmetrische Algorithmen ressourcenhungriger als symmetrische Algorithmen sind und der Schlüsselaustausch z. B. per RSA erfolgt. Im Gegensatz zu einem reinen asymmetrischen Verfahren (siehe 6.3) wird jedoch lediglich der Schlüsselaustausch asymmetrisch durchgeführt.

Hybride Verfahren (z.B. TLS)			
Schlüsselmanagement Disziplinen:			**32**
Generierung	✓	Asymmetrischer Schlüssel > 2000 (128) Bit Symmetrischer Schlüssel > 128 Bit	
Trennung	✓	Aushandlung mehrerer Sitzungsschlüssel	
Speicherung	✓	Lediglich ein asymmetrisches Schlüsselpaar benötigt. Je laufender Sitzung ein symmetrischer Schlüssel.	
Austausch	✓		
Vernichtung	✓	Sitzungsschlüssel gelten nur innerhalb einer Sitzung und besitzen eine limitierte Lebensdauer.	
Funktionale Anforderungen (Sicherheitsziele)			**32**
Vertraulichkeit	✓	Verschlüsseln von Nachrichten	
Integrität	✓	Signieren von Nachrichten	
Verfügbarkeit	✓		
Authentizität	✓	Signatur und Zertifikate	
Aktualität	✓	Je aufgebauter Verbindung ein Schlüssel	
Nicht-funktionale Anforderungen			
Low-power	⦿		
Skalierbar	✓		
Hohe Lebens-dauer	✓		
Wartbarkeit	✓		

Abbildung 20: TLS Bewertungsergebnis

6.6 Konzepte aus der Forschung

Die Sicherheit in der Heimautomatisierung ist immer wieder Thema auf IT-Security Konferenzen und in Fachzeitschriften. Insbesondere wenn neue Angriffe auf Smart Home Produkte gefunden oder Angriffe mithilfe von Automatisierungsprodukten durchgeführt werden. So wurde z. B. mithilfe eines Botnetzwerkes, das unter dem Namen „Mirai-IoT-Botnetz“ bekannt geworden ist, der DNS Dienst Dyn angegriffen, wodurch Dienste wie z. B. Twitter, Amazon und Spotify über mehrere Stunden lahmgelegt wurden [21]. Diese Angriffe und die im Allgemeinen nicht zufriedenstellende Sicherheit in der Automatisierung führen zu einer Vielzahl von Forschungsabreiten. Nachfolgend werden zwei Arbei-

ten aus der Forschung zur Verbesserung der Sicherheit in Automationsnetzwerken betrachtet und mithilfe des Bewertungsschemas evaluiert. Dabei wird wie bereits in den vorangegangenen Konzepten auch die Umsetzbarkeit innerhalb eines HAN betrachtet.

6.6.1 Physical Layer Security

Während klassische Sicherheitskonzepte – wie z. B. TLS – in den oberen Schichten (OSI-Schicht 3) der Kommunikationsprotokolle umgesetzt werden, wird in der Forschung zunehmend der „Physical Layer“ (OSI-Schicht 1) zur Steigerung der Kommunikationssicherheit genutzt. Diese sogenannten „Physical Layer Security“ (PLS) Konzepte, nutzen die unvorhersehbaren und zufälligen Eigenschaften von Wireless-Kanälen z. B. zur Generierung eines gemeinsamen symmetrischen Schlüssels. Nachfolgend werden zwei Konzepte aus dem Bereich der PLS näher betrachtet.

Robust Key Generation from Signal Envelopes in Wireless Networks

Die erste Forschungsarbeit [22] nutzt für eine Schlüsselgenerierung die physikalischen Eigenschaften einer kabellosen Kommunikation. Dabei werden das sogenannte Rayleigh-Fading und die Kanal-Reziprozität für die Generierung eines Bit-Streams genutzt. Als Rayleigh-Fading bezeichnet man eine durch die genutzte Frequenz auftretende Schwankung in der Empfangsfeldstärke einer Signalwelle. Kanal-Reziprozität besagt das sich Ursache und Wirkung innerhalb eines Kommunikationskanales vertauschen lassen, ohne das die resultierenden Verhältnisse verändern werden. Somit können beide Kommunikationspartner die gleichen Schwankungen der Feldstärke registrieren.

Zu Beginn des Verfahrens einigen sich die beiden Kommunikationspartner auf einen Schwellenwert (englisch: Threshold). Dieser Schwellenwert zeigt an, ab wann eine Abweichung der Empfangsfeldstärke nach unten als sogenannter „Deep Fade“ registriert wird. Jetzt können die beiden Kommunikationspartner auf Basis der empfangenen Kommunikationskurven (Up- und Down-Link), die einzelnen Zeitpunkte der Kurve mit dem Schwellenwert vergleichen. Dabei wird ein Wert unter dem Schwellwert als Bit 1 und über dem Schwellwert als Bit 0 registriert. Durch die Kanal-Reziprozität resultieren aus beiden Kurven beinahe identische Bitfolgen. Trotzdem können kleinere Unstimmigkeiten innerhalb der Bitfolgen auftreten. So können z. B. zum Beginn und Ende der Kurven kleinere Unstimmigkeiten auftreten, da „Deep Fades“ über mehrere Zeitslots auftreten können. Des Weiteren können die beiden Bitfolgen leicht zueinander verschoben sein oder einer der Knoten registriert einen „Deep Fade“ wo der andere Knoten keinen registriert. Innerhalb des Konzeptes werden solche Unstimmigkeiten mithilfe von „Error Correction Codes“ und Filtern herausgerechnet und abgefangen. Nachdem beide Kommunikationspartner eine Bitfolge besitzen, wird mithilfe eines sogenannten „Secure Fuzzy Information Reconciliator“ (SFIR) aus der Bitfolge ein Schlüssel generiert. Für die genaue Konstruktion des SFIR siehe [23]. Innerhalb einer durchgeführten Simulation der Autoren konnte aus einer 2000 Bit langen Folge mithilfe des SFIR ein 81 Bit Schlüssel erzeugt werden.

Bewertung des Konzepts

Im Gegensatz zu den etablierten Konzepten aus den vorangegangenen Kapiteln behandelt das Forschungskonzept lediglich die Schlüsselgenerierung und bietet somit keine Verfahren und Algorithmen zur Sicherung der Vertraulichkeit, Integrität und Authentizität. Das erläuterte Konzept ermöglicht – abhängig von der heraus gerechneten Bitfolge – die Erzeugung variable langer Schlüssel. In der durchgeführten Simulation wurde ein 81 Bit Schlüssel erzeugt, der Ressourcenschonen gespeichert werden kann. Um eine Schlüsseltrennung zu ermöglichen, müssen im Anschluss an die initiale Erzeugung jedoch weitere Schlüssel separat generiert und verschlüsselt übertragen werden. Ähnlich zu einem hybriden Verfahren. Um überhaupt eine Verschlüsselung zu ermöglichen, muss jedoch noch ein symmetrisches Verfahren hinzugefügt werden. Dies erfordert je nach benötigter Schlüssellänge für das genutzte Verfahren eine Erweiterung des generierten Schlüssels.

Neben den fehlenden Verfahren stellt Wiedereinmal die „Low-Power“ Bedingung ein Problem dar. Da insbesondere die Kommunikation größerer Pakete – wie sie für die Schlüsselgenerierung benötigt werden – die meist limitierte Energiequelle stark belasten kann.

Robust Key Generation from Signal Envelopes in Wireless Networks			
Schlüsselmanagement Disziplinen:			**16**
Generierung	✓	Schlüssellänge variable	
Trennung	⦿	Lediglich auf Umwegen möglich	
Speicherung	✓		
Austausch	✓		
Vernichtung	✓		
Funktionale Anforderungen (Sicherheitsziele)			**0**
Vertraulichkeit	⦿	Geeignete Verfahren müssen nachträglich hinzugefügt werden	
Integrität	✗		
Verfügbarkeit	✓		
Authentizität	✗		
Aktualität	✓		
Nicht-funktionale Anforderungen			
Low-power	⦿	Hoher Kommunikationsaufwand	
Skalierbar	✓		
Hohe Lebensdauer	⦿	Hoher Kommunikationsaufwand	
Wartbarkeit	✓		

Abbildung 21: Robust Key Generation from Signal Envelopes Bewertungsergebnis

6.6.2 Key Management in Wireless Sensor Networks

Das zweite Forschungskonzept [24] setzt ähnlich zum TLS auf ein hybrides Kryptographieverfahren. Dabei verwenden die Autoren eine Kombination von ECC mit 180 Bit Schlüsseln und AES mit 128 Bit Schlüsseln. Grundlegend stellen die Autoren zwei Protokolle für einen direkten und indirekten Beitritt eines neuen Knoten zu einem Automatisierungsnetzwerk vor. Diese Protokolle stammen aus dem Buch „Security for Multihop

Wireless Networks“ [25] und ermöglichen einen sicheren Netzbeitritt ohne großen Kommunikationsaufwand. Auf Basis dieser Protokolle haben die Autoren weitere Protokolle konstruiert, die einen Schlüsselwiederruf und ein Schlüsselupdate ermöglichen. Grundlage aller Protokolle der Arbeit ist eine Basisstation, die ihr eigenes Schlüsselpaar, den öffentlichen Schlüssel jedes anderen Netzwerkteilnehmers und einen symmetrischen Netzwerkschlüssel besitzt. Der Netzwerkschlüssel wird zu allen beigetretenden Knoten übertragen und eignet sich somit z. B. für Broadcast Nachrichten. Nachfolgend werden diese Protokolle näher erläutert.

Beitrittsprotokolle

Möchte ein neuer Knoten **I** dem Netzwerk beitreten, kann generell zwischen zwei Szenarien unterschieden werden. Zum einen ist der Knoten **I** in direkter Reichweite zur Basisstation **B**. Zum anderen lediglich in Reichweite eines bereits beigetretenden Knoten **R**. Liegt der Knoten in direkter Reichweite wird das in Abbildung 23 dargestellte Protokoll durchgeführt. Die Nutzung von ECC ermöglicht eine Besondere Art des Diffie-Hellman Schlüsselaustausch($\boldsymbol{K_{DH}(I,S)}$), die beinahe ohne Kommunikation eine Schlüsseleinigung ermöglicht. Dafür berechnen die Kommunikationspartner mit ihrem privaten Schlüssel und dem öffentlichen Schlüssel des Kommunikationspartners einen gemeinsamen Schlüssel. Diese Berechnung wird durch die folgenden Gleichung ermöglicht.

$$\begin{aligned}\boldsymbol{K_{DH}(I,S)} &= \boldsymbol{sk(I)} \times \boldsymbol{pk(S)} = \boldsymbol{sk(I)} \times (\boldsymbol{sk(S)} \times \boldsymbol{G}) \\ &= (\boldsymbol{sk(I)} \times \boldsymbol{G}) \times \boldsymbol{sk(S)} = \boldsymbol{pk(I)} \times \boldsymbol{sk(S)} \\ &= \boldsymbol{K_{DH}(S,I)}\end{aligned}$$

Abbildung 22: Key Management in Wireless Sensor Networks Gleichung

Zu Beginn des Protokolls verschlüsselt der Knoten **I** einen generierten Zufallswert $\boldsymbol{n}$ und seine Identität **I** mit dem errechnet Schlüssel $\boldsymbol{K_{DH}(I,S)}$. Dieses Chiffrat verschickt er an die Station **S** (siehe Abbildung 23). Die Station entschlüsselt die empfangenen Daten und verifiziert die Identität des Knoten. Bei erfolgreicher Verifizierung generiert die Station einen neuen symmetrischen Sitzungsschlüssel $\boldsymbol{K(I,S)}$. Anschließend verschlüsselt die Basisstation den empfangenen Zufallswert, seine Identität und den Sitzungsschlüssel mit dem öffentlichen Schlüssel des Knoten und verschickt dies.

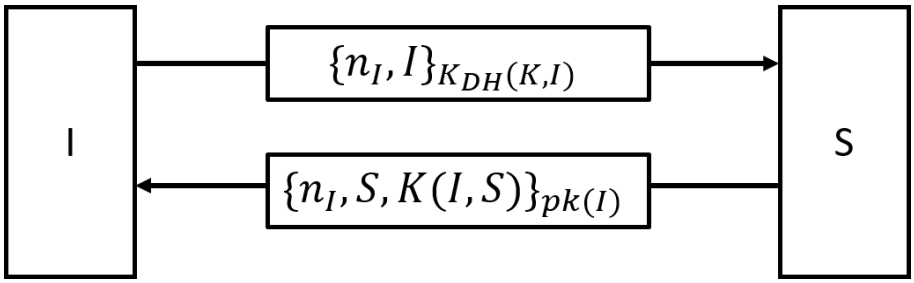

Abbildung 23: Direkter Beitritt zum Netzwerk

Liegt der Knoten **I** nicht in direkter Reichweite zur Basisstation **S**, wird das gezeigte Protokoll leicht angepasst (siehe Abbildung 24). Im Gegensatz zur direkten Übertragung der ersten Nachricht, wird diese vom Knoten **R**, der bereits im Netzwerk angemeldet ist, unverändert direkt an die Station weitertransportiert. Die Station entschlüsselt die Nachricht und verifiziert den Knoten **I**. Bei erfolgreicher Verifizierung verschlüsselt die Station den empfangenen Zufallswert **n**, seine Identität **S** und den öffentlichen Schlüssel des Knoten **I** mit dem Schlüssel $\boldsymbol{K_{DH}(S,R)}$. Das so verschlüsselte Paket sendet die Station an **R**. Daraufhin verschlüsselt **R** mithilfe des empfangenen öffentlichen Schlüssels von **I**

die Zufallszahl **n**, seine Identität **R** und einen generierten Sitzungsschlüssel $\boldsymbol{K(R,I)}$. Jetzt besitzen die beiden Knoten **I** und **R** einen gemeinsamen symmetrischen Schlüssel.

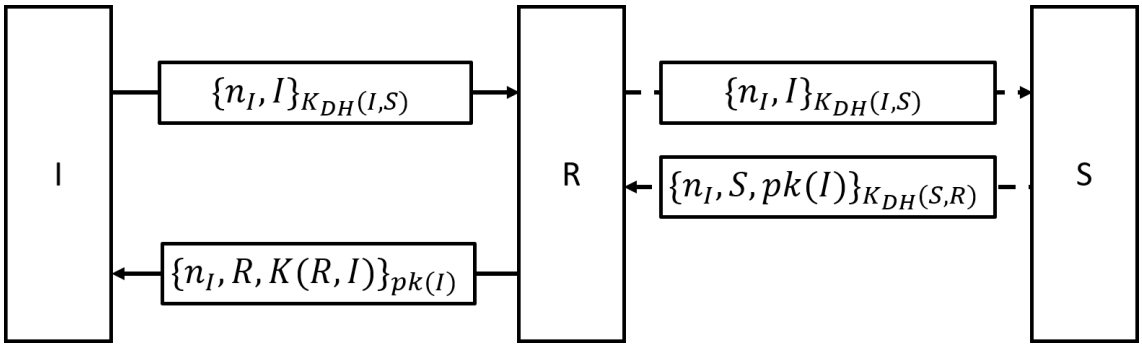

Abbildung 24: Indirekter Beitritt zum Netzwerk

Schlüsselwideruf

Das erste eigene Protokoll der Forschungsarbeit bietet ein sicheres Schlüsselwiderrufverfahren. Dafür erzeugt die Basisstation eine Liste von Knoten $\boldsymbol{M_1, \ldots, M_k}$, die in direkter Nachbarschaft eines Knoten **I** liegen und widerrufen werden sollen. Diese Liste wird zusammen mit einem Zufallswert $\boldsymbol{n}$ mit dem Schlüssel $\boldsymbol{K_{DH}(S,I)}$ verschlüsselt an den Knoten **I** übertragen. Daraufhin entschlüsselt **I** die empfangene Liste, löscht alle entsprechenden symmetrischen Schlüssel und verweigert zukünftige Kommunikationen mit den Knoten. Als Bestätigung für die Basisstation antwortet **I** mit dem empfangenen Zufallswert, verschlüsselt mit dem Schlüssel $\boldsymbol{K_{DH}(S,I)}$.

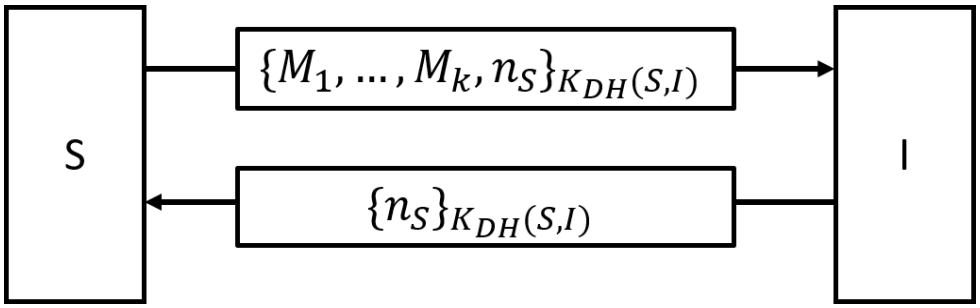

Abbildung 25: Schlüsselwideruf

Schlüsselaktualisierung

Neben einem Protokoll zum Widerrufen von Schlüsseln, bieten die Autoren für jede Art von Schlüssel, die in ihrem Konzept genutzt werden, ein eigenes Protokoll zur Aktualisierung. Somit können Schlüsselpaare, Sitzungsschlüssel und auch der Netzwerkschlüssel sicher aktualisiert werden. Nachfolgend wird beispielhaft das Protokoll zum Erneuern eines Sitzungsschlüssels erläutert. Für die anderen Protokolle wird auf das Originaldokument [24] verwiesen.

Für die Aktualisierung eines Sitzungsschlüssels generiert **I** einen neuen Schlüssel $\boldsymbol{K'(I,R)}$. Dieser wird mit dem alten Sitzungsschlüssel $\boldsymbol{K(I,R)}$ verschlüsselt und dieses Chiffrat zusätzlich nochmals mit dem öffentlichen Schlüssel des Knoten **R**. Somit muss ein Angreifer sowohl den alten Sitzungsschlüssel als auch den privaten Schlüssel $\boldsymbol{sk(R)}$ besitzen, um den neuen Sitzungsschlüssel mitlesen zu können.

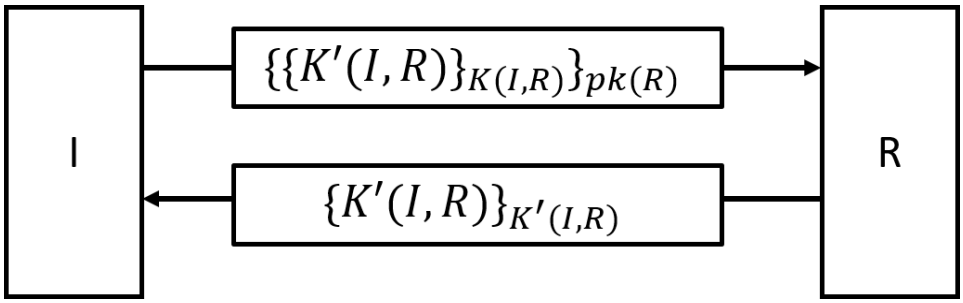

Abbildung 26: Aktualisierung eines Sitzungsschlüssels

Bewertung des Konzepts

Das vorgestellte Konzept setzt ähnlich zu TLS auf ein hybrides Kryptosystem mit einer Kombination aus ECC und AES. Damit bietet es dieselben Eigenschaften bei der Generierung, Trennung und Speicherung der Schlüssel. Im Gegensatz zu TLS ist das Konzept speziell für den Einsatz in einem Automatisierungsnetzwerk entwickelt. Somit bietet es für den Beitritt von neuen Knoten, den Austausch von Schlüssel und die Vernichtung von Schlüsseln speziell konstruierte Protokolle. Um die Sicherheit dieser Protokolle zu evaluieren, nutzten die Autoren die Software Scyther. Scyther ist eine vom Department of Computer Science der Universität von Oxford entwickelte Software zur Überprüfung von Sicherheitsprotokollen auf Konzeptionsebene. Somit erfüllt das Konzept alle Schlüsselmanagement Disziplinen. Das große Problem bei der aktuellen Version des Konzeptes ist die fehlende Integritätssicherung. Die Autoren wollen diese mit einer zukünftigen Version integrieren, diese ist zur Erstellung der Arbeit aber noch nicht erschienen. Des Weiteren wird die Authentizität nicht wie beim TLS über Zertifikate gewährleistet, sondern lediglich über Zufallswerte, die innerhalb der Chiffrate übertragen werden. Zuletzt werden noch die nicht-funktionalen Anforderungen betrachtet. Da das Konzept – wie bereits bei den Schlüsselmanagement Disziplinen erwähnt wurde – speziell für das Automatisierungsszenario entwickelt ist, erfüllt das Konzept auch die nicht-funktionalen Anforderungen.

Key Management in Wireless Sensor Networks			
Schlüsselmanagement Disziplinen:			**32**
Generierung	✓	ECC 160 Bit AES 128 Bit	
Trennung	✓	Generierung mehrer Sitzungsschlüssel möglich	
Speicherung	✓	Kurze Schlüssel ermöglichen eine effiziente Speicherung	
Austausch	✓		
Vernichtung	✓	Eigenes Schlüsselwiederufprotokoll	
Funktionale Anforderungen (Sicherheitsziele)			**0**
Vertraulichkeit	✓	ECC und AES	
Integrität	✗	Wird in der aktuellen Version der Autoren nicht gewährleistet	

Verfügbarkeit	✓	
Authentizität	✓	Wird über Zufallswerte in Verbindung mit der ECC gewährleistet
Aktualität	✓	Wird über die Schlüsselaktualisierungsprotokolle gewährleistet
Nicht-funktionale Anforderungen		
Low-power	✓	
Skalierbar	✓	
Hohe Lebensdauer	✓	
Wartbarkeit	✓	

Abbildung 27: Key Management in Wireless Sensor Networks Bewertungsergebnis

6.7 Zusammenfassung der Bewertungen

Die Anzahl von verschiedenen Schlüsselmanagementkonzepten – unabhängig von etablierten Konzepten oder Forschungskonzepten – ist beinahe unzählbar. Dabei werden die verschiedenen Konzepte meist an ein spezielles Szenario angepasst oder besitzen besondere Hardwareanforderungen. Die Evaluierung der verschiedenen Konzepte zeigt jedoch eine klare Tendenz zu den hybriden Kryptografieverfahren. Die Nutzung von asymmetrischen Kryptografieverfahren für einen sicheren Schlüsselaustausch, kombiniert mit symmetrischen Kryptografieverfahren für einen schnellen Datentransport, erzielen die besten Ergebnisse in der Bewertung. So erzielt TLS die Höchstpunktzahl von 64 und das Forschungskonzept „Key Management in Wireless Sensor Networks“ (KMiWSN), lediglich durch die fehlende Integritätssicherung eine Bewertung von 32. Ein Problem bei der Bewertung der Forschungskonzepte, liegt im Fehlen von ausreichend externen Evaluierungen oder Tests. Bei einem etablierten Konzept wie z. B. TLS wurden eine Vielzahl von Test von Kryptologen oder anderen Forschern durchgeführt, um die Sicherheit zu gewährleisten. Bei den meisten Forschungskonzepten hingegen, wurden lediglich Tests von den Entwicklern durchgeführt und davon einige auch nur auf theoretischer Protokollebene. Die nachfolgende Tabelle fasst die Bewertungsergebnisse zusammen.

Konzept:	**RSA/SHA-1**	**ECC/SHA-1**	**TLS**	**Rayleigh-Fading**	**KMiWSN**
Schlüsselmanagement Disziplinen:	8	8	32	16	32
Funktionale Anforderungen:	16	16	32	0	0
Ergebnis	**24**	**24**	**64**	**16**	**32**

Abbildung 28: Bewertungsergebnisse "State-of-the-Art"

7 Vergleich verschiedener Konzeptideen

In den vorangegangenen Kapiteln wurden verschiedene Konzepte für einen sicheren Schlüsselaustausch, sowie vollständige Schlüsselmanagement Konzepte besprochen und zuteilen bewerten. Die Erkenntnisse und Ergebnisse dieser Kapitel ermöglichen es im folgenden verschiedene Konzeptideen zu erläutern und miteinander vergleichen. Anschließend wird die Vielversprechendste der Konzeptideen ausgewählt und ausgearbeitet.

Die besonderen Eigenschaften eines Automationsnetzwerkes (siehe Kapitel 2 und 3) limitieren die Möglichkeiten eines Schlüsselaustausches stark. So besitzen die meisten Netzwerkteilnehmer keine direkte Interaktionsschnittstellen, wodurch eine einfache Passworteingabe – wie z. B. an einem Computer oder Smartphone – nicht möglich ist. Des Weiteren besitzen viele der Netzwerkteilnehmer limitierte Ressourcen, die ein aufwendiges Verfahren meist ausschließen. Zusätzlich können sich die zur Verfügung stehenden Schnittstellen zwischen den Teilnehmern unterscheiden. So kann z. B. ein Aktor, der in einer Lampe verbaut ist, lediglich einen Empfänger besitzen und keine Möglichkeit zum Versenden von Nachrichten. Andersherum kann z. B. ein Wandschalter lediglich eine Sende-Schnittstelle besitzen und keine Empfangsmöglichkeit. Diese Schnittstellen Einschränkungen gilt es bei einer Konzeptkonstruktion zu beachten. Zuletzt darf auch der Kosten- und Platzaspekt nicht vernachlässigt werden. Da Automationsnetzwerke schnell aus mehreren hundert Teilnehmern bestehen können, dürfen die einzelnen Chips nicht zu teuer werden und der Platz für einen Chip in z. B. Lampen, Schaltern und Rollläden ist begrenzt. Somit können teure und zuteilen große Erweiterungscontroller nicht genutzt werden.

Das in Kapitel 6 am besten bewertete Schlüsselmanagementkonzept – außerhalb reiner Automatisierungslösungen – ist das hybride Kryptoverfahren TLS. Innerhalb des TLS wird mithilfe von Schlüsselpaaren und Zertifikaten eine gegenseitige Authentifizierung und eine anschließende Schlüsseleinigung ermöglicht. Wie in Kapitel 6.3 erläutert wurde, wird für die Authentifizierung der Teilnehmer eine unabhängige Instanz (CA) benötigt, die eine Echtheitsprüfung der Zertifikate ermöglicht. Um ein solches System in der Heimautomatisierung umzusetzen zu können, müsste z. B. auf den Hersteller zurückgegriffen oder in jedem Heimautomatisierungsnetzwerk eine eigenen PKI umgesetzt werden. Infolge dessen würde die allgemeine Komplexität des Heimautomatisierungsnetzwerkes deutlich erhöht und die Kosten für den Hersteller steigen. Da der Hersteller im ersten Szenario einen eigenen CA-Dienst bereitstellen müsste. Ein weiterer Nachteil von TLS in der Heimautomatisierung ist das Übertragungsprotokoll TCP (Transmission Control Protocol). TCP ist ein verbindungsorientiertes Transportprotokoll, das lediglich Unicast Verbindungen zwischen Kommunikationsteilnehmern erlaubt.

Die zweite Konzeptidee basiert auf dem in Kapitel 6.3 vorgestellten Out-of-Band Schlüsselaustausch. Im Gegensatz zum TLS könnte so auf eine externe CA und die Ressourcen teure Public/Private Kryptografie verzichtet werden. Jeder Knoten kann bei der Produktion einen symmetrischen Schlüssel generieren oder erhalten, der zusätzlich Out-of-Band gesichert wird z. B. in einem QR-Code. Mithilfe dieses Codes könnte ein Kontrollsystem, das eine gesicherte Kommunikation mit dem Knoten aufbauen möchte, den benötigten Schlüssel erhalten. Vorteil einer solchen Lösung sind die reduzierte Netzkomplexität im Gegensatz zu PKI Konzepten und die physikalische Sicherheit der Schlüssel durch die Möglichkeit sie gesichert im Haus zu lagern. Jedoch wie bereits erläutert wurde, besitzen die meisten Automatisierungschips keine direkten Interaktionsschnittstellen, die das Einlesen von Passwörtern, QR-Codes oder RFID-Transpondern ermöglichen. Infolge dessen könnte z. B. lediglich ein leistungsfähiger Managementknoten die jeweiligen Schlüssel auslesen und somit eine gesicherte Unicast Kommunikation ermöglichen. Des Weiteren wäre eine Multi- und Broadcast Kommunikation auch nicht direkt möglich, da die einzelnen Knoten keine gemeinsamen Schlüssel oder Multi- bzw. Broadcastadresse teilen. Somit wäre Multi- und Broadcast Kommunikation lediglich über

den Managementknoten möglich und bei einem Ausfall dessen wäre das gesamte Netzwerk lahmgelegt.

Die dritte Konzeptidee stellt eine Kombination der beiden erläuterten Konzeptideen dar und basiert auf dem Verschlüsselungsprotokoll „Datagram Transport Layer Security“ (DTLS). DTLS wurde 2006 in der Version 1.0 standarisiert und 2012 zur Version 1.2 aktualisiert. Dabei basiert DTLS auf dem TLS Protokoll, ermöglicht jedoch eine Kommunikation über das verbindungslose Transportprotokoll UDP. Die Abläufe im DTLS Protokoll entsprechen weitgehend denen von TLS, lediglich Stellen die für das geänderte Kommunikationsprotokoll angepasst werden müssen wurden Änderungen durchgeführt. Darunter Wiederherstellung der Zuverlässigkeit des Handshakes, eine explizite Nummerierung der transportierten Pakete und ein optionaler Replay-Schutz für einzelne Pakete. Damit keine PKI Struktur aufgebaut werden muss, wird auf die „Pre-Shared-Key“ Cipher-Suites des DTLS zurückgegriffen. Diese ermöglichen eine abgesicherte Kommunikation, solange die beiden Kommunikationspartner bereits einen geteilten symmetrischen Schlüssel besitzen. Dieser Schlüssel lässt sich mit einer Out-of-Band Kommunikation sicher an einen Managementknoten übermitteln und ermöglicht eine über DTLS abgesicherte Kommunikation. Nachfolgend wird ein neues Schlüsselmanagementkonzept auf Basis der dritten Konzeptidee konstruiert und erläutert.

8 Eigenes Konzept

Auf Basis der identifizierten Eigenschaften und Sicherheitsanforderungen von Heimautomatisierungssystemen, wird im folgenden Kapitel ein eigenes Schlüsselmanagementkonzept auf Basis eines Out-of-Band Schlüsselaustausches und darauf aufbauender Nutzung des DTLS Kommunikationsprotokolls erläutert. Dabei ermöglicht die Out-of-Band Kommunikation den direkten Austausch symmetrischer Kommunikationsschlüssel und DTLS eine bestmögliche Absicherung der restlichen Netzwerkkommunikation. Als symmetrisches Verschlüsselungsverfahren wird der in den etablierten Heimautomatisierungsstandards verwendete AES im DTLS Protokoll genutzt, der selbst auf leistungsarmer Hardware und in einer reinen Softwareimplementierung gute Ver- und Entschlüsselungsgeschwindigkeiten bietet. Für die Authentifikation von Paketen und die Integritätssicherung wird der SHA-2 Algorithmus verwendet, der zum Stand der Arbeit den sichersten Hashalgorithmus darstellt. Nachfolgend werden die verschiedenen Eigenschaften und Protokolle des Konzeptes näher beschrieben.

8.1 Netzwerkstruktur

Bevor die verschiedenen Protokolle in den nachfolgenden Kapiteln beschrieben werden, wird zu Beginn die allgemeine Netzwerkstruktur – auf die das Konzept aufsetzt – erläutert. Innerhalb des Netzwerkes wird zwischen drei verschiedenen Arten von Knoten (englisch: Nodes) unterschieden. Zum Ersten nicht authentifizierte und unautorisierte Knoten (**K**), die dem Netzwerk noch nicht beigetreten sind. Diese Knoten stellen den Ausgangspunkt eines initialen Netzwerkbeitritts dar. Zum Zweiten bereits authentifizierte und autorisierte Knoten (**AK**), die dem Netzwerk bereits beigetreten und somit im Besitz des allgemeinen Netzwerkschlüssels sind. Zum Dritten die sogenannten Management Knoten (**MK**), die neue Knoten authentifizieren und autorisieren können. Des Weiteren stellen die **MK** die Schnittstelle zum Nutzer dar. Dafür benötigten **MK** eine Empfangsmöglichkeit für die genutzte Out-of-Band Kommunikation, eine Eingabemöglichkeit zum Empfang von Nutzerinteraktionen z. B. Touchscreen und eine Ausgabemöglichkeit zur Ausgabe der Ergebnisse oder aktueller Sensorwerte. Damit eignen sich z. B. Smartphones oder Tablets als MK. Die nachfolgende Abbildung (Abbildung 29) zeigt beispielhaft ein solches Netzwerk.

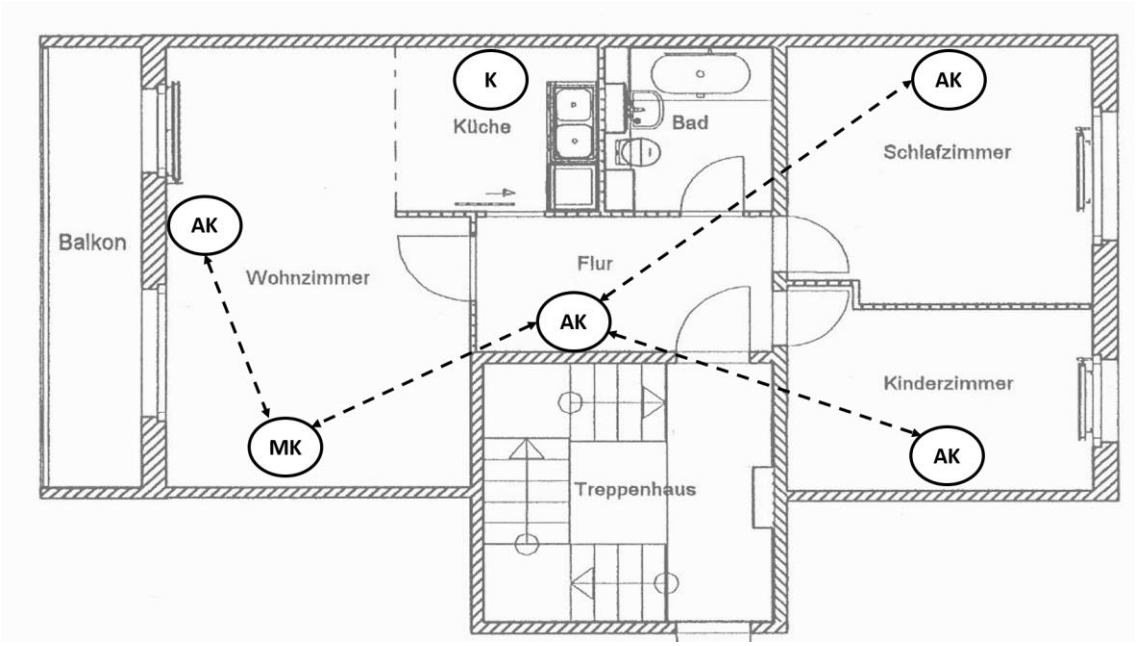

Abbildung 29: Netzwerkstruktur des Konzeptes

8.2 Schlüsselerzeugung

Im Rahmen des Konzeptes werden lediglich sogenannte Link-Schlüssel verwendet, wobei jeder Link-Schlüssel lediglich für die Absicherung eines Verbindungstyps innerhalb einer Kommunikationsgruppe genutzt wird (siehe 8.3). Mithilfe dieser symmetrischen AES Schlüssel lassen sich die in Kapitel 3 identifizierten Kommunikationsarten Unicast, Multicast und Broadcast absichern. Für eine sichere Schlüsselerzeugung benötigen die Knoten sogenannte Pseudozufallszahlen, die mithilfe von kryptografisch sicheren Zufallszahlengenerator generiert werden sollten. Kryptografisch sicherere Zufallszahlengeneratoren bieten besonders hohe Qualitätsanforderungen an die Zufälligkeit der erzeugten Zahlen und werden bereits in viele Mikrokontroller integriert. Der bereits implementierte Link-Schlüssel, der für einen Netzwerkbeitritt benötigt wird, kann somit vom Knoten selbst generiert oder bei fehlen eines kryptografischen Zufallszahlengenerator auch bei der Produktion eingespeist werden. Wobei einzig innerhalb der Produktionsumgebung eine solche Einspeisung möglich sein sollte. Weitere Link-Schlüssel sollten von einem MK generiert und gesichert an die benötigten Knoten versendet werden.

8.3 Schlüsseltrennung

Wie im vorherigen Kapitel erwähnt wurde, werden im Konzept lediglich sogenannte Link-Schlüssel definiert. Wobei ein Link-Schlüssel ausschließlich für die Absicherung einer einzelnen Verbindung (Uni-, Multi- oder Broadcast) genutzt wird. Infolge dessen ist die Gesamtzahl an Link-Schlüssel innerhalb eines Netzwerkes von der Anzahl von Knoten und den gewünschten Kommunikationsgruppen abhängig. Eine Kommunikationsgruppe definiert innerhalb des Konzeptes eine Gruppe von Knoten, die einen gemeinsamen Schlüssel teilen und somit in der Lage sind gesichert miteinander zu kommunizieren. Somit lassen sich Multi- und Broadcast Kommunikationen, jeweils mit einem eigenen Link-Schlüssel absichern. In einem Beispielszenario würde z. B. ein Netzwerk mit fünf Knoten und vier Kommunikationsgruppen neun Link-Schlüssel besitzen. Wobei die einzelnen Knoten – abgesehen vom MK – lediglich im Besitz ihres eigenen initialen Link-Schlüssels und Schlüsseln von beigetretenden Gruppe sind. Die Verwaltung der Kommunikationsgruppen wird vom MK durchgeführt. Somit erhält ein Anwender die maximale Freiheit, welche Schlüssel und Gruppen er in seinem Netzwerk möchte. Des Weiteren reduziert eine solche Schlüsseltrennungsstrategie die Schlüsselanzahl auf ein Minimum, wodurch der benötigte Speicherplatz reduziert werden kann.

8.4 Schlüsselspeicherung

Wie bereits in Kapitel 4.3 erläutert wurde, wird von einer direkten und ungesicherten Speicherung von Schlüssel in Mikrocontrollern abgeraten. Des Wegen sollten die Schlüssel in speziell für die Speicherung von Schlüsseln optimierte Speicherbereiche oder in verschlüsselten Dateien hinterlegt werden. Solche Speicherbereiche werden heutzutage in vielen Mikro- und Kryptochips angeboten und beinahe jede Programmiersprache unterstützt sogenannte „KeyStores“, die verschlüsselte Dateien für eine Schlüsselaufbewahrung erzeugen.

8.5 Schlüsselaustausch

Innerhalb des Konzeptes werden verschiedene Austauschprotokollen definiert. Zum Ersten ein Beitrittsprotokoll, dass dem MK erlaubt mithilfe des gelesenden Schlüssel einen Knoten zum Netzwerk hinzufügen. Zum Zweiten ein Gruppenbeitritts- oder Gruppenregistrierungsprotokoll, dass dem MK ermöglicht einem AK eine neue Kommunikationsgruppe mitzuteilen. Dabei muss neben der gewählten Multi- oder Broadcast Adresse, der verwendete Schlüssel ausgetauscht werden. Des Weiteren wird ein Aktualisierungsprotokoll definiert, das eine sichere Aktualisierung einzelner Link-Schlüssel ermöglicht.

8.5.1 Netzwerkbeitritt

Die Wahl einer Out-of-Band Kommunikation für den initialen Schlüsselaustausch, reduziert die durchgeführten und benötigten Operationen innerhalb eines Netzwerkbeitritts. Somit entfallen typische Nachrichten wie z. B. der Austausch von Berechnungsparametern wie beim Diffie-Hellman oder öffentliche Schlüssel wie beim RSA. Des Weiteren müssen neue Knoten nicht in spezielle Beitrittsmodi geschaltet werden, da der MK ohne direkte Beteiligung des Knoten den Schlüssel erhält. Möchte der MK einen neuen Knoten in das Netzwerk einbinden, benötigt er lediglich den mitgelieferten Schlüssel, den er über die Out-of-Band Kommunikation z. B. einen QR-Code erhält und einen Zuweisungsparameter (Knoten-ID). Diese ID sollte über das gesamte Netzwerk eindeutig sein und wird in folgenden Kommunikationen zur Identifizierung des Knoten und des benötigten Schlüssels verwendet. Dabei wird die genaue Konstruktion der ID innerhalb des Konzeptes nicht festgelegt, da sich je nach Implementierung evtl. andere Identifikatoren nutzen lassen oder weitere Parameter gewünscht sind z. B. ein Knoten-Typ. So kann z. B. in einem IP-basierten Netzwerk eine IP-Adresse oder eine Domain als Identifikator genutzt werden, da diese einzigartig über das Netzwerk vergeben werden können. Andererseits kann natürlich auch ein einfacher Zahlenwert, der über alle Knoten einzigartig ist z. B. eine Chip ID, als Identifikator hinterlegt werden. Im Fall einer Adressen unabhängigen Chip-ID muss zusätzlich ein Discovery Verfahren – wie z. B. DNS-SD – genutzt werden, womit der **MK** die entsprechende Knoten-Adresse erhält.

Um eine ungefähre größen Vorstellung einer solchen Kombination zu geben, wird in Abbildung 30 ein QR-Code inklusive ID und 256 Bit Schlüssel dargestellt.

coapNode412512|9D65FF9F147C6F46FEA7C9AD24518

Abbildung 30: Beispiel QR-Code

8.5.2 Gruppenschlüsselaustausch

Für eine erfolgreiche und gesicherte Kommunikation innerhalb einer Multi- oder Broadcastgruppe, benötigen die einzelnen Gruppenmitglieder eine ausgewählte Multi- oder Broadcastadresse inklusive Port (**A**), ein innerhalb der Gruppe bekanntes Passwort (**LK**) und eine Sender-ID (**SID**). Die Sender-ID dient in der Gruppe dazu, empfangene Pakete einem Gruppenmitglied zuordnen zu können. Dabei gilt es zu beachten – wie bereits in Kapitel 7 erläutert wurde – das nicht zwangsläufig jeder Knoten eine Sendeschnittstelle besitzt. Somit benötigen nicht zwangsläufig alle Gruppenmitglieder eine Sender-ID. Möchte ein Nutzer über den **MK** eine neue Kommunikationsgruppe z. B. Beleuchtung konfigurieren, generiert der **MK** einen AES Gruppenschlüssel und wählt eine freie Kommunikationsadresse (Multi- oder Broadcast) aus. Diese Parameter versendet der **MK** über Unicast Verbindungen an die gewünschten Gruppenmitglieder inklusive einer Sender-ID, falls die Knoten eine Sendeschnittstelle besitzen. Dabei nutzt er die

beim Netzwerkbeitritt erhaltenen Schlüssel zur Absicherung der einzelnen Verbindungen. Grundlegend benötigen die Gruppenmitglieder lediglich diese beiden Parameter für eine Kommunikation innerhalb der Gruppe. Jedoch sind abhängig vom genutzten Kommunikationsprotokoll oder den jeweiligen Anwenderpräferenzen noch weitere Parameter denkbar oder nötig. Das Konzept schränkt dabei nicht die Implementierungsmöglichkeiten ein, solange die versendeten Parameter wie beschrieben verschlüsselt kommuniziert werden. Die **AK** mit einer Sendeschnittstelle bestätigen den Erhalt der neuen Gruppe mit einer Antwortnachricht (**ACK**). Abbildung 31 zeigt die durchgeführte Kommunikation für einen Gruppenschlüsselaustausch inklusive einer Bestätigung, die lediglich die erhaltene Sender-ID enthält.

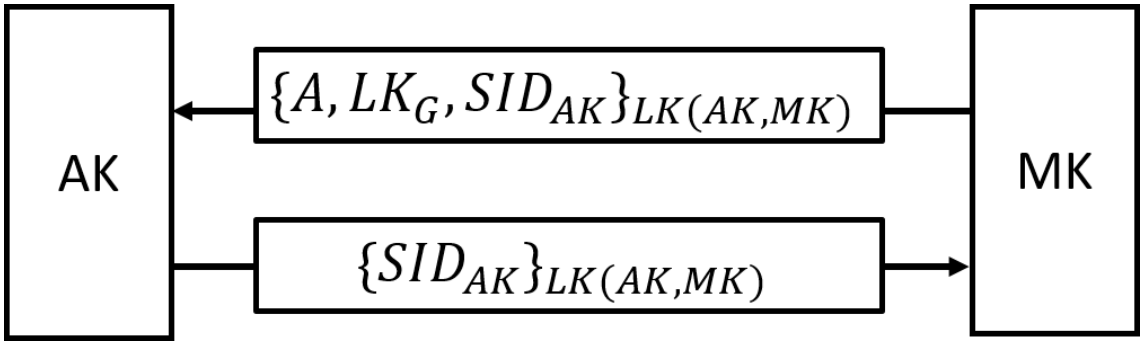

Abbildung 31: Gruppenschlüsselaustausch

8.5.3 Schlüsselaktualisierung

Um einen Langzeitschutz des Systems zu gewährleisten und mögliche Angriffe – wie z. B. „Replay Angriffe" – zu unterbinden oder deren Auswirkung zu verringern, wird ein Aktualisierungsprotokoll für Link-Schlüssel definiert. Somit lassen sich die Schlüssel in regelmäßigen Abständen oder nach einem Angriff abgesichert aktualisieren. Mithilfe des Aktualisierungsprotokolls in Abbildung 32 lässt sich jeder Link-Schlüssel im Netzwerk aktualisieren. Dabei wird nicht zwischen einem Uni- ,Multi- oder Broadcast Schlüssel unterschieden. Dies ermöglicht eine sichere Schlüsselaktualisierung auch bei einer Kompromittierung eines oder mehrerer Knoten im Netzwerk, da neu generierte Schlüssel nicht einfach per Multi- oder Broadcast an jeden Teilnehmer übertragen werden. Sondern mithilfe der jeweiligen Unicast Link-Schlüssel abgesichert vom **MK** direkt an die entsprechenden Knoten übertragen werden. So kann am **MK** entschieden werden, welche Knoten den neuen Schlüssel erhalten sollen und welche nicht z. B. kompromittierte Knoten. Infolge dessen eignet sich eine ähnliche Konstruktion wie sie bereits beim Gruppenschlüsselaustausch (siehe Abbildung 31) genutzt wurde. Im Gegensatz zum Gruppenschlüsselaustauschprotokoll muss bei einer Schlüsselaktualisierung jedoch keine neue Sender-ID vergeben werden. Die bestehenden Sender-IDs können auch bei einem Schlüsselwechsel ohne Störungen weiter verwendet werden. Sollten dem **MK** im Gruppenschlüsselaustausch die IDs ausgehen, kann er unter den gespeicherten Sender-IDs nach freien Stellen suchen. Wobei selbst 2 Byte Sender-ID Zahlenraum bereits über 60.000 verschiedene IDs ermöglichen würden, was ein fehlen von freien IDs unwahrscheinlich macht. Ein Sonderfall für das Protokoll stellt die Aktualisierung eines Unicast Link-Schlüssels zwischen dem **MK** und einem **AK** dar, da dieser Schlüssel für die gesicherte Kommunikation der Aktualisierungsnachricht verwendet wird. Um mögliche Kommunikationsfehler in der Kommunikation zu vermeiden, sendet der jeweilige Empfängerknoten die Bestätigungsnachricht, bevor er die neuen Referenzen hinterlegt. Des Weiteren muss der MK auf Basis der Knoteninformationen und dem generierten Schlüssel die genutzte Out-of-Band Quelle aktualisieren. Dies bedeutet das z. B. ein neuer QR-Code generiert oder die entsprechenden RFID-Transponder Daten aktualisiert werden müssen.

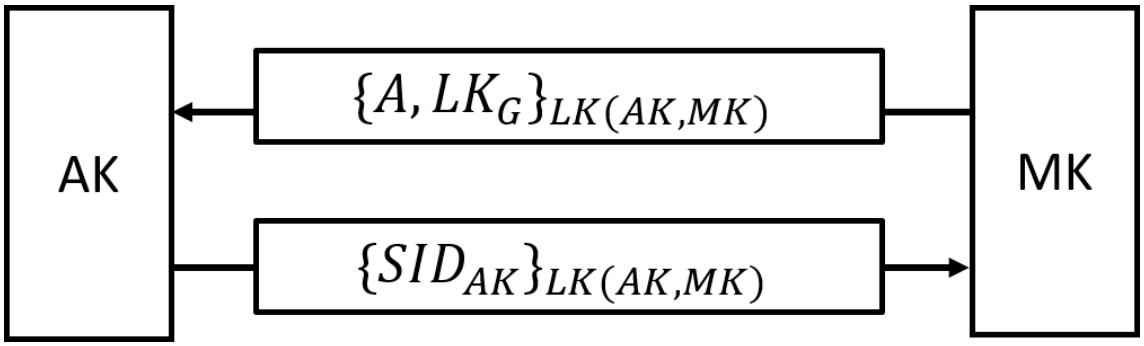

Abbildung 32: Schlüsselaktualisierung

8.6 Schlüsselvernichtung

Im Gegensatz zu den genannten PKI Konzepten, wird innerhalb des entwickelten Konzeptes kein direktes Protokoll für eine Schlüsselvernichtung benötigt. Sollte das Netzwerk angegriffen werden oder verschiedene Knoten nach einem Angriff korrumpiert sein, lässt sich die Sicherheit des Netzwerkes durch eine Schlüsselaktualisierung wieder herstellen. Da die Aktualisierung über die jeweiligen Unicast Link-Schlüssel durchgeführt wird, lassen sich betroffene oder vermutlich betroffene Knoten von der Aktualisierung ausschließen. Somit erhalten die Angreifer selbst nicht die neuen Schlüssel des Netzwerkes.

9 Proof-of-Concept

Wie das in Kapitel 8 entwickelte Konzept zeigt, vereinfacht eine Out-of-Band Kommunikation den initialen Schlüsselaustausch drastisch. Dies lässt sich durch das Fehlen von aufwendigen mathematischen Operationen wie z. B. bei ECC, angepassten Kommunikationsmethoden z. B. Rayleigh-Fading, zusätzlichen Kommunikationspartnern z. B. eine CA und spezieller Hardware erklären. Somit benötigen die einzelnen Aktoren oder Sensoren im Netzwerk kaum Rechenleistung oder spezielle Kommunikationschips, was ihre Produktionskosten und die eigentliche Chipgröße minimal hält. Des Weiteren benötigen die erläuterten Protokolle kaum Kommunikationsnachrichten, wodurch die limitierten Energieressourcen der Knoten minimal belastet werden.

Zur Überprüfung des entwickelten Konzeptes wird im nachfolgenden Kapitel eine Implementierung auf Basis eines Android Smartphones als MK und mehreren Raspberry-Pi als Knoten bzw. AK gezeigt. Hierbei setzt die Arbeit auf ein bestehendes Projekt von Jannis Konrad einem Mitarbeiter der Hochschule Bonn-Rhein-Sieg auf. Innerhalb des Projektes nutzte Herr Konrad die öffentlich erhältliche C-Implementierung des DTLS Protokolles „tinydtls“ [26] und die ebenfalls öffentlich erhältliche Python-Implementierung des „Constrained Application Protocol“(CoAP) namens „aiocoap“ [27], zur Umsetzung einer abgesicherten Unicast Kommunikation zwischen verschiedenen Raspberry-Pi Knoten. Zusätzlich dazu erweiterte er die bestehenden Pakete (tinydtls und aiocoap), sodass die Knoten neben direkter Unicast Kommunikation auch Multi- oder Broadcast Nachrichten über das DTLS Protokoll versenden und empfangen können. Diese Erweiterung, die lediglich als Draft [28] veröffentlicht wurde, ist aktuell noch nicht im DTLS Standard vorhanden und somit nicht Bestandteil anderer DTLS Softwarepakete für Programmiersprachen wie z. B. Java. Die jeweiligen Raspberry-Pi Knoten nutzen hierbei das DTLS Protokoll auf Basis eines „Pre-Shared Key“ (PSK) zur Etablierung einer gesicherten Kommunikation. Hierbei müssen in der aktuellen Umsetzung des Projektes die Schlüssel über einen Computer direkt in die einzelnen Knoten eingespeichert werden. Mithilfe des in dieser Arbeit entwickelten Konzeptes wird diese Einschränkung aufgehoben. Da eine abgesicherte Multi- und Broadcast Kommunikation vom MK ausgehend für das entwickelte Konzept nicht benötigt wird und die Umsetzung dieser außerhalb des Rahmen der Arbeit liegt, wird in den nachfolgenden Kapiteln nicht näher darauf eingegangen. Dabei gilt es noch anzumerken, dass die Nachfrage nach einer Multicast DTLS Kommunikation innerhalb der Entwicklergruppen wächst und somit in naher Zukunft mit einer direkten Implementierung zu rechnen ist.

Bevor auf die Umsetzung des Konzeptes eingegangen wird, werden zu Beginn kurz die verwendeten Protokolle CoAP und DTLS erläutert. Da das Konzept zur Steigerung der Implementierungsfreiheit bewusst, möglichst unabhängig von einen Kommunikationsprotokoll entwickelt wurde. Anschließend werden die Testumgebung bzw. das exemplarische Smart Home System und die einzelnen Knoten näher beschrieben. Darauf aufbauend wird die Implementierung der einzelnen Konzept-Protokolle erläutert und in Kapitel 10 das Konzept und die Implementierung Evaluiert und ein Ausblick in zukünftige Erweiterungen gegeben.

9.1 Kommunikations- und Verschlüsselungsprotokoll

Innerhalb des exemplarischen Smart Home Systems kommunizieren die einzelnen Knoten über, das speziell für die Nutzung innerhalb ressourcenarmer Netzwerke entwickelte Protokoll, CoAP. Zur Absicherung dieser Kommunikation wird DTLS mit einem PSK genutzt. Beide Protokolle basieren dabei jeweils auf einem stark verbreiteten Internetprotokoll. CoAP nutzt Techniken und Methoden von HTTP und DTLS ist beinahe identisch mit dem in Kapitel 6.5 erläuterten TLS Protokolls. Die nachfolgenden Kapitel beschreiben die wichtigsten Eigenschaften und Abläufe der Protokolle für den Rahmen der Arbeit.

9.1.1 Constrained Application Protocol (CoAP)

CoAP wurde von von der „Internet Engineering Task Force“ (IETF) als rein binäres Web-Transfer-Protokoll für das IoT entwickelt. Dabei basiert es auf „Representational State Transfer“ (REST), der verbreitesten Architektur für Webdienste, wobei ein Server durch einen Anwendungsprozess verschiedene Ressourcen zur Verfügung stellen kann. Diese Ressourcen können dann über einen „Uniform Resource Identifier“ (URI) identifiziert und angesprochen werden. CoAP wurde bewusst möglichst nahe am weitverbreiteten Internetprotokoll HTTP entwickelt, um eine Interoperabilität der beiden Protokolle zu vereinfachen. Dabei wird CoAP genau wie HTTP über zwei Schichten organisiert, die Anwendungs- und die Transportschicht. Innerhalb der Anwendungsschicht werden mithilfe von REST Ressourcen implementiert und Identifier festgelegt. Eine beispielhafte CoAP URI ist „**coap://sensor/room/temperature**“. Dabei besteht der Identifier aus drei Teilen:

- Dem Protokoll bestimmenden Präfix „**coap://**“
- Der Adressierung des Empfängers „**sensor**“
- Dem genauen Ressourcen Pfad „**room/temperature**“

Zum Ansprechen der Ressourcen bietet CoAP genau wie HTTP die Kommandos GET zum Auslesen von Daten wie z. B. Sensoren und PUT, POST und DELETE für das Verändern von Daten an. Der größte Unterschied zwischen HTTP und CoAP liegt im Transportprotokoll. HTTP nutzt TCP, das einen großen Overhead für deterministische Transaktionen wie z. B. ein wiederholtes Auslesen eines Sensors, besitzt. CoAP hingegen nutzt UDP, das einen wesentlich kleineren Overhead besitzt und sich somit besser für das IoT oder andere ressourcenbeschränkte Netzwerke eignet. Des Weiteren unterscheidet CoAP in der Transportschicht zwischen vier verschiedenen Nachrichtentypen:

- „Confirmables“ sind zu bestätigende Nachrichten.
- „Non-Confirmables“ sind nicht zu bestätigende Nachrichten.
- „Acknowledgement“ ist die Bestätigung einer Nachricht.
- „Reset“ zeigt das eine (Confirmables) Nachricht empfangen wurde, aber nicht weiter verarbeitet wird.

9.1.2 Datagram Transport Layer Security (DTLS)

Wie bereits in Kapitel 0 erläutert wurde ist DTLS eine UDP Umsetzung des weit verbreiteten TLS Protokoll. Dabei bietet es einen Schutz gegen das Abhören, Manipulieren oder Verfälschen von Client-Server Kommunikation. Innerhalb des Protokoll-Stack stellt sich DTLS zwischen die Anwendungs- und Transportschicht zur Absicherung der Kommunikation (siehe Abbildung 33**Fehler! Verweisquelle konnte nicht gefunden werden.**). Da innerhalb der Arbeit DTLS mit einem PSK genutzt wird, werden innerhalb der gezeigten Kommunikationsabläufe lediglich die PSK relevanten Nachrichten angegeben. So werden im Handshake in Abbildung 37 lediglich die PSK relevanten Nachrichten beschrieben.

Application (CoAP)
Security (DTLS)
Transport (UDP)
Network (IPv4/IPv6)
PHY/MAC (IEEE 802.15.4)

Abbildung 33: Protokoll Stack CoAP-DTLS

Nachfolgend werden die wichtigsten Änderungen des DTLS am TLS Protokoll, der DTLS Record Layer und anschließend der DTLS Handshake näher beschrieben.

1. **Explizite Datensätze:** TLS wird auf einem langen durchgehenden Datenstrom durchgeführt, der je nach vorhandener Paketgröße aufgeteilt wird. Beim DTLS Protokoll werden die einzelnen Datensätze (englisch: „Records“) des Protokolls direkt auf einzelne UDP-Datagramme angepasst.
2. **Explizite Sequenz Nummern:** TLS nutzt einen MAC zur Integritätssicherung seiner Pakete. Dabei wird neben der Nachricht auch eine implizite Sequenznummer als Eingabewert für den MAC verwendet. Diese Nummer wird fortlaufend mit jedem neuen Paket erhöht und ermöglicht so die Verifizierung das keine Pakete verloren oder dupliziert wurden. Im DTLS Protokoll, wo ein Paketverlust nicht ausgeschlossen werden kann, wird die Sequenznummer explizit für jeden Record festgelegt. Des Weiteren wird bei DTLS die Sequenznummer in 16 Bit „epoch“ und 48 Bit „subsequenznummer“ aufgeteilt, wodurch die „Cipher Suite“ Aushandlung vereinfacht wird.
3. **Toleranz für Abweichungen:** Da das TLS Protokoll auf einer verbindungsorientierten Kommunikation beruht, werden Abweichungen z. B. die Reihenfolge in der einzelne Datagramme empfangen werden, abgelehnt. Beim DTLS Protokoll hingegen müssen solche Abweichungen, die bei einer UDP Kommunikation nicht ausgeschlossen werden können, toleriert werden und geeignete Maßnahmen getroffen werden. So werden spezielle „window“ Mechanismen genutzt um zu früh erhaltene oder leicht verspätete Pakete richtig zu interpretieren.
4. **Zustandslose Verschlüsselung:** Da innerhalb einer UDP Kommunikation einzelne Datagramme verloren gehen können, dürfen die Algorithmen keinen Zustand verwenden der mit jedem neuen Datagramm geändert wird. Somit entfallen Verfahren wie der RC4 Stromverschlüsselungsalgorithmus.

Eine genaue Auflistung aller Änderungen und genauere Informationen zu den erläuterten Änderungen sind in Section 4 des DTLS RFC [29] zu finden. Der Record Layer dient in Protokollen zur Kapselung der höheren Protokolle wie z. B. dem Handshake Protokoll. Dabei werden im Record Layer die Daten falls benötigt fragmentiert und beim Empfänger wieder zusammengesetzt. Abbildung 34 zeigt den Aufbau des DTLS Record Layer.

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Content type	**Version**		**Epoch**		**Subsequence number**		
Subsequence number			**Length**				

Abbildung 34: DTLS Record Layer

UDP Kommunikation und darauf basierende Sicherheitsprotokolle sind äußerst anfällig für „Denial-of-Service“ Angriffe wie z. B. die Initiierung einer Vielzahl von Handshake Anfragen, die auf der Serverseite Ressourcen blockieren. Um diese Angriffe abzuwehren, nutzt DTLS eine sogenannte zustandslose Cookie Technik (siehe Abbildung 36). Dabei erzeugt der Server nach dem Erhalt einer „ClientHello“ Nachricht einen Cookie und sendet diesen zusammen mit einem „HelloVerifyRequest“ zurück an den Client. Der Cookie sollte dabei ohne zusätzliche Statusinformationen pro-Client generiert werden, wodurch auf der Serverseite Ressourcen gespart werden können. Eine Möglichkeit wäre die folgende Gleichung:

$$\boldsymbol{Cookie} = \boldsymbol{HMAC}(\boldsymbol{Secret}, \boldsymbol{Client} - \boldsymbol{IP}, \boldsymbol{Client} - \boldsymbol{Parameters})$$

Abbildung 35: DTLS Cookie Zusammensetzung

Nach dem Erhalt des Cookies muss der Client eine erneute „Hello“ Nachricht an den Server inklusive Cookie senden. Lediglich bei einer erfolgreichen Überprüfung des Cookies führt der Server mit dem Handshake fort. Dieses Verfahren reduziert DoS-Angriffe, da ein Angreifer in der Lage sein muss die „VerifyRequest“ Nachricht zu empfangen und korrekt zu verarbeiten.

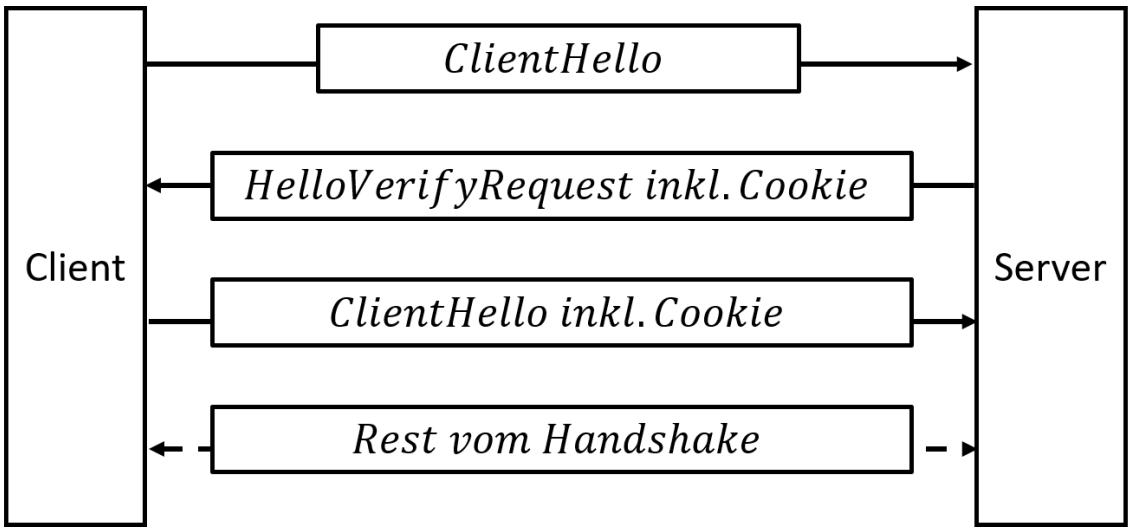

Abbildung 36: DTLS DoS Gegenmaßnahmen

DTLS Handshake Nachrichten werden in Gruppen sogenannte „Flights“ organisiert (siehe Abbildung 37). Dabei besitzt ein normaler störungsfreier Handshake sechs Flights. Die in Abbildung 36 gezeigte Kommunikation zeigt den Beginn des Handshakes und die ersten drei Flights. Im Anschluss an diese erzeugt der Server den vierten Flight, der aus den Nachrichten „ServerHello“, „ServerKeyExchange“ und „ServerHelloDone“ besteht. Dabei zeigen „ServerHello“ und „ServerHelloDone“ den Anfang und das Ende des Flight. Innerhalb der „ServerKeyExchange“ Nachricht teilt der Server dem Client mit welchen PSK er für die Kommunikationssitzung nutzen möchte. Dabei wird eine Key-ID und unterstützte Kryptoverfahren übertragen. Die versendete Key-ID muss auf beiden Seiten für denselben Schlüssel genutzt werden. Nachdem der Client den gesamten Flight erhalten und verarbeitet hat, generiert er den fünften Flight. Dieser besteht aus den Nachrichten „ClientKeyExchange“, „ChangeCipherSpec“ und „Finishend“. „ClientKeyExchange“ dient dabei als Bestätigung der gewählten Key-ID und der Auswahl unterstützter Kryptoverfahren. „ChangeCipherSpec“ teilt dem Server mit das der Client zukünftige Nachrichten über das ausgehandelte Kryptoverfahren versendet und „Finished“ zeigt das Ende des fünften Flights und Handshakes an. Der sechste und letzte Flight besteht lediglich aus den Nachrichten „ChangeCipherSpec“ und „Finished“, die dieselben Funktionen erfüllen wie beim Client.

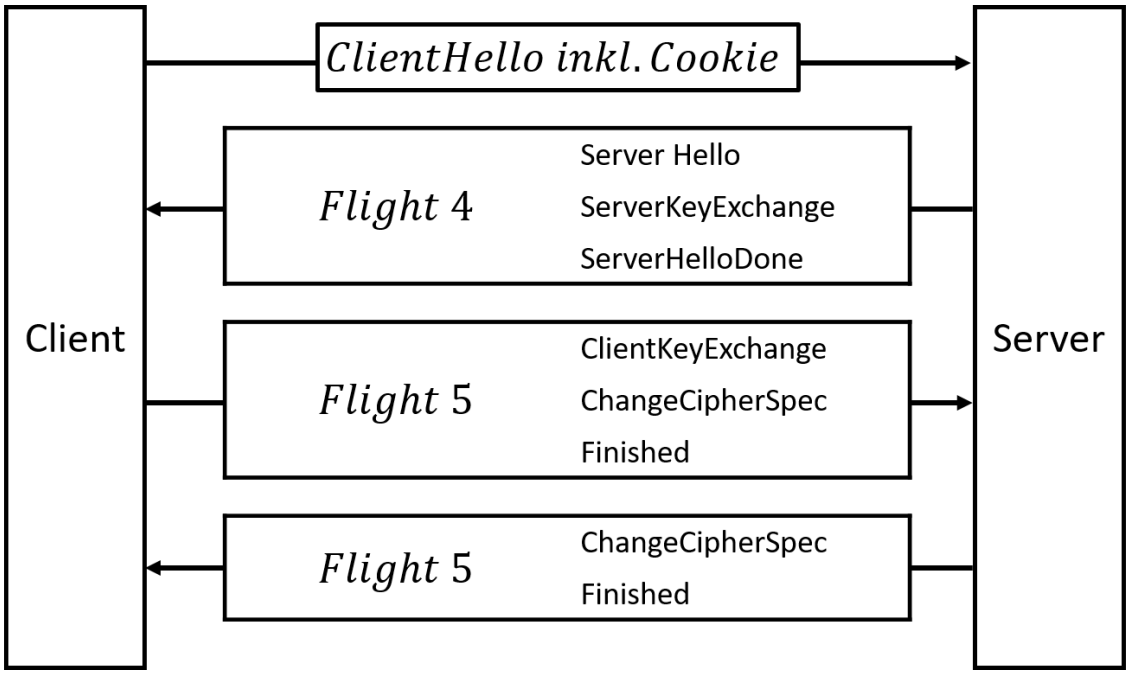

Abbildung 37: DTLS Handshake Protokoll

9.2 Exemplarisches Smart Home System (Testumgebung)

Das exemplarische Smart Home System, indem die nachfolgende Implementierung getestet wurde, besteht aus einem MK und vier AK. Somit können alle definierten Konzeptprotokolle getestet werden. Als MK wird ein Android Smartphone genutzt, wodurch ein realitätsnahes Szenario geschaffen wird. Da sich heutzutage viele Smart Home Systeme über Apps von Smartphones oder Tablets konfigurieren und steuern lassen. Die App wird mit der minimalen SDK Version 19 von Google entwickelt und ist infolge dessen auf jedem Android Smartphone oder Tablet ab Version 4.4 KitKat lauffähig. Zum Zeitpunkt der Erzeugung der Arbeit sind das rund 90% aller Android Geräte [30]. Als Knoten innerhalb des Netzwerkes werden Raspberry-Pis der ersten Generation genutzt, wovon drei mit Lampen verbunden wurden und so als Lampensteuerung genutzt werden können. Der letzte Raspberry-Pi kann als Schalter genutzt werden, um die erzeugten Kommunikationsgruppen zu testen. Auf den einzelnen Knoten läuft die in Kapitel 9 erläuterte Software aus dem Projekt von Herr Konrad. Die einzelnen Raspberry-Pis sind mit einem ARM Prozessor mit einem Kern und 700 MHZ, 512 MB RAM und lediglich eine Ethernet Schnittstelle ausgestattet. Somit werden die einzelnen Knoten nicht über W-LAN angesprochen, das innerhalb eines Smart Home Systems wünschenswert ist. Dies verändert jedoch nicht die Aussagekraft der Tests, da innerhalb des Konzeptes jede Kommunikation über einen Router geführt wird und die genaue Schnittstelle zwischen dem Router und den Knoten unabhängig vom Konzept ist.

9.3 Schlüsselerzeugung im Management Knoten (Android)

Im Rahmen des Schlüsselaustausches und der Erzeugung einer Gruppe müssen neue sichere symmetrische AES Schlüssel erzeugt werden. Dies wird innerhalb einer Android Applikation über einen sogenannte „KeyGenerator“ ermöglicht. Dabei bietet Android verschiedene Instanzen des Generators für eine Vielzahl von Schlüsseltypen ECC, AES, RSA usw. an. In der Implementierung wird die Open Source Bibliothek „BouncyCastle“ (BC) für alle schlüsselrelevanten Operationen – wie z. B. Generierung oder Speicherung – genutzt. Abbildung 38 zeigt wie ein solcher Generator für 128 Bit AES Schlüssels initiiert, sowie ein neuer Schlüssel generiert wird. Dabei wird die gezeigte Funktion sowohl bei der Schlüsselaktualisierung als auch der Erzeugung einer neuen Kommunikationsgruppe genutzt.

```
/**
 * Generate a Symmetric Key
 */
private Key genSymmetricKey() {
    Key k = null;
    try {
        KeyGenerator kgen = KeyGenerator.getInstance("AES", "BC");
        kgen.init(128);
        k = kgen.generateKey();
    } catch (NoSuchAlgorithmException | NoSuchProviderException e) {
        e.printStackTrace();
    }
    return k;
}
```

Abbildung 38: Generierung eines AES Schlüssels (Android)

9.4 Schlüsselspeicherung und –abruf mit Android KeyStores

Einer der Hauptkritikpunkte an den etablierten Smart Home Systeme liegt in der mangelnden oder fehlenden sicheren Schlüsselspeicherung. Um innerhalb der Konzeptimplementierung die Schlüssel sicher im Smartphone, das meist eine Vielzahl weiterer Apps beinhaltet, zu sichern, wird ein sogenannter „KeyStore“ genutzt. Dieser ermöglicht es eine spezielle Datei im Smartphone zu erzeugen, die mit einem Passwort verschlüsselt abgespeichert wird. Bei einem BouncyCastle „KeyStore“ sind dies .bks Dateien. Dabei ist ein „KeyStore“ mit einer Datenband vergleichbar, die einen Zugriff auf Schlüssel mit dem richtigen Alias und Datenbank Passwort erlaubt. Abbildung 39 zeigt die Initiierung eines solchen „KeyStores“ und anschließende die Speicherung eines AES Schlüssels unter eine gegebenen Knoten-ID mithilfe des „KeyStore“ Passwortes (**setKeyEntry**).

```
// Key Store Init
File store = new File(getFilesDir(), "keyStore.bks");
KeyStore keyStore = KeyStore.getInstance("BKS", "BC");
keyStore.load(null, null);
keyStore.store(new FileOutputStream(store),
        getString(R.string.key_Store_Pass).toCharArray());
// Key Store Init

keyStore.load(new FileInputStream(store),
        getString(R.string.key_Store_Pass).toCharArray());
keyStore.setKeyEntry(node.getNodeKID(), mAesKey,
        getString(R.string.key_Store_Pass).toCharArray(), null);
keyStore.store(new FileOutputStream(store),
        getString(R.string.key_Store_Pass).toCharArray());
```

Abbildung 39: Schlüsselspeicherung über einen KeyStore

Um zu einem späteren Zeitpunkt einen Schlüssel aus dem „KeyStore“ abzurufen, wird der in Abbildung 40 gezeigte Code genutzt. Dazu muss zu Beginn der gewünschte „KeyStore“ aus der Datei mit dem richtigen Passwort geladen werden (**load**). Anschließend können mit der Methode **getKey(Alias,KeyStorePassword)** Schlüssel vom KeyStore mithilfe des vorher festgelegten Alias abgerufen werden.

```
// Key Store Init
File store = new File(getFilesDir(), "keyStore.bks");
KeyStore keyStore = KeyStore.getInstance("BKS", "BC");
// Key Store Init

keyStore.load(new FileInputStream(store),
        getString(R.string.key_Store_Pass).toCharArray());

SecretKeySpec key = (SecretKeySpec) keyStore.getKey(nodeInformation[4],
        getString(R.string.key_Store_Pass).toCharArray());
```

Abbildung 40: Schlüsselzugriff über einen KeyStore

9.5 Implementierung der Konzeptprotokolle

Nachdem die sichere Erzeugung, Speicherung und der Abruf von Schlüsseln gezeigt wurde, wird in dem nachfolgenden Kapitel die Implementierung der in Kapitel 8 konstruierten Protokolle beschrieben. Dazu wird die Open Source Implementierung des CoAP und DTLS Protokolls Californium bzw. Scandium genutzt [31]. Californium ist eine in Java entwickelte Implementierung des CoAP Protokolles (RFC7252), die mit Scandium eine Erweiterung um das DTLS Protokoll (RFC 6347) besitzt. Dabei wird die Entwicklung von Californium und Scandium von Bosch und Sierra-Wireless unterstützt. Des Weiteren wird „ZXing“ eine Open-Source 1D/2D Barcode und Image Bibliothek genutzt, die eine Verarbeitung von QR-Codes ermöglicht. Somit lassen sich nicht nur QR-Codes einlesen, sondern auch neue QR-Codes bei der Schlüsselaktualisierung erzeugen.

Ausführlichere Programmcode Abbildungen sind im Anhang **Fehler! Verweisquelle konnte nicht gefunden werden.** zu finden oder die vollständige Android Applikation im folgenden Git-Repository [32].

9.5.1 Knotenregistrierung via QR-Code

Die Registrierung eines neuen Knoten innerhalb des MK verläuft in drei Phasen. In der ersten Phase wird der QR-Code Scanner aufgerufen, der den Code ausliest und die vorhandenen Informationen für die weitere Verarbeitung zurückgibt. Dabei besteht der QR-Code (siehe Abbildung 41) aus einem einzigartigen DNS Service Namen, einem Knotentyp z. B. Light und dem AES Schlüssel. Zum Aufruf des Scanners muss der Anwender lediglich den in Abbildung 42 gezeigten Optionsknopf „Add Node“ drücken. Innerhalb der zweiten Phase sucht der MK mithilfe des Service Namen nach dem entsprechenden Knoten im Netzwerk. Hierfür wird der in Android integrierte „NSDManager“ genutzt, der ein DNS Discovery Dienst mittels DNS-SD anbietet. Infolge dessen wird überprüft, ob der entsprechende Knoten im Netzwerk verfügbar und unter welcher IP Adresse er zu erreichen ist. Dieser Discovery Dienst lässt sich auch nach einem IP Wechsel (DHCP) einzelner Knoten wiederholen. Um die Knoten für den Discovery Dienst sichtbar zu machen, müssen diese lediglich einen Service unter dem Typ „_coap._udp“ und dem Namen im QR-Code anbieten. Dies ermöglicht eine eindeutige Zuweisung eines QR-Codes zu einem Knoten, wenn jeder Knoten als DNS Namen einen Zufallswert wie z. B. eine Chip-ID enthält. Um eine zusätzliche Sicherheitsbarriere zu gewährleisten, kann dieser DNS Service mit einem Tastendruck am Knoten kombiniert werden. Somit würde der Knoten lediglich für ein limitiertes Zeitfenster über den DNS Dienst erreichbar sein. In der dritten Phase der Registrierung, die lediglich nach erfolgreicher DNS Discovery aufgerufen wird, wird mithilfe der gesammelten Knoteninformationen DNS-Name, allgemeiner vom Nutzer festgelegter Knotenname, IP-Adresse, Knotentyp und einer zufälligen Key-ID ein neuer Knoten zur App internen SQLite Datenbank hinzugefügt. Des Weiteren wird mithilfe der Key-ID, der ausgelesene AES Schlüssels in den „KeyStore“ eingespeichert.

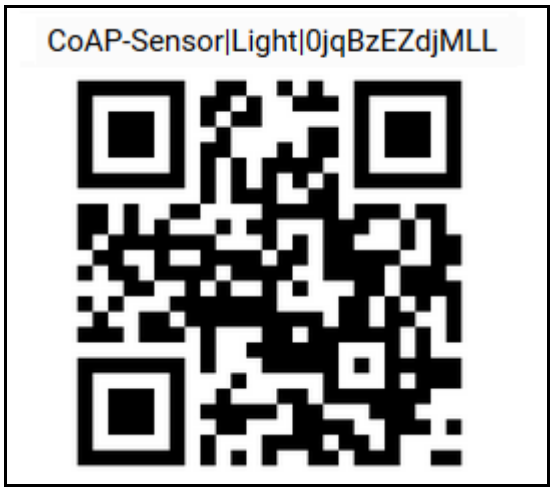

Abbildung 41: Beispiel QR-Code eines Knoten

Um eine Modularität der App zu gewährleisten, können auf Basis des Knotentyps neue Knotenfunktionalitäten hinzugefügt werden. So wurde innerhalb des Proof-of-Concept eine Light_Activity implementiert, die für Knoten des Types Light eine angepasste Funktionsoberfläche für den Nutzer anbietet. In Abbildung 42 ist diese Funktionalität dargestellt. Links ist die Hauptoberfläche der App zu sehen, die eine Auflistung aller registrierter Knoten und Gruppen im Netzwerk beinhaltet. Rechts die Oberfläche die der Nutzer erhält, wenn er auf einen Knoten links klickt. Wie zu erkennen ist beinhaltet die Oberfläche rechts einen „Switch Light“ Button, der es dem Nutzer ermöglicht gesichert die unter der IP erreichbare Lampe an oder aus zu schalten. Diese Kommunikation wird zum Testen der implementierten Unicast Verbindung verwendet.

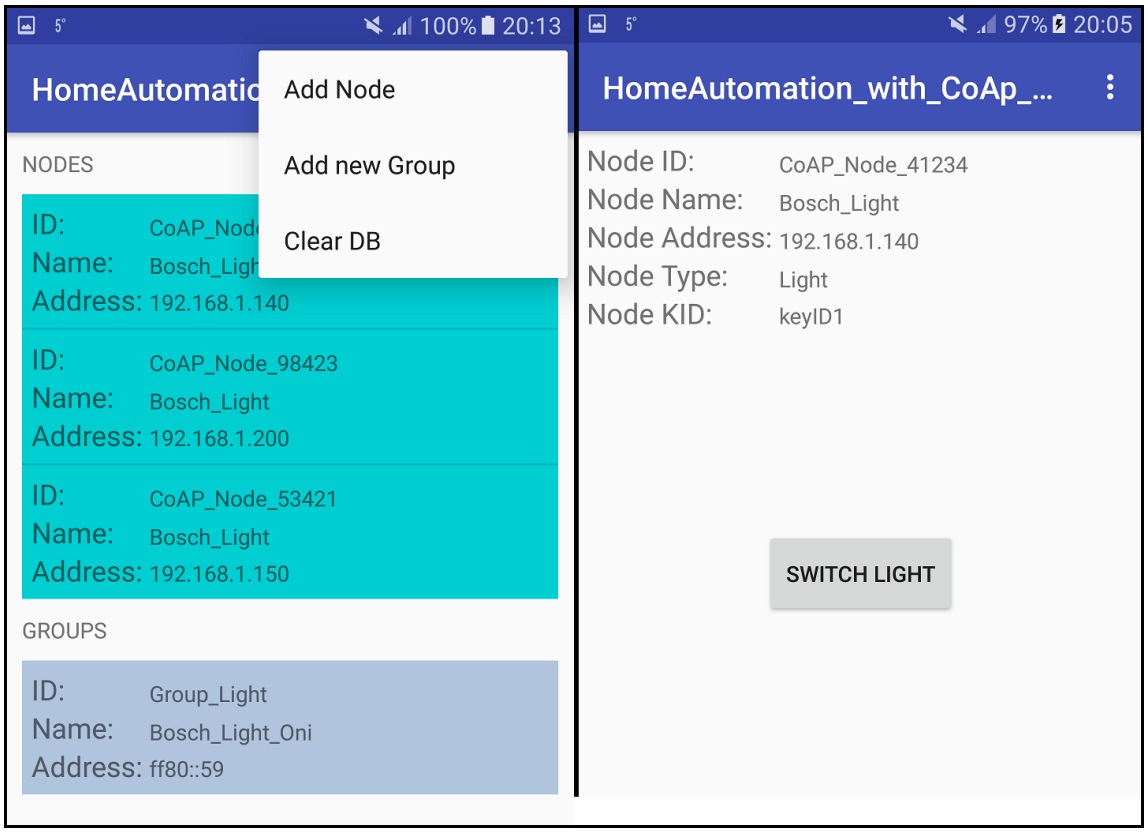

Abbildung 42: Screenshot Knoten Activity

Zum Abschluss des Kapitels wird die Implementierung des in Abbildung 42 gezeigten „Switch Light“ Button erläutert. Dabei wird die CoAP inklusive DTLS Umsetzung mithilfe der Californium Bibliothek beschrieben. Abbildung 43 zeigt den gesamten Programmcode, der für den Aufbau einer DTLS Session zu einem Knoten und anschließender CoAP Kommunikation benötigt wird. Dabei wird zu Beginn, wie bereits in Kapitel 9.4 beschrieben wurde ein „KeyStore“ geladen. In diesen „KeyStore“ wurde vorher, der aus dem QR-Code gelesene Schlüssel, unter einer KeyID (siehe Abbildung 42) gesichert. Anschließend kann dieser Schlüssel auf Basis der in der Datenbank für den Knoten gespeicherten Key-ID aus dem „KeyStore“ ausgelesen. Californium nutzt für die Erzeugung einer DTLS Session ein sogenanntes „DtlsConnectorConfig“ Objekt, das alle Einstellungen für eine DTLS Verbindung beinhaltet. Bei der Erzeugung wird zusätzlich eine „InetSocketAddress“ übergeben, die die genutzte Senderadresse beschreibt. Im Falle des Button in der Light_Activity wird die allgemeine Smartphone IP „InetSocketAddress(0)“ verwendet. Anschließend werden die unterstützten Cipher Suites ausgewählt. Da die genutzten Raspberry-Pis lediglich TLS bzw. DTLS mit einem 128 Bit AES Key in CCM Modus unterstützen, wird lediglich diese Suite ausgewählt. Die letzte Einstellung, die am „DTLSConnector“ vorgenommen wird ist die Einstellung eines „PskStore“. Hier wird der genutzte Schlüssel und die genutzte Key-ID für den DTLS Handshake festgelegt. Dabei muss die Key-ID, wie bereits erläutert wurde, auf beiden Seiten der Kommunikation übereinstimmen. Somit wird hier der feste Wert „GPIO“ hinterlegt, der innerhalb des Knoten zum Ansprechen einer Lampe registriert wurde. Diese Implementierungsstrategie bedeutet, dass eine Liste von festgelegten IDs für definierte Funktionen veröffentlicht werden müsste, die von Entwicklern für ihre eigenen MK und Knoten genutzt werden kann. Somit könnte jeder Hersteller und jeder Entwickler auf Basis des Konzeptes seine Knoten absichern und eine Interoperabilität zu anderen Herstellern gewährleisten. Zum Abschluss der Methode wird ein sogenannter „CoapClient“ erzeugt, der mithilfe einer übergebenen CoAP-URI (args[0]) eine Kommunikation ermöglicht. Die URI wird hierbei wie in Kapitel 9.1.1 konstruiert. Für das Verschicken von CoAP-Anfragen benötigt der Client zum Abschluss noch einen sogenannten „Endpoint“, der im Falle einer DTLS Kommunikation mithilfe des „DtlsConnectorConfig“ Objektes erzeugt bzw. gebildet (dtlsBuilder.build) wird. Anschließend können über den Client beliebige CoAP Kommandos z. B. POST oder GET sicher verschickt werden.

```
// Key Store Init
File store = new File(getFilesDir(), "keyStore.bks");
KeyStore keyStore = KeyStore.getInstance("BKS", "BC");
// Key Store Init

keyStore.load(new FileInputStream(store), "endPass".toCharArray());
SecretKeySpec key = (SecretKeySpec) keyStore.getKey(nodeInformation[4],
        "endPass".toCharArray());

DtlsConnectorConfig.Builder dtlsBuilder =
        new DtlsConnectorConfig.Builder(new InetSocketAddress(0));

dtlsBuilder.setSupportedCipherSuites(new CipherSuite[]
        {CipherSuite.TLS_PSK_WITH_AES_128_CCM_8});
dtlsBuilder.setPskStore(new StaticPskStore("GPIO",
        Base64.encodeToString(key.getEncoded(),
                Base64.DEFAULT).trim().getBytes("UTF-8")));

CoapClient client = new CoapClient(args[0]);
client.setEndpoint(new CoapEndpoint(new DTLSConnector(dtlsBuilder.build()),
        NetworkConfig.getStandard()));

client.post("1", MediaTypeRegistry.TEXT_PLAIN);
```

Abbildung 43: Light Activity Switch (Example DTLS Connection)

9.5.2 Erzeugung von Kommunikationsgruppen

Ähnlich zur Knotenregistrierung besteht die Erzeugung von Kommunikationsgruppen aus drei Phasen. In Phase eins wählt der Nutzer aus seiner Liste von registrierten Knoten die gewünschten Gruppenmitglieder aus und gibt der Gruppe einen Namen. Dafür wurden die in Abbildung 44 gezeigten Funktionen implementiert. In der Hauptoberfläche der Applikation wurde neben dem „Add Node“ Optionsknopf ein „Add new Group“ Knopf implementiert. Dieser ruft nach einem Click das rechts in Abbildung 44 gezeigte Fenster auf. Hier kann der Anwender den Gruppennamen festlegen und die gewünschten Mitglieder auswählen. Anschließend wird mit einem Druck auf den „Generate Group“ Button die zweite Phase der Erzeugung gestartet.

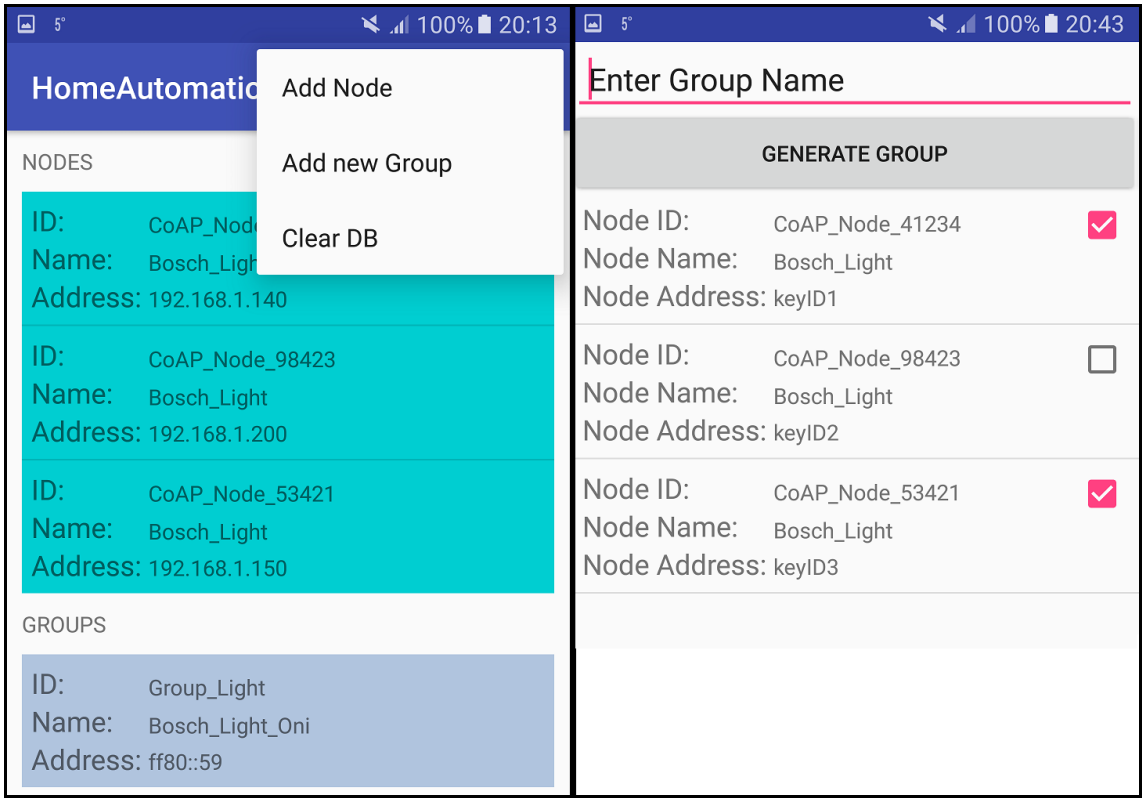

Abbildung 44: Screenshot Gruppenerzeugung

Innerhalb der zweiten Phase generiert der MK einen neuen 128 Bit AES Schlüssel und wählt eine freie Multicast Adresse für die Gruppe aus. Für die Erzeugung eines Schlüssels werden zwei verschiedene Methoden angeboten (siehe Abbildung 45). Zum Einen die Methode „genSymmetricKey“, die mithilfe eines Schlüsselgenerators einen 128 Bit AES-Schlüssel erzeugt. Zum Anderen die Methode „genRandomString“, die eine zufällige alphanumerische Zeichenfolge erzeugt, die als AES Schlüssel genutzt werden kann. Der Schlüsselgenerator erzeugt einen Schlüssel auf Byte-Ebene, wodurch dieser in verschiedenen Codierungen z. B. UTF-8 Sonderzeichen beinhalten kann. Diese Sonderzeichen können innerhalb der Kommunikation mit Knoten, abhängig von der gewählten Codierung auf den Knoten, zu Problemen führen. Infolge dessen werden zwei verschiedene Methoden angeboten, wovon die untere für die Schlüsselerzeugung innerhalb des Konzeptes genutzt wird. Abschließend wird die neue Gruppe in der dritten Phase jedem Gruppenmitglied via DTLS Unicast mitgeteilt. Dabei werden die einzelnen DTLS Sessions wie bereits in der Light-Activity (siehe Abbildung 43) aufgebaut. Innerhalb der Sessions wird den Knoten über eine CoAP POST Nachricht die Multicastadresse, der generierten Gruppenschlüssel und eine einzigartige Sender-ID übertragen. Mithilfe dieser Informationen können die einzelnen Knoten der Multicastgruppe beitreten und abgesichert

Nachrichten versenden bzw. empfangen. Der MK speichert die gesammelten Gruppeninformationen Gruppen-ID, Gruppenname, Multicastadresse, Schlüssel-ID und die Gruppenmitglieder in seiner Datenbank (SQlite).

```
/**
 * Generate a Symmetric Key
 */
private Key genSymmetricKey() {
    Key k = null;
    try {
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        kgen.init(128);
        k = kgen.generateKey();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return k;
}

/**
 * Generate random KeyString
 */
private String genRandomString(int len) {
    final String AB = "0123456789" +
            "ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
            "abcdefghijklmnopqrstuvwxyz";
    SecureRandom rnd = new SecureRandom();
    StringBuilder sb = new StringBuilder(len);
    for (int i = 0; i < len; i++)
        sb.append(AB.charAt(rnd.nextInt(AB.length())));
    return sb.toString();
}
```

Abbildung 45: Generierung eines zufälligen AES Schlüssels

9.5.3 Knoten- oder Gruppenschlüsselaktualisierung

Die Aktualisierung eines Schlüssels wird vom Anwender – wie das Registrieren eines neuen Knoten oder einer neuen Gruppe – über einen Menü Button ermöglicht. Abbildung 44 zeigt den entsprechenden Button, sowohl in der Knotenansicht (links) als auch in der Gruppenansicht (rechts). Dabei besitzt die eigentliche Aktualisierung abhängig vom Schlüsseltyp (Knoten- oder Gruppenschlüssel) entweder zwei oder drei Phasen. Im Falle einer Gruppenschlüsselaktualisierung sind es zwei Phasen. In der ersten Phase wird wie bereits bei der Erzeugung einer Kommunikationsgruppe ein neuer Schlüssel generiert (siehe Abbildung 45). Dieser Schlüssel wird im Anschluss in Phase zwei an die einzelnen Gruppenmitglieder per Unicast DTLS Kommunikation übertragen werden. Hierbei wird dasselbe Nachrichtenformat (Multicastadresse, Gruppenschlüssel und Sender-ID), wie bereits beim Gruppenbeitritt genutzt wurde, verwendet. Die Multicastadresse wird hierbei als Identifizierungsparameter genutzt und ermöglicht es einem Knoten den richtigen Schlüssel zu Aktualisieren. Dabei gilt es zu beachten das die einzelnen Knoten mit einer bereits bekannten Multicastadresse anders umgehen müssen. Dies kann durch eine spezielle CoAP Ressource, die lediglich für eine Schlüsselaktualisierung z. B.

„/keyUpdate“ genutzt wird, erreicht werden oder lediglich durch eine Filterung eingehender Nachrichten auf der Gruppenregistrierungsressource z. B. „/mc“. Innerhalb der Implementierung wird eine eigene Aktualisierungsressource genutzt.

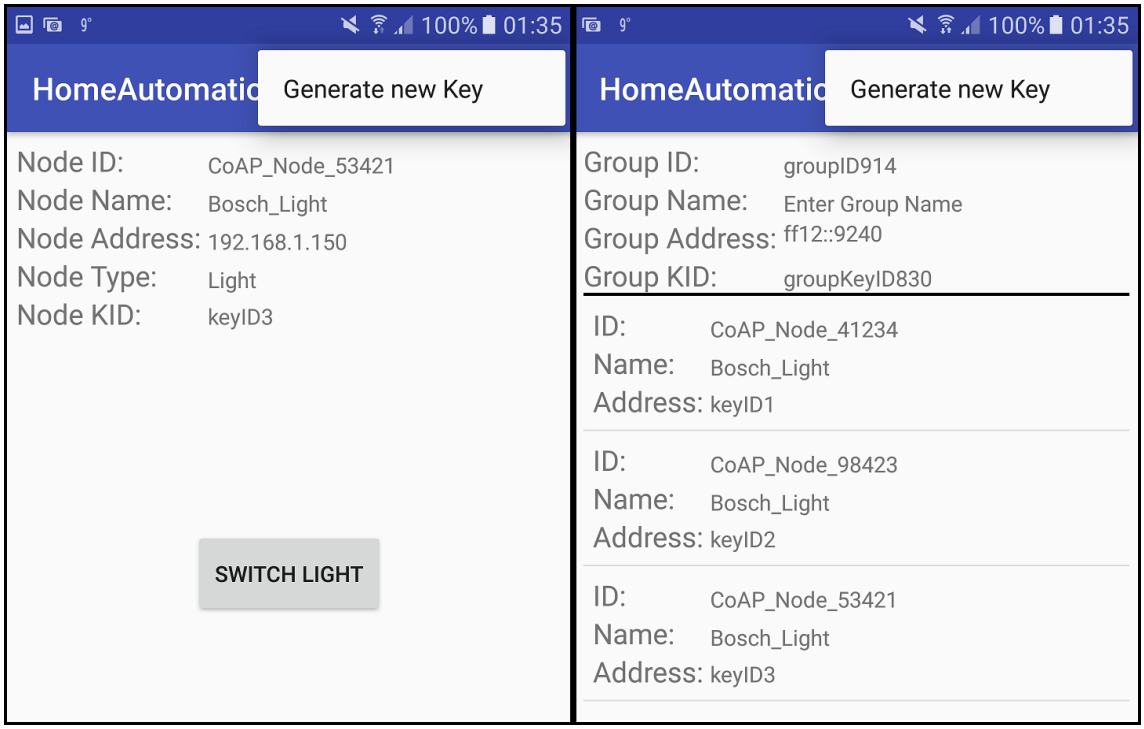

Abbildung 46: Screenshot Schlüsselaktualisierung

Wie bereits in Kapitel 8.5.3 erläutert wurde, besitzt die Aktualisierung eines Knotenschlüssels eine zusätzliche Phase. Innerhalb dieser Phase wird der neu erzeugte Knotenschlüssel für den gewählten Out-of-Band Transport verarbeitet. Somit teilt sich die Aktualisierung eines Knotenschlüssels in drei Phasen. Phase eins und zwei sind beinahe identisch zu einer Gruppenschlüsselaktualisierung. Innerhalb der ersten Phase wird ein neuer Schlüssel generiert, der anschließend in der zweiten Phase abgesichert mit DTLS an den jeweiligen Schlüssel verschickt wird. Hierbei unterscheidet sich der Aufbau der verschickten CoAP Nachricht. Diese beinhaltet lediglich die Empfängeradresse des Knoten und den neu generierten Schlüssel, da eine Sender-ID bei Unicast Kommunikation nicht benötigt wird. Dabei wird die Empfängeradresse als Identifizierungsparameter für den Knoten mit angegeben. So kann der Empfängerknoten, den für die genutzte Verbindung, verwendeten Schlüssel aktualisieren. In der letzten Phase wird dann aus dem Schlüssel und den benötigten Knoteninformationen z. B. Knotentyp, die verwendete Out-of-Band Ressource aktualisiert. Im Proof-of-Concept wird hier die Open-Source Bibliothek „ZXing“ [32] genutzt. „ZXing“ ist eine auf Java-basierende 1D/2D Barcode Bildverarbeitungsbibliothek, die es ermöglicht aus einem gegebenen String einen QR-Code zu erzeugen.

10 Evaluierung

Nachdem in Kapitel 8 ein Schlüsselmanagementkonzept für ein Smart Home System auf Basis eines Out-of-Band Schlüsselaustausches entwickelt und anschließend in Kapitel 9 eine Android-Implementierung für das Konzept erläutert wurde, wird im nachfolgenden Kapitel das Konzept mithilfe des in Kapitel 5 beschriebenen Bewertungsschema evaluiert und die Implementierung der Kommunikationsprotokolle mithilfe der Netzwerkanalysesoftware Wireshark überprüft.

Zu Beginn der Evaluierung wird ein genauerer Blick auf die Umsetzung der Schlüsselmanagement Disziplinen im Konzept geworfen. Wie in Abbildung 47 zu sehen ist, erfüllt das Konzept alle Schlüsselmanagement Disziplinen. Die Generierung der verwendeten Schlüssel wird innerhalb des Konzeptes nur mithilfe sicherer Zufallszahlengeneratoren erlaubt und die innerhalb der symmetrischen Kryptografie wesentlich kürzeren und leichter zu berechnenden Schlüsseln vereinfachen die Verarbeitung auf ressourcenarmen Knoten. Des Weiteren lassen sich die aktuell 128 Bit langen Schlüssel ohne Probleme auf 256 Bit oder mehr erweitern, sollten in den nächsten Jahren 128 Bit Schlüssel nicht mehr ausreichen. Ein Hauptkritikpunkt bei den etablierten Heimautomatisierungsstandards war der im gesamten Netzwerk gleiche Masterschlüssel. Um dieses Problem zu lösen, nutzt das Konzept eine Strategie, die an TLS Sitzungsschlüssel angelehnt ist. Hierbei wird ein Schlüssel pro Kommunikationsgruppe genutzt, wodurch ein erfolgreicher Angriff auf einen Knoten – solange es nicht der MK ist – nicht gleich das gesamte Netzwerk gefährdet. Des Weiteren werden die gelesenen Out-of-Band Schlüssel, lediglich für die Konfiguration von einzelnen Knoten oder Kommunikationsgruppen genutzt. Dies steigert weiter die Sicherheit, da die innerhalb der Gruppen genutzten Schlüssel niemals öffentlich oder ungesichert lesbar sind. Um eine sichere Aufbewahrung der einzelnen Schlüssel zu ermöglichen, werden im Konzept „KeyStores“ oder ähnliche verschlüsselte Speichermethoden gefordert. Somit können Angreifer nicht ohne Probleme – wie bei aktuellen Automatisierungsstandards – Schlüssel aus der Firmware oder Ähnlichem auslesen. Die kombinierte Verwendung einer Out-of-Band Kommunikation für den initialen Beitritt bzw. die Konfiguration und die anschließende Nutzung eines verschlüsselten Kommunikationsprotokolls für die Verteilung weiterer Schlüssel, ermöglicht eine ausschließlich abgesicherte Kommunikation innerhalb des gesamten Netzwerkes. Somit werden zu keinem Zeitpunkt unverschlüsselte Kommunikationen zwischen Knoten geführt.

Bei der Bewertung der funktionalen Anforderungen wird ein Problem einer rein symmetrischen Umsetzung sichtbar. Im Vergleich zu asymmetrischen oder hybriden Systemen wie z. B. dem TLS, die mithilfe von Zertifikaten einen genauen Authentizitätsnachweis bieten, wird im Konzept lediglich eine Out-of-Band Kommunikation genutzt. Grundlegend ermöglicht diese ebenfalls einen gewissen Grad von Authentizitätsnachweis, da lediglich der entsprechende Knoten im Netzwerk im besitzt des angegebenen Schlüssels sein sollte und auf die gelesene ID antwortet. Jedoch ist die Stärke eines Authentizitätsnachweises mithilfe von Zertifikaten über eine CA – wie sie z. B. bei TLS verwendet wird – wesentlich sicherer als lediglich ein Out-of-Band Nachweis. Davon abgesehen erfüllt das Konzept die restlichen funktionalen Anforderungen. Da es Vertraulichkeit über AES Verschlüsselung bietet, eine Integritätssicherung über einen MAC und die Verfügbarkeit über das Konzept nicht gestört wird. Des Weiteren können alle Schlüssel im Konzept aktualisiert werden, wodurch der mögliche Angriffsrahmen von Replay Angriffen auf ein Minimum reduziert werden kann und somit Aktualität der Nachrichten gewährleistet wird.

Da das Konzept mit dem Fokus auf ein Heimautomatisierungsnetzwerk entwickelt wurde, erfüllt es alle Punkte der letzten Bewertungskategorie (nicht-funktionale Anforderungen). Hierbei ermöglichen die gewählten Kryptografieverfahren einen möglichst ressourcensparenden Betrieb, was eine hohe Lebensdauer der Knoten ermöglicht. Zusätzlich ermöglicht das Out-of-Band Beitrittsprotokoll eine ungestörte Skalierung des Systems, da Knoten unabhängig von den restlichen Teilnehmern des Netzwerkes mit Aus-

nahme des MK dem Netzwerk oder einzelnen Kommunikationsgruppen beitreten können. Des Weiteren können einzelne Knoten mithilfe der gespeicherten Unicast Schlüssel jederzeit neu konfiguriert oder Sicherheitsparameter aktualisiert werden.

Schlüsselmanagement auf Basis eines Out-of-Band Schlüsselaustausches			
Schlüsselmanagement Disziplinen:			**32**
Generierung	✓	Schlüssellänge > 128 bit (variabel)	
Trennung	✓	Je Kommunikationsgruppe ein Schlüssel	
Speicherung	✓	Verschlüsselter „KeyStore“	
Austausch	✓	Out-of-Band und CoAP mit DTLS	
Vernichtung	✓		
Funktionale Anforderungen (Sicherheitsziele)			**16**
Vertraulichkeit	✓	AES Verschlüsselung (DTLS)	
Integrität	✓	MAC (DTLS)	
Verfügbarkeit	✓		
Authentizität	⦿	Out-of-Band Schlüsseltransport	
Aktualität	✓	Schlüsselaktualisierung	
Nicht-funktionale Anforderungen			
Low-power	✓		
Skalierbar	✓		
Hohe Lebensdauer	✓		
Wartbarkeit	✓		

Abbildung 47: Bewertung des entwickelten Konzeptes

Um die Sicherheit der implementierten Kommunikationsprotokolle zu gewährleisten, wurde die Kommunikation der verschiedenen Protokolle aus Kapitel 9.5 mit der Datenprotokollanalyse Software Wireshark mitgelesen und analysiert. Nachfolgend wird lediglich die Kommunikation zwischen dem MK und einem AK, die bei der Registrierung einer neuen Kommunikationsgruppe durchgeführt wird, näher betrachtet. Abbildungen zu den Mitschnitten der zwei verbleibenden Protokolle können im Anhang gefunden werden. Abbildung 48 zeigt eine Unicast Kommunikation zwischen dem MK und einem AK. Dabei

kann im Ablauf der Pakete gut der in Kapitel 9.1.2 erläuterte DTLS Handshake nachvollzogen werden. Zu Beginn der Kommunikation kann die Verifizierung der eingegangen „Client Hello“ Nachricht“, womit sich reguläre DoS Angriffe unterbinden lassen, erkannt werden. Hierbei wurden die „Client Hello“ Nachricht, der „Hello Verify Request“, der wiederholte „Client Hello“ und die finale „Server Hello“ Nachricht näher betrachtet.

No.	Source	Destination	Info
1	fe80::ae5f:3eff:…	fe80::6042:2d95…	Client Hello
2	fe80::6042:2d95:…	fe80::ae5f:3eff…	Hello Verify Request
3	fe80::ae5f:3eff:…	fe80::6042:2d95…	Client Hello
4	fe80::6042:2d95:…	fe80::ae5f:3eff…	Server Hello
5	fe80::6042:2d95:…	fe80::ae5f:3eff…	Server Hello Done
6	fe80::ae5f:3eff:…	fe80::6042:2d95…	Client Key Exchange, Change Cipher Spec, Finished
7	fe80::6042:2d95:…	fe80::ae5f:3eff…	Change Cipher Spec
8	fe80::6042:2d95:…	fe80::ae5f:3eff…	Finished
9	fe80::ae5f:3eff:…	fe80::6042:2d95…	Application Data
…	fe80::6042:2d95:…	fe80::ae5f:3eff…	Application Data

Abbildung 48: Unicast Kommunikation zwischen MK und AK

Wie in Abbildung 49 gezeigt wird ist innerhalb dem „Hello Verify Request“ gut der generierte und versendete Cookie zu erkennen. Dieser wird unverschlüsselt transportiert und vom Client in seiner nachfolgenden „Client Hello“ Nachricht zur Verifizierung der Handshake Anfrage benötigt.

```
Datagram Transport Layer Security
  DTLSv1.2 Record Layer: Handshake Protocol: Hello Verify Request
      Content Type: Handshake (22)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 0
      Sequence Number: 0
      Length: 31
    Handshake Protocol: Hello Verify Request
        Handshake Type: Hello Verify Request (3)
        Length: 19
        Message Sequence: 0
        Fragment Offset: 0
        Fragment Length: 19
        Version: DTLS 1.2 (0xfefd)
        Cookie Length: 16
        Cookie (16 bytes)

0000  ac 5f 3e 26 5d c4 b8 27  eb d0 d6 bf 86 dd 60 02   ._>&]..' ......`.
0010  6f b4 00 34 11 40 fe 80  00 00 00 00 00 00 60 42   o..4.@.. ......`B
0020  2d 95 d5 bf 07 f5 fe 80  00 00 00 00 00 00 ae 5f   -....... ......._
0030  3e ff fe 26 5d c4 16 33  82 62 00 34 63 ae 16 fe   >..&]..3 .b.4c...
0040  fd 00 00 00 00 00 00 00  00 00 1f 03 00 00 13 00   ........ ........
0050  00 00 00 00 00 00 13 fe  fd 10 c5 6d 4e 2a 89 6e   ........ ...mN*.n
0060  0e 55 67 45 10 6c b8 3c  1f 0f                     .UgE.l.< ..
```

Abbildung 49: DTLS „Server Hello Verify“ Cookie

Innerhalb dieser „Client Hello“ Nachricht (siehe Abbildung 50) kann sowohl der empfangene Cookie als auch die ausgewählte Cipher Suite „TLS_PSK_WITH_AES_CCM_8“ identifiziert werden.

```
Datagram Transport Layer Security
  DTLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 0
      Sequence Number: 1
      Length: 96
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 84
        Message Sequence: 1
        Fragment Offset: 0
        Fragment Length: 84
        Version: DTLS 1.2 (0xfefd)
      > Random
        Session ID Length: 0
        Cookie Length: 16
        Cookie: c56d4e2a896e0e556745106cb83c1f0f
        Cipher Suites Length: 2
      Cipher Suites (1 suite)
          Cipher Suite: TLS_PSK_WITH_AES_128_CCM_8 (0xc0a8)
```

Abbildung 50: DTLS zweiter „Client Hello“

Nachdem Client (MK) und Server (AK) erfolgreich die „Hello“ Phase des Handshakes abgeschlossen haben, erzeugt der Client die in Abbildung 51 gezeigte „Key Exchange“ Nachricht. Diese beinhaltet innerhalb der Nachricht den genutzten PSK Alias in Hexadezimaldarstellung. Innerhalb des Tests wurde der entsprechende Knoten mit dem PSK Alias „GPIO“ angesprochen, was einer Hexadezimalzahl von 4750494f entspricht.

```
Datagram Transport Layer Security
  DTLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 0
      Sequence Number: 2
      Length: 18
    Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 6
        Message Sequence: 2
        Fragment Offset: 0
        Fragment Length: 6
      PSK Client Params
          Identity Length: 4
          Identity: 4750494f
```

Abbildung 51: DTLS "Client Key Exchange"

```
Datagram Transport Layer Security
  DTLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 1
      Sequence Number: 1
      Length: 60
      Encrypted Application Data: 0001000000000001acae450d0c0cb696adff5da8ae9aa77e...

0000  b8 27 eb d0 d6 bf ac 5f  3e 26 5d c4 86 dd 60 00   .'....._ >&]...`.
0010  00 00 00 51 11 ff fe 80  00 00 00 00 00 00 ae 5f   ...Q.... ......._
0020  3e ff fe 26 5d c4 fe 80  00 00 00 00 00 00 60 42   >..&]... ......`B
0030  2d 95 d5 bf 07 f5 82 62  16 33 00 51 55 56 17 fe   -......b .3.QUV..
0040  fd 00 01 00 00 00 00 00  01 00 3c 00 01 00 00 00   ........ ..<.....
0050  00 00 01 ac ae 45 0d 0c  0c b6 96 ad ff 5d a8 ae   .....E.. .....]..
0060  9a a7 7e 6c 4a 00 a2 30  e6 43 02 2c 4d f5 67 16   ..~lJ..0 .C.,M.g.
0070  36 3f a2 9f 82 9e 62 10  41 89 64 f9 68 cd cb 89   6?....b. A.d.h...
0080  83 fd 63 b4 df 40 01                               ..c..@.

Datagram Transport Layer Security
  DTLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 1
      Sequence Number: 1
      Length: 60
      Encrypted Application Data: 0001000000000001acae450d0c0cb696adff5da8ae9aa77e...

0000  48 02 f9 a1 a0 f8 e7 ac  a2 fd ea f6 b2 6d 63 10   H....... .....mc.
0010  ff 66 66 31 32 3a 3a 31  35 34 31 2c 31 32 4d 43   .ff12::1 541,12MC
0020  6d 79 4c 46 76 74 6e 76  2c 37 33 32               myLFvtnv ,732
```

Abbildung 52: Application Data Ver -und Entschlüsselt

Nachdem der MK und der AK den Handshake erfolgreich beendet und mithilfe der „Change Cipher Suite“ Nachrichten den jeweiligen Kommunikationspartner über den Wechsel zur verschlüsselten Kommunikation informiert haben, wird die abgesicherte DTLS Session initiiert. Über diese Session wird anschließend die CoAP POST Nachricht bestehend aus einer Multicastadresse, einem generierten AES Schlüssel und einer Sender-ID verschlüsselt verschickt. Abbildung 52 zeigt diese Nachricht einmal in verschlüsselter Form (oben) und den entschlüsselten CoAP Nachrichtenteil (unten). Innerhalb des entschlüsselten Nachrichtenteils lässt sich gut die angesprochene CoAP Ressource „mc“ erkennen, sowie die gesendeten Parameter Multicastgruppe (ff12::1541), AES Gruppenschlüssel (12MCmyLFvtnv) und Sender-ID (732). Die Analyse der implementierten Kommunikationsprotokolle mit Wireshark hat gezeigt, das zu keinem Zeitpunkt sensible Informationen wie z. B. ein Schlüssel ungesichert über das Netzwerk kommuniziert werden. Des Weiteren wurde die Sicherheit der Implementierung des entwickelten Konzeptes bestätigt.

11 Zusammenfassung und Ausblick

Zum Abschluss der Thesis werden die Ergebnisse und Erkenntnisse der Arbeit zusammengefasst und ein Ausblick auf zukünftige Arbeiten und Projekte auf Basis des entwickelten Konzeptes gegeben.

Die Ausgangslage der Arbeit waren etablierte Heimautomatisierungsprotokolle wie z. B. Zigbee oder Z-Wave, die ein unzureichendes Schlüsselmanagement besitzen. So werden Schlüssel unsicher gespeichert, lediglich ein einzelner Masterschlüssel für alle Knoten verwendet oder unverschlüsselt Schlüssel ausgetauscht. Diese Strategien des Schlüsselmanagements können im Rahmen der Informationssicherheit lediglich als bedenkliche und unsichere Praktiken bezeichnet werden. Infolge dessen wurden die wichtigsten Eigenschaften der Sicherheit in der Heimautomatisierung, eines Sicherheitskonzeptes und einem sicheren Schlüsselmanagement identifiziert und erläutert.

Auf Basis dieser Erkenntnisse wurde ein Bewertungsschema für die Evaluierung von etablierten Konzepten wie z. B. TLS und aktuellen Forschungskonzepten entwickelt. Mithilfe des Bewertungsschemas wurde dann eine Vielzahl von verschiedenen Schlüsselmanagementstrategien insbesondere Schlüsselaustauschverfahren evaluiert und miteinander verglichen. Die so gewonnen Erkenntnisse wurden anschließend für die Entwicklung eines eigenen Schlüsselmanagement Konzeptes auf Basis eines Out-of-Band Schlüsselaustausches genutzt.

Das Konzept beinhaltet dabei konkrete Strategien und Empfehlungen für die Kategorien Schlüsselgenerierung, -trennung, -speicherung, -austausch und –vernichtung. Dabei werden z. B. kryptografisch sichere Schlüsselgeneratoren, eine verschlüsselte Speicherung der Schlüssel und voneinander unabhängige Schlüssel je Kommunikationsgruppe gefordert. Im Konzept wird für jeden Automatisierungsknoten ein Eigener initialer Out-of-Band Schlüssel genutzt, der mithilfe eines Managementknotens oder Controller dazu genutzt werden kann den entsprechenden Knoten zu konfigurieren, bestehende Schlüssel zu aktualisieren oder weitere Schlüssel auszuhandeln. Im Anschluss an die Entwicklung des Konzeptes, wurde es in einer Proof-of-Concept Implementierung auf Realisierbarkeit hin überprüft. Dafür wurde das Konzept in einer Android Applikation, auf Basis eines bestehenden Automatisierungsnetzwerkes mit DTLS gesicherter CoAP Kommunikation, implementiert.

Im Anschluss an die erfolgreiche Implementierung des Konzeptes, wurde das Konzept sowie die Implementierung auf Basis des entwickelten Bewertungsschemas und mithilfe des Datenpaket Analysewerkzeug Wireshark hin evaluiert. Dabei konnte die erfolgreiche Umsetzung des Konzeptes bestätigt werden, wodurch die Entwicklung eines Schlüsselmanagementkonzeptes für ein Heimautomatisierungssystem erfolgreich durchgeführt wurde.

Das größte Problem bei der Entwicklung eines Schlüsselmanagementkonzeptes für die Heimautomatisierung, liegt im schmalen Grat zwischen Sicherheit und Verlust des Nutzerkomforts. Sobald die Sicherheit eines Systems, nur über komplizierte Verfahren oder Abläufe ermöglicht werden kann, ist die Wahrscheinlichkeit das Nutzer das System ablehnen besonders hoch. Das zweite Problem liegt in den aktuell starken Begrenzungen der Systemressourcen eines typischen Heimautomatisierungsknoten. So können viele Verfahren aus der IT nicht ohne Probleme wie z. B. lange Berechnungszeiten in ein Automatisierungsnetzwerk implementiert werden.

Die aktuelle Implementierung stellt lediglich die grundlegenden Funktionalitäten für das Erfüllen des Konzeptes dar. Somit können basierend auf dem Konzept und der Implementierung weitere Projekte zur Ausweitung der Funktionalitäten und Entwicklung einer marktreiferen Applikation angeboten werden. Des Weiteren fehlt der aktuellen Implementierung die Möglichkeit vom Managementknoten via Multicast abgesichert zu einer erzeugten Kommunikationsgruppe direkt Nachrichten zu senden oder zu empfangen. Dies könnte ebenfalls als zukünftiges Projekt formuliert werden.

12 Literaturverzeichnis

[1] T. Rick, „Home Automation Keymanagement," 2016. [Online]. Available: http://mc-lab.inf.h-brs.de/doc/keymanagement/Praxisprojekt_Schluesselmanagement_ThorstenRick_Final.pdf. [Zugriff am 26 Juni 2016].

[2] W. Granzer, Secure Communication in Home and Building Automation Systems, Wien: TU Wien, 2010.

[3] C. G. Günther und A. Brown Boveri, „An Identity-Based Key-Exchange Protocol," [Online]. Available: http://download.springer.com/static/pdf/16/chp%253A10.1007%252F3-540-46885-4_5.pdf?originUrl=http%3A%2F%2Flink.springer.com%2Fchapter%2F10.1007%2F3-540-46885-4_5&token2=exp=1466969809~acl=%2Fstatic%2Fpdf%2F16%2Fchp%25253A10.1007%25252F3-540-46885-4_5.pdf%.

[4] R. W. Zhu, G. Yang und D. S. Wong, „An efficient identity-based key exchange protocol with KGS forward," [Online]. Available: http://ac.els-cdn.com/S0304397507001120/1-s2.0-S0304397507001120-main.pdf?_tid=016c2efa-3bd7-11e6-b7a3-00000aacb360&acdnat=1466970711_cfd40214db33f2ac677ea515d6f6ff92. [Zugriff am 22 06 2016].

[5] C. P. Pfleeger, S. L. Pfleeger und J. Margulies, Security in Computing, United States: Prentice Hall, 2015.

[6] M. Bedner und T. Ackermann, „Schutzziele der IT-Sicherheit," 2010. [Online]. Available: https://www.uni-kassel.de/fb07/fileadmin/datas/fb07/5-Institute/IWR/Ro%C3%9Fnagel/veroeffentlichungen/bedner_ackermann_schutzziele_der_it_sicherheit_dud_2010_323.pdf. [Zugriff am 11 10 2016].

[7] D. Dzung, M. Naedele, T. P. von Hoff und M. Crevatin, „Security for Industrial Communication Systems," 2005. [Online]. Available: ftp://ftp.unicauca.edu.co/Facultades/FIET/DEIC/Materias/Redes%20Industriales/Articulos/seguridad_en_comunicaciones_industriales.pdf. [Zugriff am 11 10 2016].

[8] W. Kastner, G. Neugschwandtner, S. Soucek und M. H. Newman, „Communication Systems for Building Automation and Control," 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.324.2621&rep=rep1&type=pdf. [Zugriff am 19 07 2016].

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler und J. D. Tygar, „SPINS: Security Protocols for Sensor Networks," 2001. [Online]. Available: http://www.netsec.ethz.ch/publications/papers/mc2001.pdf. [Zugriff am 11 10 2016].

[10] A. J. Menzes, P. C. van Oorschot und S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2001.

[11] C. Karlof, N. Sastry und D. Wagner, „TinySec: A Link Layer Security Architecture For Wireless Sensor Networks," 3 11 2004. [Online]. Available: https://www.cs.berkeley.edu/~daw/papers/tinysec-sensys04.ps. [Zugriff am 17 10 2016].

[12] A. Seyit, Campete und Y. Bülent, „Key Distribution Mechanisms for Wirless Sensor Netzworks: a Survey," 23 03 2005. [Online]. Available: https://www.cs.rpi.edu/research/pdf/05-07.pdf. [Zugriff am 17 10 2016].

[13] E. Barker, „NIST Recommendation for Key Management Part 1," Januar 2016. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf. [Zugriff am 1 November 2016].

[14] BSI, „Geeignetes Schlüsselmanagement," Bundesamt für Sicherheits in der Informationstechnik, 2013. [Online]. Available: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02046.html. [Zugriff am 1 November 2016].

[15] Wikipedia, „Public-Key-Infrastructure," 7 August 2007. [Online]. Available: https://en.wikipedia.org/wiki/Public_key_infrastructure. [Zugriff am 24 November 2016].

[16] I. Grebe, „Elliptische Kurven in der Kryptografie," Universität Potsdam, 2005. [Online]. Available: http://www.cs.uni-potsdam.de/ti/lehre/05-Kryptographie/slides/Elliptische_Kurven.pdf. [Zugriff am 5 Dezember 2016].

[17] T. Mattis, „Moderne Verschlüsselungen," 3 Dezember 2014. [Online]. Available: https://camo.githubusercontent.com/a2841d8666576b62444317495aec8c705ebd2d3a/68747470733a2f2f7261772e6769746875622e636f6d2f616d696e746f732f417274696b656c2f6d61737465722f4b727970746f6772616669652f637572766652e706e67. [Zugriff am 7 Dezember 2016].

[18] Network Working Group, „The Transport Layer Security (TLS) Protocol," Network Working Group, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246. [Zugriff am 27 12 2016].

[19] Network Working Group, „Advanced Encryption Standard (AES) Ciphersuites for Transport Layer," 2002. [Online]. Available: https://tools.ietf.org/html/rfc3268. [Zugriff am 27 12 2016].

[20] C. Eckert, IT Sicherheit Konzepte, Oldenbourg : Oldenbourg Wissenschaftsverlag GmbH, 2013.

[21] H. Gierow, „IP-Kamera nach 98 Sekunden mit Malware infiziert," Golem, 21 11 2016. [Online]. Available: http://www.golem.de/news/mirai-iot-botnet-ip-kamera-nach-98-sekunden-mit-malware-infiziert-1611-124602.html. [Zugriff am 28 12 2016].

[22] B. Azimi-Sadjadi, A. Kiayias, A. Mercado und B. Yener, „Robust Key Generation from Signal Envelopes in Wireless Networks," 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315295. [Zugriff am 4 Januar 2017].

[23] B. Azimi-Sadjadi, A. Kiayias, A. Mercado und B. Yener, „Robust Key Generation Generation from Signal Envelops in Wireless Networks," 2 November 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1315295.

[24] I. Mansour, G. Chalhoub und P. Lafourcade, „Key Management in Wireless Sensor Networks," 17 April 2015. [Online]. Available: http://www.mdpi.com/2224-2708/4/3/251. [Zugriff am 4 Januar 2017].

[25] S. Khan und J. L. Mauri, Security for Multihop Wireless Networks, Boca Raton, FL, USA: CRC Press, 2014.

[26] J. Konrad, „tinydtls," 2017. [Online]. Available: https://git.fslab.de/jkonra2m/tinydtls.

[27] J. Konrad, „aiocoapMC," 2017. [Online]. Available: https://git.fslab.de/jkonra2m/aiocoapMC.

[28] University of Glasgow Singapore, „DTLS-based Multicast Security in Constrained Environments," 3 Juli 2014. [Online]. Available: https://tools.ietf.org/html/draft-keoh-dice-multicast-security-08. [Zugriff am 20 Januar 2017].

[29] Google, Inc., „Datagram Transport Layer Security Version 1.2," Google, Inc., Januar 2012. [Online]. Available: https://tools.ietf.org/html/rfc6347. [Zugriff am Februar 2017].

[30] Google, „Dashboards," Google, 6 3 2017. [Online]. Available: https://developer.android.com/about/dashboards/index.html. [Zugriff am 10 3 2017].

[31] „GitHub: Californium," [Online]. Available: https://github.com/eclipse/californium. [Zugriff am 6 3 2017].

[32] T. Rick, „GitLab: Smart_Home_Key_Management_via_Smartphone," 12 3 2017. [Online]. Available: https://git.fslab.de/trick2s/Smart_Home_Key_Management_via_Smartphone. [Zugriff am 12 3 2017].

[33] „GitHub: Official ZXing ("Zebra Crossing") project home," [Online]. Available: https://github.com/zxing/zxing. [Zugriff am 6 3 2017].

[34] H. Diffie und M. E. Hellman, „New Directions in Cryptography," 6 11 1976. [Online]. Available: https://ee.stanford.edu/%7Ehellman/publications/24.pdf. [Zugriff am 12 10 2016].

13 Anhang

13.1 Applikations Methoden

```
private class MyDiscoveryListener implements NsdManager.DiscoveryListener {
    final String TAG = "SERVICE DISCOVERY:";
    String[] nodeInformation = null;

    MyDiscoveryListener(String[] nodeInformation) { this.nodeInformation = nodeInformation; }

    //  Called as soon as service discovery begins.
    @Override
    public void onDiscoveryStarted(String regType) { Log.d(TAG, "Service discovery started"); }

    @Override
    public void onServiceFound(NsdServiceInfo service) {
        // A service was found!  Do something with it.
        Log.d(TAG, "Service discovery success" + service);
        if (service.getServiceName().toLowerCase().equals(nodeInformation[0].toLowerCase())) {
            mNsdManager.stopServiceDiscovery(this);
            mNsdManager.resolveService(service, new MyResolveListener(nodeInformation));
        }
    }

    @Override
    public void onServiceLost(NsdServiceInfo service) {...}

    @Override
    public void onDiscoveryStopped(String serviceType) {...}

    @Override
    public void onStartDiscoveryFailed(String serviceType, int errorCode) {...}

    @Override
    public void onStopDiscoveryFailed(String serviceType, int errorCode) {...}


}
```

Abbildung 53: DNS Discovery Service

```
private class MyResolveListener implements NsdManager.ResolveListener {
    final String TAG = "SERVICE RESOLVER:";
    String[] nodeInformation = null;

    public MyResolveListener(String[] nodeInformation) {...}

    @Override
    public void onResolveFailed(NsdServiceInfo serviceInfo, int errorCode) {...}

    @Override
    public void onServiceResolved(final NsdServiceInfo serviceInfo) {
        Log.e(TAG, "Resolve Succeeded. " + serviceInfo);


        runOnUiThread(new Runnable() {
            Node node = new Node(nodeInformation[0], "NAME",
                    serviceInfo.getHost().toString().substring(1),
                    nodeInformation[1], "keyID" + (new Random().nextInt(100000 - 1) + 1));


            @Override
            public void run() {
                try {
                    // Key Store Init
                    File store = new File(getFilesDir(), "keyStore.bks");
                    KeyStore keyStore = KeyStore.getInstance("BKS", "BC");
                    keyStore.load(null, null);
                    keyStore.store(new FileOutputStream(store), "endPass".toCharArray());


                    // Format Key in Key Object
                    keyStore.load(new FileInputStream(store), "endPass".toCharArray());
                    byte[] aesByte = Base64.decode(nodeInformation[2].getBytes("UTF-8"),
                            Base64.NO_WRAP);
                    SecretKeySpec mAesKey = new SecretKeySpec(aesByte, "AES");

                    //Save Key to Store
                    keyStore.setKeyEntry(node.getNodeKID(), mAesKey, "endPass".toCharArray()
                            , null);
                    keyStore.store(new FileOutputStream(store), "endPass".toCharArray());

                    // Create new Node in SQLite DB
                    dbHelper.createNode(node);
                    listViewNodes.setAdapter(new SimpleCursorAdapter(
                            MainActivity.this, R.layout.node_info,
                            dbHelper.fetchAllNodes(),
                            nodeColumns,
                            to,
                            0));

                    // Refresh the ListView
                    displayListView();

                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

    }
}
```

Abbildung 54: DNS Resolve Service

```
// Generate new Communication Group + add GroupName
CommunicationGroup newGroup = new CommunicationGroup(generateRandomMCAddress());
newGroup.setGroupName(eText.getText().toString());

/*
 * Generate new Group AES Key and save to KeyStore + save to Group Object
 */
byte[] aesByte = null;
try {
    aesByte = Base64.decode(genRandomString(12).getBytes("UTF-8"), Base64.NO_WRAP);
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
SecretKeySpec aesKey = new SecretKeySpec(aesByte, "AES");

String groupKID = "groupKeyID" + (new Random().nextInt(1000 - 1) + 1);
newGroup.setGroupKID(groupKID);

try {
    loadKeyStore();
    keyStore.setKeyEntry(groupKID, aesKey, "endPass".toCharArray(), null);
    storeKeyStore();
} catch (NoSuchProviderException | KeyStoreException | CertificateException | NoSuchAlgorithm
    e.printStackTrace();
}
```

Abbildung 55: Generierung eines neuen Gruppenschlüssels

```
/*
* Get the checked ListItems from ListView
* Add them to the Group and send the GroupData to the Nodes
 */
Cursor itemCursor;
List<Integer> checkedItemPositions = dataAdapter.getCheckedListItems();

for (Integer position : checkedItemPositions) {
    itemCursor = (Cursor) listView.getItemAtPosition(position);
    Node node = new Node(itemCursor.getString(1),
            itemCursor.getString(2),
            itemCursor.getString(3),
            itemCursor.getString(4),
            itemCursor.getString(5));

    InetAddress inetAddress = InetAddress.getByName(node.getNodeAdress());

    Log.i("NewGroupMemberAdded:", node.toString());
    newGroup.addMember(node);

    /*
    * Send Group Parameter to checked Nodes
    * PARAMETER  execute({ URI , NodeKID , GroupKID , GroupMCAddress , NodeSenderID})
     */
    if (inetAddress instanceof Inet4Address) {
        new CoApDTLS().execute("coap://" + inetAddress.toString().substring(1) + ":5683" + "/mc",
                itemCursor.getString(5),
                groupKID, newGroup.getMCAddress().toString(),
                "" + nodeSenderCounter);
    } else if (inetAddress instanceof Inet6Address) {
        new CoApDTLS().execute("coap://" + "[" + inetAddress.toString().substring(1) + "]" + ":5683" + "/mc",
                itemCursor.getString(5),
                groupKID, newGroup.getMCAddress().toString(),
                "" + nodeSenderCounter);
    }

}
```

Abbildung 56: Verteilung der Gruppenparameter an die Gruppenmitglieder

```
private void updateKey() {
    String qrInput = "";
    String keyString = "";
    try {
        loadKeyStore();

        // Generate new AES Key
        byte[] aesByte = new byte[0];
        try {
            keyString = genRandomString(12);
            aesByte = Base64.decode(keyString.getBytes("UTF-8"), Base64.NO_WRAP);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        SecretKeySpec aesKey = new SecretKeySpec(aesByte, "AES");

        // Generate and Save QR-Code
        qrInput = nodeInformation[0] + "|" + nodeInformation[3] + "|" + keyString;
        saveQR_Code(encodeAsBitmap(qrInput));


        // Send the updated Key to Node
        InetAddress inetAddress = InetAddress.getByName(nodeInformation[2]);

        if (inetAddress instanceof Inet4Address) {
            new DTLS_KeyUpdate().execute(new String[]{"coap://" + nodeInformation[2]
                    + ":5683" + "/keyUpdate", keyString});
        } else if (inetAddress instanceof Inet6Address) {
            new DTLS_KeyUpdate().execute(new String[]{"coap://" + "[" + nodeInformation[2] + "]"
                    + ":5683" + "/keyUpdate", keyString});
        }
    } catch (IOException | CertificateException | NoSuchProviderException | KeyStoreException |
        e.printStackTrace();
    }
}
```

Abbildung 57: Generierung eines Knotenschlüssels und Austausch mit dem Knoten

```
/*
* Save QR-Code to external Storage
 */
private void saveQR_Code(Bitmap bitmap) {
    OutputStream fOut = null;
    try {
        File file = new File(Environment.getExternalStorageDirectory(),
                nodeInformation[0] + ".jpeg");
        fOut = new FileOutputStream(file);

        bitmap.compress(Bitmap.CompressFormat.JPEG, 85, fOut);
        fOut.flush();
        fOut.close();

        MediaStore.Images.Media.insertImage(getContentResolver(),
                file.getAbsolutePath(), nodeInformation[0], nodeInformation[0]);

    } catch (IOException e) {
        e.printStackTrace();
    }
}

/*
* Generate QR-Code with Key-String
 */
Bitmap encodeAsBitmap(String str) throws WriterException {
    BitMatrix result;
    try {
        result = new MultiFormatWriter().encode(str,
                BarcodeFormat.QR_CODE, 500, 500, null);
    } catch (IllegalArgumentException iae) {
        // Unsupported format
        return null;
    }
    int w = result.getWidth();
    int h = result.getHeight();
    int[] pixels = new int[w * h];
    for (int y = 0; y < h; y++) {
        int offset = y * w;
        for (int x = 0; x < w; x++) {
            pixels[offset + x] = result.get(x, y) ? BLACK : WHITE;
        }
    }
    Bitmap bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
    bitmap.setPixels(pixels, 0, bitmap.getWidth(), 0, 0, w, h);
    return bitmap;
}
```

Abbildung 58: Erzeugung und Speicherung eines QR-Codes via mithilfe von ZXing

13.2 Wireshark

```
> Internet Protocol Version 6, Src: fe80::6042:2d95:d5bf:7f5, Dst: ff02::fb
> User Datagram Protocol, Src Port: 5353, Dst Port: 5353
v Multicast Domain Name System (response)
    Transaction ID: 0x0000
  > Flags: 0x8400 Standard query response, No error
    Questions: 0
    Answer RRs: 4
    Authority RRs: 0
    Additional RRs: 0
  v Answers
    > _coap._udp.local: type PTR, class IN, CoAP-Sensor._coap._udp.local
    > CoAP-Sensor._coap._udp.local: type TXT, class IN, cache flush
    > CoAP-Sensor._coap._udp.local: type SRV, class IN, cache flush, priority 0, weight 0
    > CoAP-Sensor.local: type AAAA, class IN, cache flush, addr fe80::6042:2d95:d5bf:7f5

0000  33 33 00 00 00 fb b8 27  eb d0 d6 bf 86 dd 60 06   33.....' ......`.
0010  a8 f0 00 87 11 ff fe 80  00 00 00 00 00 00 60 42   ........ ......`B
0020  2d 95 d5 bf 07 f5 ff 02  00 00 00 00 00 00 00 00   -....... ........
0030  00 00 00 00 00 fb 14 e9  14 e9 00 87 af df 00 00   ........ ........
0040  84 00 00 00 00 04 00 00  00 00 05 5f 63 6f 61 70   ........ ..._coap
0050  04 5f 75 64 70 05 6c 6f  63 61 6c 00 00 0c 00 01   ._udp.lo cal.....
0060  00 00 11 94 00 0e 0b 43  6f 41 50 2d 53 65 6e 73   .......C oAP-Sens
0070  6f 72 c0 0c c0 28 00 10  80 01 00 00 11 94 00 01   or...(.. ........
0080  00 c0 28 00 21 80 01 00  00 00 78 00 14 00 00 00   ..(.!... ..x.....
0090  00 16 33 0b 43 6f 41 50  2d 53 65 6e 73 6f 72 c0   ..3.CoAP -Sensor.
00a0  17 c0 55 00 1c 80 01 00  00 00 78 00 10 fe 80 00   ..U..... ..x.....
00b0  00 00 00 00 00 60 42 2d  95 d5 bf 07 f5            .....`B- .....
```

Abbildung 59: DNS Discovery Antwort eines Knoten

```
> Frame 9: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits) on interface 0
> Ethernet II, Src: SamsungE_26:5d:c4 (ac:5f:3e:26:5d:c4), Dst: Raspberr_d0:d6:bf (b8:27:eb:d0:d6:bf)
> Internet Protocol Version 6, Src: fe80::ae5f:3eff:fe26:5dc4, Dst: fe80::6042:2d95:d5bf:7f5
> User Datagram Protocol, Src Port: 59656, Dst Port: 5683
v Datagram Transport Layer Security
  v DTLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 1
      Sequence Number: 1
      Length: 74
      Encrypted Application Data: 0001000000000001beaea646c07c5f4ac6dbdb63679901b0...

0000  45 02 a8 ff f1 6f 9e 05  36 b9 6b 65 79 55 70 64   E....o.. 6.keyUpd
0010  61 74 65 10 ff 66 65 38  30 3a 3a 36 30 34 32 3a   ate..fe8 0::6042:
0020  32 64 39 35 3a 64 35 62  66 3a 37 66 35 2c 5a 6b   2d95:d5b f:7f5,Zk
0030  58 39 50 67 69 4c 53 62  70 4b                     X9PgiLSb pK
```

Abbildung 60: Key Update entschlüsselte Applikationsdaten