Sichere Dienste-Suche in Sensornetzen

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von
Dipl.-Inform.
Hans-Joachim Hof
aus Aalen

Tag der mündlichen Prüfung: 30. 11. 2007

Erster Gutachter: Prof. Dr. Martina Zitterbart
Zweiter Gutachter: Prof. Dr. Georg Carle

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagungen

Diese Arbeit hätte nicht entstehen können, wenn ich nicht breite Unterstützung und Hilfe am Institut für Telematik und in meinem privaten Umfeld gehabt hätte. Deshalb möchte ich hier den maßgeblichen Personen ganz herzlich danken!

An erster Stelle sei Frau *Professor Zitterbart* gedankt für die hervorragende Betreuung der Arbeit, für fruchtbare Diskussionen, für Unterstützung mit finanziellen Mitteln für Hardware und Veröffentlichungen auf internationalen Konferenzen und insbesondere dafür, dass ihre Tür für Mitarbeiter immer offen war. Herzlichen Dank!

Herzlicher Dank gilt ebenfalls Herrn *Professor Carle*, der trotz hoher Arbeitsbelastung das Koreferat meiner Arbeit übernahm und mir hilfreiche Kommentare gab.

Ich möchte allen meinen Kollegen am Institut für Telematik danken, die mich durch Diskussionen und technische Hilfestellung unterstützt haben. Im Speziellen danke ich *Michael Conrad* für den Zugang zu Ressourcen, ohne die diese Arbeit nicht hätte entstehen können. Ich danke *Detlev Maier* für seine Hilfe rund um die Sensor-Hardware, insbesondere für die beispiellose Unterstützung im Vorfeld der MobiSys-Konferenz, und für manche Lebensweisheit. Ich danke *Dr. Oliver Waldhorst* für seine Kommentare zu meiner Dissertation, die mir im richtigen Augenblick sehr geholfen haben. Ich danke *Andreas Kuntz* für das Probelesen von Texten und für zahllose Diskussionen zum Thema dienstorientierte Sensornetze. Ich danke *Bernhard Hurler* für die Zusammenarbeit bei der Erstellung des Demonstrators und für eine unvergessliche Woche auf Puerto Rico.

Dank gilt auch meinen ehemaligen HiWis, Studien- und Diplomarbeitern, insbesondere *Christian Haas*, *Anton Hergenröder*, *Ingmar Baumgart*, *Sebastian Mies*, *Johannes Plöttner*, *Pascal Birnstill* und *Matthias Wurm*. Es freut mich ganz besonders, dass sich Christian, Ingmar und Sebastian ebenfalls für eine Promotion am Institut für Telematik entschieden haben.

Schließlich gilt mein ganz besonderer Dank meiner *Frau Anika*, die mich während der gesamten Zeit meiner Promotion unterstützte, mich aufgebaut hat, wenn die Experimente mal wieder nicht funktionierten, und mir insbesondere in der „heißen Phase“ den Rücken freigehalten hat.

Diese Arbeit wäre nicht möglich gewesen ohne die kontinuierliche Unterstützung, die mir meine Eltern *Lore und Dr. Hans-Jürgen Hof* während der Schulzeit, dem Studium und der Promotion gewährt haben. Ich danke Euch ganz herzlich!

1. Einleitung

Bereits im Altertum war mit dem Heliographen (ca. 405 vor Christus) eine Methode zur drahtlosen Kommunikation bekannt. Während damals noch Spiegel zur Übertragung von Lichtimpulsen zum Einsatz kamen, spielt heute in der drahtlosen Kommunikation vor allem die Übertragung durch Funk eine Rolle. Diese wurde erstmals 1896 durch Guglielmo Marchese Marconi demonstriert, basierend unter anderem auf wissenschaftlichen Erkenntnissen des Karlsruher Forschers und Nobelpreisträgers Heinrich Rudolf Hertz. In den 111 Jahren von 1896 bis heute kristallisierten sich verschiedene Bereiche in der drahtlosen Kommunikation heraus, mit ganz unterschiedlichen Anwendungsgebieten: Mobilfunknetze, Satellitenkommunikation, drahtlose LANs, schnurlose Telefone, Ad-hoc Netze und viele weitere.

Eine wichtige Rolle in der Entwicklung der Informatik im Allgemeinen und der drahtlosen Kommunikation im Speziellen spielte die ständig voranschreitende Miniaturisierung der eingesetzten Elektronik. Bereits 1965 sagte Gordon Moore – Mitbegründer von Intel – voraus, dass sich die Zahl der Transistoren auf einem Chip ungefähr alle 2 Jahre verdoppeln würde [69]. Dieser Vorhersage hat sich in der Vergangenheit annähernd bestätigt und wird heute als „Mooresches Gesetz“ bezeichnet. Die fortschreitende Miniaturisierung sowie Fortschritte bei der drahtlosen Kommunikation war 1991 der Anlass für einen viel beachteten Artikel [112] in dem die Vision von Ubiquitous Computing umrissen wird: in Zukunft werden die Menschen von einer Vielzahl von Miniatur-Computern umgeben sein, die in unsere Umgebung eingebettet sind und untereinander und mit den Menschen kommunizieren. Diese Vision wurde in vielen Forschungsvorhaben aufgegriffen und weiter differenziert. Weitere Forschungsfelder entstanden, in denen die drahtlose Kommunikation zwischen Miniatur-Computern eine wichtige Rolle spielt, unter anderem auch *drahtlose Sensornetze*.

Drahtlose Sensornetze bestehen aus Sensorknoten, die drahtlos, üblicherweise über Funk, miteinander kommunizieren, um durch Sensoren gemessene Werte zu übermitteln oder um Aktoren anzusteuern. In [84] wird der Entwurfsraum von Sensornetzen

im Allgemeinen umrissen. Aus dem Artikel wird deutlich, wie vielfältig Sensornetze und deren Anwendungen sein können.

Eine spezielle Klasse von drahtlosen Sensornetzen sind *dienstorientierte Sensornetze*. Zusätzlich zu den oben beschriebenen Eigenschaften von drahtlosen Sensornetzen verfügen dienstorientierte Sensornetze über eine Abstraktion, die so genannte Dienstabstraktion, welche die Bereitstellung von jedweder Funktionalität in Form von Diensten ermöglicht. Ein Dienst ist in diesem Kontext eine Funktionalität, die über eine spezifizierte Schnittstelle in Anspruch genommen werden kann. In dienstorientierten Sensornetzen werden Anwendungen durch Erstellen einer Reihe von Diensten und der Definition deren Zusammenspiels erstellt. Die Dienstabstraktion bietet viele Vorteile: mittels Diensten kann z. B. einfach Redundanz erzeugt werden, um mit dem Ausfall von Diensten umgehen zu können. So kann z. B. derselbe Dienst von verschiedenen Sensorknoten zur Verfügung gestellt werden, womit der Benutzer nach einem Ausfall immer noch mehrere Dienste zur Auswahl hat. Da die Schnittstellen zum Zugriff auf Dienste festgelegt sind, ermöglicht es die Dienstabstraktion, dass verschiedene Applikationen dieselben Dienste verwenden bzw. dass neue Applikationen durch Kombination bestehender Dienste erzeugt werden können. Weiterhin ermöglicht es die Dienstabstraktion, einfach verteilte Anwendungen zu erstellen. Darüber hinaus bietet die Dienstabstraktion die Möglichkeit, veraltete Dienste (unter Beibehaltung der Schnittstelle) gegen neue auszutauschen.

1.1 Problemstellung

Um für die Komposition von Anwendungen aus Diensten maximale Flexibilität zu erreichen, wird ein Verfahren benötigt, um die bereits im Netz vorhandenen Dienste aufzufinden. Diese so genannte *Dienste-Suche* stellt eine Basiskomponente dienstorientierter Sensornetze dar. Viele der oben angesprochenen Vorteile von dienstorientierten Sensornetzen, wie z. B. *Redundanz*, *Wiederverwendbarkeit* und *Austauschbarkeit*, können unkompliziert realisiert werden, wenn der Sensorknoten, auf dem ein Dienst ausgeführt wird, nicht von vornherein festgelegt und bekannt ist (z. B. über seine Vermittlungsschichtadresse), sondern Sensorknoten im Netz angebotene Dienste rein durch Kenntnis eines Dienstnamens (oder einer anderen geeigneten Beschreibung) im laufenden Betrieb auffinden können.

Anwendungen für dienstorientierte Sensornetze bestehen aus einer Reihe von Diensten, die miteinander kommunizieren, um die gewünschte Funktionalität zu erbringen. Da dienstorientierte Sensornetze auch in sicherheitskritischen Anwendungsbereichen (z. B. Überwachung von Gefahrgut) eingesetzt werden, ist es wichtig, die *Dienstsuche sicher zu gestalten* um z. B. zu verhindern, dass ein Angreifer feindliche Dienste in eine Anwendung einschmuggeln kann, um damit die Anwendung zu infiltrieren, oder um zu verhindern, dass ein Angreifer die Dienste-Suche unbrauchbar macht.

1.2 Ziel der Arbeit

Die Problemstellung zeigt, dass eine sichere Dienste-Suche in dienstorientierte Sensornetze eine essenzielle Voraussetzung für sichere, verteilte Anwendungen ist. Ziel

dieser Arbeit ist es, eine sichere Dienste-Suche für drahtlose Sensornetze zu entwerfen und zu evaluieren.

Als Basis für die Dienste-Suche soll Content Addressable Network (CAN) [77] verwendet werden, um ein sicheres, verteiltes Diensteverzeichnis zu entwickeln, in dem der Anbieter eines Dienstes eine Dienstbeschreibung zu einem Dienst hinterlegen kann. Dabei soll eine Dienstbeschreibung z. B. alle für den Zugriff auf den Dienst notwendigen Informationen enthalten. CAN ist eine so genannte verteilte Hash-Tabelle, die es ermöglicht unter einem Dienstnamen eine Dienstbeschreibung zu speichern. Die Speicherung erfolgt dabei verteilt auf allen Knoten, die am Content Addressable Network teilnehmen. Im Entwurf von Content Addressable Networks wurden Sicherheitsaspekte nicht berücksichtigt, allerdings eignet sich die Struktur des Content Addressable Network gut, um Sicherheitsmechanismen zu etablieren. Deshalb wurde Content Addressable Network unter anderem unter folgenden Fragestellungen untersucht:

- Wie kann sichergestellt werden, dass nur vertrauenswürdige Knoten dem CAN beitreten?
- Wie können Schlüssel zwischen Knoten im CAN ausgetauscht werden?
- Wie können Nachrichten zwischen den Knoten des CAN geschützt werden?
- Wie kann der Vorgang des Veröffentlichens einer Dienstbeschreibung geschützt werden?
- Wie kann der Vorgang des Auffindens einer Dienstbeschreibung geschützt werden?
- Wie können Dienstbeschreibungen während der Speicherung geschützt werden?

1.3 Gliederung der Arbeit

In der vorliegenden Arbeit wird der Entwurf eines sicheren, verteilten Diensteverzeichnisses für Sensornetze vorgestellt. Dazu werden in Kapitel 2 Grundlagen eingeführt, die für das Verständnis der Arbeit notwendig sind. In Kapitel 3 werden mögliche Angriffe vorgestellt, ein Angreifermodell entwickelt sowie untersucht, welche kryptographischen Bausteine sich für den Entwurf eines sicheren, verteilten Diensteverzeichnisses für Sensornetze eignen. In Kapitel 4 wird anschließend der Stand der Forschung zur Dienste-Suche beleuchtet. In Kapitel 5 wird der Entwurf dieses sicheren, verteilten Diensteverzeichnisses vorgestellt. Zur Leistungsbewertung wurde das sichere, verteilte Diensteverzeichnis sowohl im Simulator GloMoSim als auch, als Prototyp, auf handelsüblichen Sensorknoten implementiert. Die Implementierung sowie die Ergebnisse der Leistungsbewertung werden in Kapitel 6 beschrieben. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick auf weitere Arbeiten in Kapitel 7

2. Grundlagen

Im diesem Kapitel werden die Grundlagen beschrieben, die zum Verständnis dieser Arbeit notwendig sind. Insbesondere werden in diesem Kapitel Grundlagen zu drahtlosen Sensornetzen (Abschnitt 2.1), kryptographischen Verfahren (Abschnitt 2.2) und dem Overlay-Netz Content Addressable Network (Abschnitt 2.3) vermittelt.

2.1 Überblick zu drahtlosen Sensornetzen

Drahtlose Sensornetze bestehen aus Instanzen, die drahtlos, häufig über Funk, miteinander kommunizieren. Die kommunizierenden Instanzen werden im folgenden *Sensorknoten* oder auch kurz nur *Knoten* genannt. Sensorknoten können je nach Funktionalität als *Sensor* und/oder als *Aktor*[1] auftreten. Ein Sensorknoten, der physikalische Eigenschaften seiner Umgebung bestimmt, wird als Sensor bezeichnet. Ein Beispiel für einen Sensor ist ein Sensorknoten, der die Helligkeit ermittelt. Ein Aktor ist ein Sensorknoten, der ein System ansteuert, das elektrische Signale in physikalische Größen wie z.B. Temperatur, Druck, Drehmoment etc. umsetzt. Ein Sensorknoten, der einen Motor an- oder abschaltet, ist ein Beispiel für einen Aktor. In [84] wird ein Überblick darüber gegeben, was heute in der Forschergemeinde unter einem drahtlosen Sensornetz verstanden wird. Der dort beschriebene Entwurfsraum ist jedoch relativ groß, weswegen im Folgenden deutlich gemacht wird, welche Annahmen über drahtlose Sensornetze der vorliegenden Arbeit zugrunde liegen:

- *Viele Sensorknoten:* Die betrachteten Sensornetze bestehen aus einer, für ein Ad-hoc-Netz, großen Anzahl von Knoten von z. B. aus 1000 und mehr Knoten.

[1] Ein Aktor wird in der deutschen Literatur auch oft mit dem Anglizismus Aktuator bezeichnet (von engl. actuator). In dieser Arbeit wird der deutsche Begriff Aktor verwendet.

- *Hohe Knotendichte:* Diese große Anzahl von Knoten wird in einem begrenzten Gebiet eingesetzt. Daraus resultiert eine hohe Knotendichte. Obwohl das Gebiet, in dem ein Sensornetz eingesetzt wird, begrenzt ist, wird in dieser Arbeit davon ausgegangen, dass sich nicht alle Knoten in Empfangsreichweite zueinander befinden, d. h. es ist unter Umständen notwendig, dass Nachrichten über mehrere Zwischenknoten weitergeleitet werden, bevor sie ihr Ziel erreichen.

- *Leistungsschwache Sensorknoten:* Die Sensorknoten in den betrachteten Sensornetzen verfügen über sehr begrenzte Ressourcen, insbesondere über sehr wenig Hauptspeicher (z. B. 4 KB) und sehr wenig Programmspeicher (z. B. 128 KB). Darüber hinaus sind die eingesetzten Prozessoren leistungsschwach und langsam im Bezug auf die Rechenleistung (z. B. mit 16 MHz getaktet). Als typische Sensorknoten werden die so genannten *MICA Motes* der Firma Crossbow [1] betrachtet, im Speziellen die MICAz Motes [104], die in der Prototyp-Implementierung dieser Arbeit verwendet wurden.

- *Niedrige Datenraten:* In Sensornetzen werden üblicherweise Kommunikationstechnologien mit niedrigen Datenraten eingesetzt. Neben einer Menge proprietärer Protokolle existiert ZigBee [3] (mit einer Datenrate bis zu 250 kbit/s) als Kommunikationsstandard für Sensornetze.

- *Infrastrukturloses, dezentrales Netz:* In den betrachteten Sensornetzen sind keine zentralen Infrastrukturkomponenten wie z. B. eine Basisstation o.Ä. vorhanden, die für jeden Sensorknoten den größten Teil der Zeit erreichbar sind.

- *Drahtlose Kommunikation:* Die Sensorknoten, die in dieser Arbeit betrachtet werden, kommunizieren drahtlos per Funkt miteinander.

- *Begrenzte Lebenszeit von Sensorknoten:* Sensorknoten verfügen üblicherweise über einen begrenzten Energievorrat. Es ist also damit zu rechnen, dass Sensorknoten von Zeit zu Zeit wegen Energiemangel ausfallen.

Aus den genannten Annahmen wird deutlich, dass die in dieser Arbeit betrachteten Sensornetze im Bezug auf die Leistungsfähigkeit eher am unteren Rand des in [84] beschriebenen Entwurfsraums einzuordnen sind.

In dieser Arbeit wird eine spezielle Klasse von drahtlosen Sensornetzen betrachtet: *dienstorientierte Sensornetze.* Ein dienstorientiertes Sensornetz ist ein drahtloses Sensornetz, das eine *Dienstabstraktion* verwendet. Diese Dienstabstraktion besagt, dass jedwede Funktionalität als *Dienst* über eine *spezifizierte Schnittstelle* angeboten wird. Betrachten wir noch einmal das oben aufgeführte Beispiel eines Sensorknotens, der die Helligkeit messen kann. Dieser Sensorknoten kann die Funktionalität der Helligkeitsmessung in Form eines Dienstes mit dem Namen `liefere_Helligkeit` zur Verfügung stellen. Ein Aufruf dieses Dienstes liefert die gemessene Helligkeit zurück. In dienstorientierten Sensornetzen werden Anwendungen durch Erstellen einer Reihe

von Diensten und der Definition deren Zusammenspiels, wie z. B. der Kommunikation zwischen den Diensten, erstellt. Dazu ein Beispiel: Um in einem Raum die Helligkeit konstant zu halten, werden der oben beschriebene Dienst `liefere_Helligkeit` sowie zwei weiterer Dienste `Jalousie_auf` und `Jalousie_ab` verwendet. Wie die Namen der Dienste schon andeuten, steuern die Dienste `Jalousie_auf` und `Jalousie_ab` eine Jalousie an, die nach oben bzw. nach unten gefahren wird. Die drei Dienste können nun zur Anwendung `Helligkeitssteuerung` zusammengesetzt werden. Die Funktionalität der Anwendung `Helligkeitssteuerung` kann wiederum als Dienst im Netz angeboten werden. Abbildung 2.1 zeigt dieses Beispiel noch einmal.

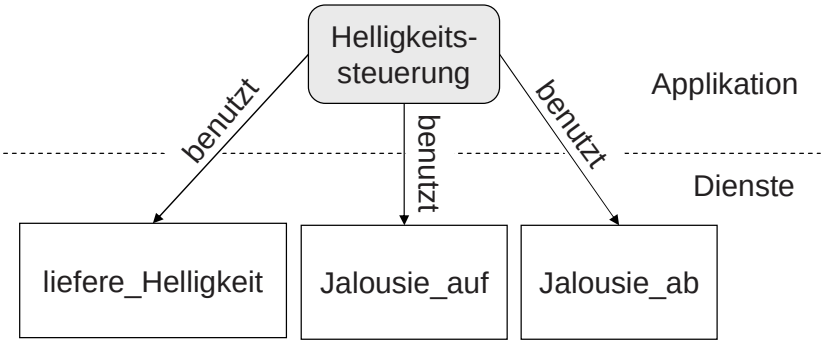


Abbildung 2.1: Beispiel einer verteilten Anwendung in einem dienstorientierten Sensornetz

Die Dienstabstraktion bietet viele Vorteile: z. B. kann mittels Diensten einfach mit dem Ausfall von Diensten umgegangen werden, indem derselbe Dienst von verschiedenen Sensorknoten zur Verfügung gestellt wird. Durch diese redundanten Dienste kann ein Dienst durch andere Dienste ersetzt werden. Da die Schnittstelle zum Zugriff auf einen Dienst festgelegt ist, ermöglicht die Dienstabstraktion die gleichzeitige Verwendung desselben Dienstes durch mehrere Anwendungen bzw. die Wiederverwendung von vorhandenen Diensten zur Erstellung neuer Applikationen durch Kombination bestehender Dienste. Weiterhin ermöglicht es die Dienstabstraktion, einfach verteilte Anwendungen zu erstellen, da durch die Dienstabstraktion bereits eine Aufteilung der Anwendung in Teile vorgegeben ist. Darüber hinaus bietet die Dienstabstraktion die Möglichkeit, veraltete Dienste (unter Beibehaltung der Schnittstelle) gegen neue auszutauschen.

2.2 Kryptographische Grundlagen

Dieses Kapitel führt in verschiedene grundlegende Definitionen und Techniken der Kryptographie ein. Insbesondere wird auch eine Notation eingeführt, die dann im Weiteren durchgehend verwendet wird. In dieser Arbeit werden keine kryptographischen Verfahren entworfen und auch nicht im Detail erklärt, weswegen für eine ausführlichere Betrachtung zu Kryptographie auf [17, 34, 105] verwiesen sei.

2.2.1 Sicherheitsanforderungen

Sicherheitsanforderungen definieren Ziele aus Sicherheitssicht. Die folgenden Sicherheitsanforderungen sind im Besonderen für diese Arbeit von Interesse. Die Definitionen sind an die Definitionen in [34] angelehnt:

- *Authentizität:* Unter der Authentizität eines Objekts (z. B. Nutzdaten einer Nachricht) bzw. eines Subjekts (z. B. eines Absenders) wird die Echtheit und Glaubwürdigkeit des Objekts bzw. Subjekts verstanden, die anhand einer eindeutigen Identität und charakteristischen Eigenschaft verifiziert werden kann.
- *Autorisierung:* Der Zugriff auf ein geschütztes Objekt ist nur denjenigen Subjekten oder Objekten möglich, die über die Berechtigung verfügen, auf dieses Objekt zuzugreifen.
- *Vertraulichkeit:* Ein Objekt (z. B. die Nutzdaten einer Nachricht) ist vertraulich, wenn kein unautorisierter Informationsgewinn erfolgen kann.
- *Integrität:* Die Integrität eines Objekts (z. B. Nutzdaten in einer Nachricht) ist gewährleistet, wenn es einem Subjekt (z. B. einem Angreifer) nicht möglich ist, das zu schützende Objekt unautorisiert und unbemerkt zu manipulieren

2.2.2 Verschlüsselung

Chiffren sind Algorithmen zur Ver- und Entschlüsselung von Daten. Chiffren werden oft eingesetzt, um Vertraulichkeit von Daten zu realisieren. Sie sind aus diesem Grund integraler Bestandteil vieler kryptographischer Protokolle. Man unterscheidet grob zwei Klassen der Verschlüsselung: symmetrische Verschlüsselung und asymmetrische Verschlüsselung. Außerdem ist es möglich, symmetrische und asymmetrische Verschlüsselungsverfahren zu einer dritten Klasse zu kombinieren, der so genannten hybriden Verschlüsselung. Im Folgenden wird genauer auf symmetrische und asymmetrische Verschlüsselung eingegangen.

Symmetrische Verschlüsselung unterscheidet sich von asymmetrischer Verschlüsselung vor allem dadurch, dass bei symmetrischer Verschlüsselung zum Ver- und Entschlüsseln der gleiche Schlüssel verwendet wird, während bei asymmetrischer Verschlüsselung verschiedene Schlüssel Verwendung finden.

Werden Daten *data* mit dem Schlüssel *key* symmetrisch verschlüsselt, so wird dies üblicherweise geschrieben als

$$c = E_{key}(data)$$

Dabei entsteht der Chiffretext c. Die entsprechende Entschlüsselung wird geschrieben als

$$data = D_{key}(c)$$

Es gilt also

$$data = D_{key}(E_{key}(data))$$

d. h. dass Daten, die mittels Schlüssel *key* verschlüsselt wurden, auch mit demselben Schlüssel wieder entschlüsselt werden können.

Innerhalb der Chiffren existieren zwei Klassen: Blockchiffren und Stromchiffren. Blockchiffren verschlüsseln Blöcke fester Länge in einem einzigen Durchgang, während Stromchiffren in jedem Arbeitsschritt lediglich ein einzelnes Zeichen verschlüsseln. Heutzutage gebräuchliche Chiffren sind AES [30] und DES/3DES [70], beides symmetrische Blockchiffren.

Asymmetrische Verschlüsselung zeichnet sich dadurch aus, dass statt einem einzelnen Schlüssel ein Schlüsselpaar, bestehend aus einem öffentlichen Schlüssel und einem privaten Schlüssel, verwendet wird. Die Sicherheit der Verschlüsselung hängt dabei lediglich von der Geheimhaltung des privaten Schlüssels ab, während der öffentliche Schlüssel, wie der Name schon sagt, öffentlich bekannt ist. Zur Verschlüsselung wird der öffentliche Schlüssel key_{pub} desjenigen verwendet, der die Daten *data* später entschlüsseln soll. In Kommunikationssystemen ist dies üblicherweise der Empfänger einer verschlüsselten Nachricht. Geschrieben wird die Verschlüsselung der Daten *data* als

$$c = E_{key_{pub}}(data)$$

Um den entsprechenden Chiffretext c wieder zu entschlüsseln wird der zu dem öffentlichen Schlüssel key_{pub} gehörige private Schlüssel key_{priv} verwendet. Dies wird geschrieben als

$$data = D_{key_{priv}}(c)$$

Es gilt also

$$data = D_{key_{priv}}(E_{key_{pub}}(data))$$

RSA [81] ist eine heute gebräuchliche asymmetrische Chiffre.

2.2.3 Message Authentication Code und Digitale Signatur

Ein Message Authentication Code (MAC) dient dazu, die Integrität von Daten zu schützen und die Authentizität des Absenders zu überprüfen. Die Sicherheit eines Message Authentication Codes beruht auf einem Schlüssel, über den sowohl der Sender als auch der Empfänger der Daten verfügen. Ein Message Authentication

Code für die Daten *data* wird mit einem Schlüssel *key* mittels dem MAC Algorithmus MAC folgendermaßen berechnet:

$$mac = MAC_{key}(data)$$

Der Message Authentication Code *mac*, der durch diese Berechnung erzeugt wird, ist von einer festen Länge, die üblicherweise zwischen 24 und 96 Bit beträgt. Eine Verifikationsfunktion ermöglicht es, bei Kenntnis des Schlüssels *key*, der Daten *data* und des Message Authentication Code *mac* zu überprüfen, ob die Daten manipuliert wurde. Weiterhin kann überprüft werden, ob der Message Authentication Code *mac* von einem Besitzer des entsprechenden Schlüssels erzeugt wurde. In Kommunikationsprotokollen wird ein Message Authentication Code üblicherweise über den Inhalt einer Nachricht erzeugt und an die Nachricht angehängt. Ein häufig benutzter Algorithmus zur Erzeugung von Message Authentication Codes ist HMAC [56].

Eine weitere Möglichkeit, die Integrität von Daten zu schützen und die Authentizität des Absenders zu überprüfen, ist die *Digitale Signatur*. Dabei wird eine asymmetrische Chiffre zum *Signieren von Daten* verwendet. Dazu wird üblicherweise der Hash-Wert h (siehe Abschnitt 2.2.4) über die zu signierenden Daten gebildet. Dieser Hash-Wert wird mit dem privaten Schlüssel des Absenders signiert. Dies wird geschrieben als

$$signatur = Sig_{key_{priv}}(h)$$

Zur Überprüfung einer Signatur *signatur* berechnet der Empfänger ebenfalls den Hash-Wert h über die signierten Daten. Anschließend wird eine Verifikationsfunktion aufgerufen, die den öffentlichen Schlüssel des Absenders verwendet, um die Signatur zu überprüfen:

$$Ver_{key_{pub}}(signatur, h)$$

Die Verifikationsfunktion ermöglicht es zu überprüfen, ob die Daten, von denen der Hash-Wert gebildet wurde, manipuliert wurden. Weiterhin kann überprüft werden, ob die Signatur von demjenigen erzeugt wurden, der über den zum bekannten öffentlichen Schlüssel gehörenden privaten Schlüssel verfügt.

2.2.4 Hash-Funktionen

Eine *Hash-Funktion* ist eine kollisionsfreie Einwegfunktion, die Nachrichten beliebiger Länge auf einen so genannten *Hash-Wert* fester Länge abbildet. *Einwegfunktionen* sind mathematische Funktionen, die mit relativ geringem Rechen- und Speicheraufwand zu berechnen, aber sehr schwer (also nur mit sehr großem Rechen- und/oder Speicheraufwand) umzukehren sind. Mathematisch genauer formuliert ist eine Einwegfunktion eine Abbildung $hash : A \mapsto B$ einer Menge A in eine Menge B, so dass

$hash(x)$ für jedes Element aus A leicht zu berechnen ist, während es für (fast) jedes Element der Menge B extrem schwer ist, ein Urbild zu finden, d.h. zu $b \in B$ ist die Erfolgswahrscheinlichkeit vernachlässigbar gering, mit polynomialem Aufwand ein $a \in A$ zu finden, so dass $hash(a) = b$. Eine Einwegfunktion heißt kollisionsfrei, wenn es schwer ist, zwei verschiedene Werte a und $\bar{a}$ in der Urbildmenge A zu finden, so dass $hash(a) = hash(\bar{a})$.

Kryptographische Hash-Funktionen weisen zwei wesentliche Eigenschaften auf:

- Es ist schwierig, eine Nachricht zu finden, die zu einem gegebenen Hash-Wert passt (Urbildresistenz).
- Es ist schwierig, zwei Nachrichten zu finden, die denselben Hash-Wert besitzen (Kollisionsresistenz).

Wird aus den Daten *data* mittels der Hash-Funktion $hash()$ ein Hash-Wert h berechnet, so wird dies geschrieben als:

$$h = hash(data)$$

Heute gebräuchliche Hash-Funktionen sind SHA-1 [71] und MD5 [82].

2.2.5 Geheimnisteilung

Ein Verfahren zur *Geheimnisteilung* ermöglicht es, ein Geheimnis wie z. B. einen Schlüssel auf n Instanzen zu verteilen, so dass eine Untermenge der Instanzen, eine so genannte *erlaubte Koalition*, das Geheimnis wiederherstellen kann, während andere Untermengen von Knoten, so genannte *nicht erlaubte Koalitionen*, das Geheimnis nicht wiederherstellen können und es ihnen auch nicht möglich ist, zusätzliche Informationen über das Geheimnis zu gewinnen.

Eine spezielle Klasse der Verfahren zur Geheimnisteilung sind die *(k,n)-Geheimnisteilungsverfahren*. Diese sind für die vorliegende Arbeit von besonderem Interesse. Bei einem (k,n)-Geheimnisteilungsverfahren werden $k \leq n$ beliebige Geheimnisteile benötigt, um das Gesamtgeheimnis zu rekonstruieren, d. h. jede Menge von k Instanzen ist eine erlaubte Koalition. Ein (k,n)-Geheimnisteilungsverfahren schützt gegen Angriffe von weniger als k zusammenarbeitenden Angreifern.

Bei einem *Verifizierenden Geheimnisteilungsverfahren* ist es möglich anhand eines Geheimnisteils zu überprüfen, ob dieses Geheimnisteil aus der Teilung des ursprünglichen Geheimnisses entstanden ist. Damit ist es möglich, manipulierte Geheimnisteile zu erkennen. So wird verhindert, dass ein Geheimnis fehlerhaft rekonstruiert wird, weil ein manipuliertes Geheimnisteil in die Rekonstruktion eingeflossen ist. Dadurch können auch aktive Angriffe (z. B. Manipulation von einzelnen Geheimnisteilen) entdeckt werden.

Einen Überblick über Verfahren zur Geheimnisteilung geben [15, 92], eine ausführliche Bibliographie zum Thema Geheimnisteilung findet sich unter [100].

2.2.6 Vertrauensanker

Der Begriff *Vertrauensanker* ist oft im Umfeld von Vertrauensmodellen [20] und Public-Key-Infrastrukturen zu lesen. Dort bezeichnet ein Vertrauensanker eine vertraute Instanz, von der jegliche Vertrauensprüfung ausgeht, z. B. indem ein Pfad von Zertifikaten zwischen einer zu prüfenden Instanz und der vertrauten Instanz aufgebaut wird. Ein Vertrauensanker wird in diesem Umfeld oft durch ein Root-Certificate-Authority-Zertifikat dargestellt. Ein solcher Vertrauensanker ist z. B. das Zertifikat der Firma VeriSign, das standardmäßig in jedem Firefox-Webbrowser eingebaut ist.

Im Kontext dieser Arbeit wird Vertrauensanker allerdings in einem weiteren Sinn und insbesondere losgelöst von Public-Key-Infrastrukturen betrachtet: In dieser Arbeit bezeichnet ein Vertrauensanker eine Instanz, der alle Knoten im Netz vertrauen.

2.2.7 Dynamische Schlüsselberechnung

In Client-Server-Architekturen ermöglicht es die dynamische Schlüsselberechnung, dass ein Server über paarweise Schlüssel mit jedem Client verfügt, aber diese Schlüssel nicht speichern muss. Dazu verfügt der Server über einen Hauptschlüssel key_{master}. Jeder Client erhält über sichere Kommunikation (z. B. eine wie auch immer geartete Out-of-Band-Kommunikationsmethode) vom Server eine ID zugewiesen. Diese ID verschlüsselt der Server mit seinem Hauptschlüssel:

$$c = E_{key_{master}}(ID)$$

Das Chiffrat c übergibt der Server dem Client. Dieses Chiffrat wird im Weiteren als gemeinsames Geheimnis $key_{Client,Server}$ zwischen dem Client und dem Server verwendet. Nach dieser Berechnung verwirft der Server die jeweiligen Schlüssel, um seine Ressourcen zu schonen. Will nun ein Client zu einem späteren Zeitpunkt mit einem Server kommunizieren, so verfügt der Client über den Schlüssel $key_{Client,Server}$, nicht aber der Server. Deshalb sendet der Client an den Server die ID im Klartext. Der Server kann mittels Verschlüsselung wie oben beschreiben den Schlüssel $key_{Client,Server}$ berechnen und mit dem Client kommunizieren. Ein Angreifer hat durch die abgehörte ID keine Möglichkeit, den Schlüssel herzuleiten, so lange es ihm nicht gelingt, die Verschlüsselung des Servers zu brechen.

Darüber hinaus kann die dynamische Schlüsselberechnung nicht nur eingesetzt werden, um Ressourcen zu sparen, sondern es ist damit zu einem gewissen Grad möglich, digitale Zertifikate mit symmetrischer Verschlüsselung nachzubilden: Dazu verwendet der Server keine zufällige ID, sondern er bildet die ID aus einer Anzahl von Attributen, die er einem Client zertifizieren möchte. Seinen diese Attribute z. B. A_1 und A_2. Dann kann die ID folgendermaßen gebildet werden:

$$ID = hash(A_1|A_2)$$

wobei | die Konkatenation von Werten bedeutet. Die Hash-Funktion $hash()$ kommt nur zum Einsatz, damit die ID immer die gleiche Länge hat und somit die Berechnung des Schlüssels nicht zu aufwendig wird. Der Server teilt dem Client die Werte der Attribute mit und leitet wie oben beschrieben den Schlüssel $key_{Client,Server}$ her. Zur Kommunikation mit der Server muss der Client nun die Attributwerte angeben. Die Attribute sind also an den Schlüssel gebunden.

2.3 Content Addressable Networks

Content Addressable Network [77], im Folgenden auch kurz CAN genannt, ist ein *Overlay-Netz* das eine *verteilte Hash-Tabelle* realisiert. *CAN dient als Basis des in dieser Arbeit beschriebenen Verfahrens zur sicheren Dienste-Suche in Sensornetzen.* Im Folgenden wird deshalb auf die für das Verständnis dieser Arbeit notwendigen Grundlagen zu CAN eingegangen. Einen ausführliche Diskussion zu CAN, die über das hier Gesagte hinausgeht, findet sich in [62].

Ein *Overlay-Netz* (oft kurz auch Overlay) ist ein „Netz über dem Netz“: Ein Overlay-Netz setzt auf einem bestehenden Netz (z. B. Internet oder ein Sensornetz) auf und bildet oberhalb dieses Netzes ein weiteres, rein logisches, Netz. Das zu Grunde liegende Netz wird deshalb auch oft als Underlay-Netz (bzw. kurz Underlay) bezeichnet. Overlays bieten eine hohe Flexibilität, da sie z. B. keine Rücksicht auf die tatsächlichen physikalischen Gegebenheiten (z. B. Standort der einzelnen Kommunikationsinstanzen) nehmen müssen und das Routing auch nach rein logischen Aspekten gestalten können, wie dies z. B. CAN tut. Dadurch kann die Struktur des Overlay-Netzes frei gewählt werden. CAN setzt dabei einen d-dimensionalen Koordinatenraum in Form eines Torus ein, während andere Overlay-Netze wie z. B. Chord einen logischen Ring verwenden.

Eine *Hash-Tabelle* dient zur Speicherung von (Schlüssel, Wert)-Paaren, d. h. in einer Hash-Tabelle können Werte abgelegt werden, auf die später anhand eines Schlüssels zugegriffen werden kann. Abbildung 2.2 zeigt ein Beispiel. In einer Hash-Tabelle wird jedoch nicht der Schlüssel an sich gespeichert, sondern der Hash-Wert des Schlüssels. Im Beispiel werden die Schlüsse-Wert-Paare (key_1, w_1), (key_2, w_2) und (key_3, w_3) gespeichert und dazu die Hash-Werte h_1, h_2 und h_3 errechnet. An dieser Stelle soll darauf hingewiesen werden, dass die hier erwähnten Schlüssel nichts mit den kryptographischen Schlüsseln zu tun haben, die z. B. in Abschnitt 2.2 vorkamen.

In einer *verteilten Hash-Tabelle* werden, ebenso wie in einer normalen Hash-Tabelle, unter dem Hash-Wert eines Schlüssels Werte abgelegt. Allerdings erfolgt die Speicherung der Werte nicht auf einem einzigen Knoten, sondern verteilt über mehrere Knoten. Dazu wird der Hash-Werteraum aufgeteilt und die Teile werden einzelnen Knoten zugewiesen. Abbildung 2.3 zeigt dies für die oben angeführte Hash-Tabelle: der zum Hash-Wert h_1 gehörende Wert wird auf Knoten 1 abgelegt, die zu den Hash-Werten h_2 und h_3 gehörenden Werte auf dem Knoten 2. Die konkrete Ausprägung einer verteilten Hash-Tabelle regelt, wie die Abbildung eines Schlüssels auf einen Knoten erfolgt, d. h. welcher Knoten welche (Schlüssel,Wert)-Paare speichert.

Hash-Wert	Wert
h_1=hash(key_1)	w_1
h_2=hash(key_2)	w_2
h_3=hash(key_3)	w_3

Abbildung 2.2: Hash-Tabelle mit drei Einträgen

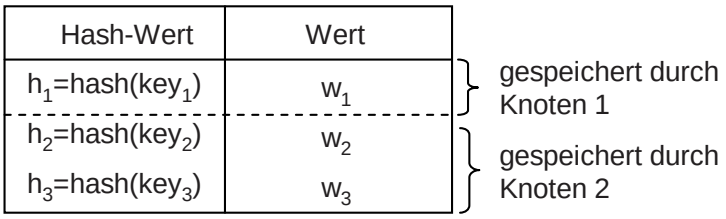


Abbildung 2.3: Auf zwei Knoten verteilte Hash-Tabelle mit drei Einträgen

Die Aufteilung des Hash-Werteraums geschieht in CAN mittels so genannter *Zonen*. CAN interpretiert dazu einen Hash-Wert als Koordinate in einem d-dimensionalen virtuellen Raum. Dies wird in Abbildung 2.4 beispielhaft für einen 2-dimensionalen virtuellen Raum gezeigt, wobei die beiden Dimensionen mit x und y benannt sind. Knoten, die an einem Content Addressable Network teilnehmen, erhalten einen Teil

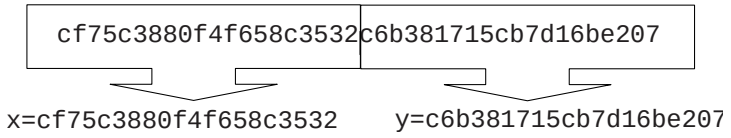
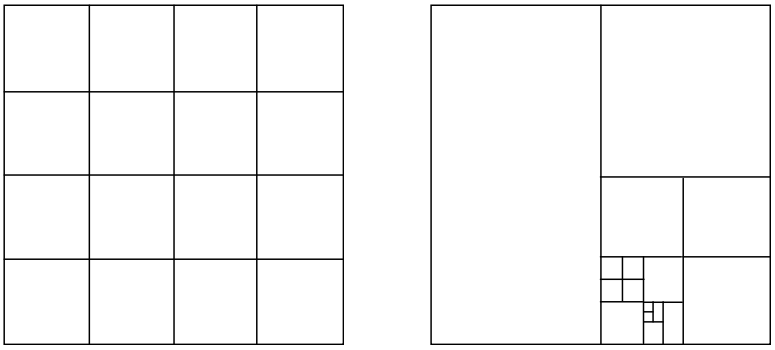


Abbildung 2.4: Interpretation eines Hash-Werts als Koordinate in einem 2-dimensionalen Raum

dieses virtuellen Raums, eben eine der oben erwähnten Zonen, zugewiesen. Abbildung 2.5(a) und Abbildung 2.5(b) zeigen beide einen zweidimensionalen Koordinatenraum, der in 16 Zonen eingeteilt ist. In Abbildung 2.5(a) sind die Zonen alle gleich groß, die Aufteilung ist also gleichmäßig, während in Abbildung 2.5(b) Zonen mit unterschiedlicher Größe existieren. In CAN wird davon ausgegangen, dass der virtuelle d-dimensionale Koordinatenraum die Form eines Torus hat. Im zweidimensionalen Fall (wie in Abbildung 2.5(a) und Abbildung 2.5(b)) entsteht ein Torus aus den in den Abbildungen 2.5(a) und 2.5(b) gezeigten zweidimensionalen Flächen, indem die jeweils gegenüberliegenden Ränder der Flächen miteinander verbunden werden. Für CAN ist nur die Oberfläche dieses Torus von Interesse.

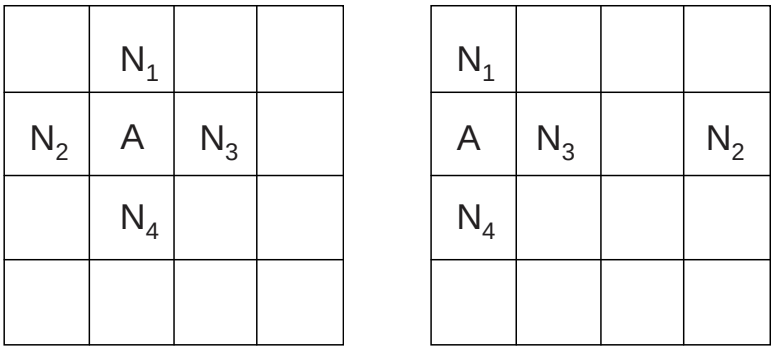
Durch die Lage einer Zone im virtuellen d-dimensionalen Raum (im Folgenden CAN-Raum genannt) werden *Nachbarbeziehungen* definiert: Zwei Zonen sind genau dann



(a) Zonen haben gleiche Größe (b) Zonen haben unterschiedliche Größe

Abbildung 2.5: Zonenaufteilung eines zweidimensionalen CAN-Raums in einem CAN mit 16 Knoten

benachbart, wenn die Koordinaten ihrer Zone in d-1 Dimensionen übereinstimmen und in einer Dimension verschieden sind, aber aneinander angrenzen. Eine Zone hat nach [77] im Durchschnitt $2d$ Nachbarn.



(a) Normalfall (b) Nachbarn am Rand

Abbildung 2.6: Nachbarn einer Zone A

In Abbildung 2.6(a) werden die Nachbarn einer Zone A gezeigt. Zu A benachbart sind die Zonen N_1, N_2, N_3 sowie N_4. Abbildung 2.6(b) zeigt denselben Fall wenn die Zone A am Rand des Koordinatenraums liegt. Die Zone N_2 ist benachbart, da der CAN-Raum die Form eines Torus hat (siehe oben). Auf diesen Nachbarbeziehungen und dem virtuellen d-dimensionalen Raum definiert Content Addressable Network ein *Overlay-Routing-Protokoll*: Die in diesem Routing-Protokoll verwendeten Adressen (z. B. Quelladresse und Zieladresse) sind definiert als Koordinaten eines Punktes im CAN-Raum. Jede Zone unterhält eine Overlay-Routing-Tabelle, in der für die benachbarten Zonen die Vermittlungsschichtadresse eines Zonenverwalters verzeichnet ist sowie der Teil des Koordinatenraums, der von dieser Zone verwaltet werden (angeben durch zwei Eckpunkte, die den von einer Zone verwalteten Raum aufspannen). Um nun eine Nachricht an einen Punkt im CAN-Raum zu versenden, sendet ein Zonenverwalter diese Nachricht an denjenigen Zonenverwalter einer benachbar-

ten Zone, der dem Ziel am nächsten liegt (Greedy-Routing). Dazu überprüft der Zonenverwalter, in welchen Dimensionen die Zielkoordinate nicht mit dem durch die eigene Zone verwalteten Koordinatenraum übereinstimmt. Der Zonenverwalter sendet die Nachricht dann an den Nachbarn, der sich in dieser Dimension in Richtung des Ziels befindet. Abbildung 2.7 verdeutlicht diesen Vorgang im Ausschnitt eines CAN-Raums: eine Nachricht wird aus Zone A an eine Koordinate in der Zone B geschickt.

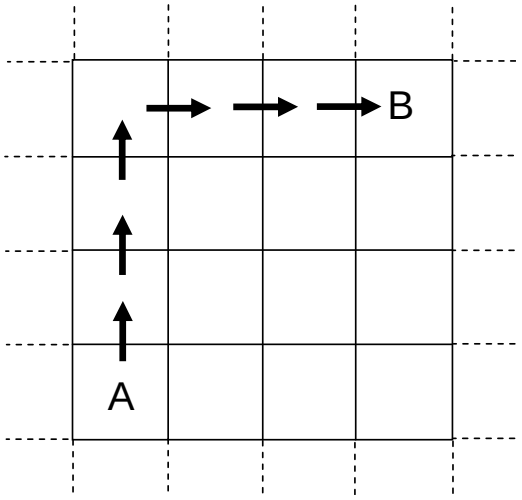


Abbildung 2.7: Routing in einem 2-dimensionalen CAN-Raum (dargestellt ist lediglich ein Ausschnitt des Raums)

Nach [77] beträgt die durchschnittliche Länge eines Routing-Pfades im Overlay, eines so genannten Overlay-Routing-Pfads, für einen d-dimensionalen CAN-Raum, der in n Zonen aufteilt ist

$$\frac{d}{4} * \sqrt[d]{n} \tag{2.1}$$

Overlay-Hops.

Ein Overlay-Hop ist dabei folgendermaßen definiert: Wird eine Nachricht von einem Absender zu einem Empfänger im Overlay gesendet, so wird diese Nachricht von Nachbar zu Nachbar im Overlay weitergeleitet, wie in Abbildung 2.7 zu sehen. Jeden Schritt in dieser Weiterleitung nennt man einen *Overlay-Hop*. Sind also zwei Knoten im Overlay benachbart, so ist die Länge des Overlay-Routing-Pfades zwischen diesen beiden Knoten ein Overlay-Hop. Hierbei ist zu beachten, dass ein *Hop im Overlay mehreren Hops im Underlay entsprechen kann*, da CAN auf die Topologie des darunter liegenden Netzes keine Rücksicht nimmt. Die Definition von Hops im Underlay ist dabei analog zur Definition von Overlay-Hops. Der in Abbildung 2.7 dargestellte Overlay-Routing-Pfad hat eine Länge von 6 Overlay-Hops.

In CAN existiert keine vorgegebene Abbildung der Zonen auf Knoten, die einen Knoten z. B. anhand der Vermittlungsschichtadresse eindeutig einer Zone zuordnen

würde. Vielmehr wird beim Beitritt eines Knotens eine Zone bestimmt, die der Knoten im Weiteren verwaltet. Unter Umständen können während der Lebenszeit eines Knotens noch weitere Zonen hinzukommen. Zum Beitritt in das Content Addressable Network sucht sich der neu hinzukommende Knoten während dem so genannten Bootstrapping einen anderen Knoten, der bereits an CAN teilnimmt. Im Folgenden werden diese Knoten mit „neu hinzukommender Knoten“ respektive „CAN-Mitglied“ bezeichnet. Der neu hinzukommende Knoten wählt einen zufälligen Punkt im CAN-Raum, den so genannten Join Point. Anschließend sendet er eine `JOIN_REQ`-Nachricht im Overlay, die an den Join Point adressiert ist. Diese Nachricht wird mittels dem oben beschriebenen Overlay-Routing-Protokoll solange von Nachbar zu Nachbar weitergereicht, bis die Zone erreicht ist, in welcher der Join Point liegt. Der Knoten, der die Zone verwaltet, in welcher der Join Point liegt, teilt seine Zone in zwei Teile und übergibt dem neu hinzukommenden Knoten eine der beiden Zonenhälften sowie alle Werte, die in dieser Zonenhälfte gespeichert sind. Die Teilung der Zone erfolgt immer in einer vorgegebenen Reihenfolge der Dimensionen, im zweidimensionalen Fall könnte z. B. immer erst in der x-Richtung geteilt werden und bei einem weiteren `JOIN-REQ` in eine der Zonenhälften in der y-Richtung und so weiter. Neben der Zonenhälfte und den Werten, die in dieser Zonenhälfte gespeichert sind, werden auch noch die entsprechenden Nachbareinträge der Overlay-Routing-Tabelle an den neu hinzukommenden Knoten übergeben. Außerdem werden die Nachbarn über die Veränderung informiert, so dass sie ebenfalls ihre Overlay-Routing-Tabellen an den neuen Zustand anpassen können. Abbildung 2.8 zeigt die Teilung einer Zone: Die Zone Z_1 ist zu den Zonen N_1, N_2, N_3 und N_4 benachbart. Nun tritt ein neuer Knoten dem CAN bei und wählt einen Join Point innerhalb der Zone Z_1. Der Zonenverwalter teil entsprechend die Zone Z_1 in zwei neue Zonen (Z_1 und Z_2) und übergibt die Zone Z_2 an den neu hinzukommenden Knoten. Weiterhin benachrichtigt der Zonenverwalter die betroffenen Nachbarn (also N_2, N_3 und N_4) über den neuen Nachbarn, damit diese ihre Overlay-Routing-Tabellen anpassen können, und teilt dem neu hinzukommenden Knoten die Nachbarn der Zone Z_2 mit (also Z_1, N_2, N_3 und N_4). Außerdem überträgt der Zonenverwalter von Z_1 alle (Hash-Wert, Wert)-Paare, die sich jetzt in in der Zone Z_2 befinden, an den neu hinzukommenden Knoten.

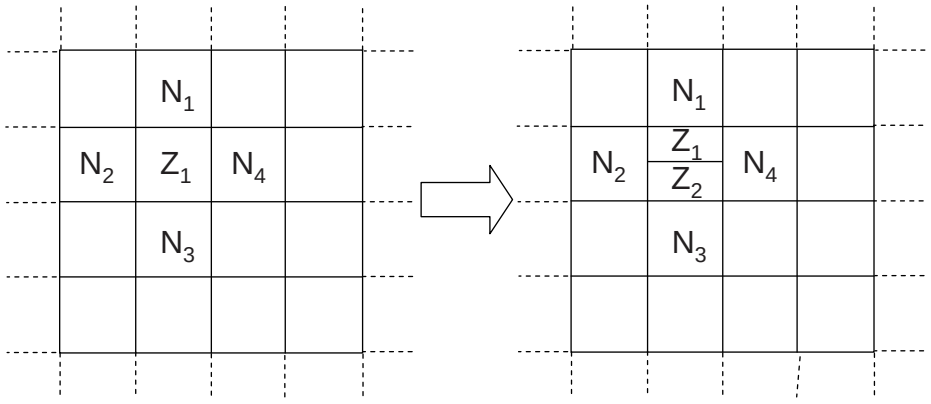


Abbildung 2.8: Teilung einer Zone beim Beitritt eines neuen Knotens

Verlässt ein Knoten das Content Addressable Network, so muss er sicherstellen, dass die von ihm verwaltete Zone in Zukunft von einem anderen Knoten verwaltet wird. Dazu übergibt er die Zone explizit an einen seiner Nachbarn. Er wählt dazu denjenigen Nachbarn aus, der seine Zone mit der eigenen Zone verschmelzen kann, weil die entsprechenden beiden Zonen durch Teilung aus einer einzigen Zone entstanden sind. Ist dies nicht möglich, z. B. weil die entsprechende benachbarte Zone bereits wieder geteilt wurde, wählt der Knoten denjenigen Nachbarn mit der kleinsten Zone aus. Durch dieses Verfahren soll die Last, die ein Knoten mit der Verwaltung des CAN hat, möglichst minimiert werden: kann eine Zone wiedervereinigt werden, so ist es nicht notwendig, zweimal den Aufwand für die Verwaltung einer Zone aufzubringen. Ist dieser Aufwand unvermeidbar, weil keine Vereinigung möglich ist, so wird diejenige Zone gewählt, die am kleinsten ist, da für diese Zone die Wahrscheinlichkeit sehr hoch ist, dass wenige Einträge im entsprechenden Teil der verteilten Hash-Tabelle vorhanden sind.

CAN kann auch mit dem Ausfall von Knoten umgehen. In CAN senden Knoten periodisch `UPDATE`-Nachrichten an ihre Nachbarn, in denen ihre Zonenkoordinaten sowie eine Liste von Nachbarn und deren Zonenkoordinaten enthalten sind. Bleiben `UPDATE`-Nachrichten aus, so gehen die Nachbarn von einem Knotenausfall aus. Sie starten einen Timer, der proportional zu ihrer eigenen Zonengröße ist. Ist der Timer abgelaufen sendet ein Knoten eine `TAKEOVER`-Nachricht an die Nachbarn der ausgefallenen Zone. Diese Nachricht enthält die eigene Zonengröße. Empfängt ein Knoten eine `TAKEOVER`-Nachricht, so vergleicht er die enthaltene Zonengröße mit seiner eigenen Zonengröße. Er stoppt seinen Timer, falls die eigene Zone größer ist. Ansonsten versendet er selbst `TAKEOVER`-Nachrichten an die Nachbarn der ausgefallenen Zone.

Content Addressable Network definiert drei Operationen: Join, Lookup und Insert.

Die *Join-Operation* dient zur Integration eines neuen Knotens in das Content Addressable Network. Sie beinhaltet wie oben beschrieben die Wahl eines Join Points, die Teilung einer Zone, die Übergabe von Werten, den Aufbau der Overlay-Routing-Tabelle im neu hinzukommenden Knoten sowie die Integration des neu hinzukommenden Knotens in die Overlay-Routing-Tabellen der Nachbarn der geteilten Zone.

Die *Insert-Operation* dient in Content Addressable Networks dazu, einen Wert zu speichern. Dazu wird der Operation ein (Schlüssel,Wert)-Paar übergeben. Aus dem Schlüssel wird der Hash-Wert berechnet. Anschließend wird eine `INSERT`-Nachricht erstellt, die im Overlay an den Hash-Wert des Schlüssels adressiert ist, und den übergebenen Wert beinhaltet. Das Overlay-Routing trägt dafür Sorge, dass die `INSERT`-Nachricht denjenigen Knoten erreicht, der die Zone verwaltet, in der der Hash-Wert des Schlüssels liegt. Dort wird unter dem Hash-Wert der Wert gespeichert.

Die *Lookup-Operation* dient dazu, im Content Addressable Network gespeicherte Werte wieder abzufragen. Dazu wird der Operation ein Schlüssel übergeben. Aus diesem wird der Hash-Wert berechnet und im Overlay eine `LOOKUP`-Nachricht an den Hash-Wert verschickt. Diese Nachricht erreicht dank des Overlay-Routings die Zone, in welcher der entsprechende Wert gespeichert ist. Der Verwalter dieser Zo-

ne antwortet mit einer Liste von Werten, die zu dem entsprechenden Hash-Wert gespeichert sind.

Mit diesen drei Operationen kann CAN als Grundlage für ein Verfahren zur sicheren Dienste-Suche in dienstorientierten Sensornetzen verwendet werden: Dazu hinterlegen Dienstanbieter Dienstbeschreibungen mittels der Insert-Operation im CAN. Dabei dient der Dienstname als Schlüssel. Ein Knoten kann dann die Insert-Operation benutzen, um anhand des Dienstnamens Dienstbeschreibungen zu einem Dienst aufzufinden.

3. Sicherheitsbetrachtungen

In diesem Kapitel wird untersucht, welche Angriffe möglich sind, wenn CAN zur Dienste-Suche in drahtlosen Sensornetzen eingesetzt wird. Dazu wird in Abschnitt 3.1 ein Angreifermodell entwickelt, das den weiteren Überlegungen zu Grunde liegt. Anschließend werden in Abschnitt 3.2 verschiedene Angriffe vorgestellt, die es zu verhindern gilt. Schließlich wird in Abschnitt 3.3 ein Überblick über die Leistungsfähigkeit verschiedener Implementierungen kryptographischer Algorithmen gegeben. Auf Basis dieses Überblicks wurde entschieden, welche kryptographischen Bausteine für den Entwurf eingesetzt werden können.

3.1 Angreifermodell

Viele Arbeiten im Bereich der Netzsicherheit verwenden das Dolev-Yao-Angreifermodell [31]: In diesem Modell hat jeder Angreifer *vollständige Kontrolle über die Kommunikation* und kann insbesondere Nachrichten abhören, löschen oder modifizieren. Allerdings ist es einem Angreifer nicht möglich, Geheimnisse, wie z. B. Schlüssel, zu erraten oder die verwendeten kryptographischen Protokolle zu brechen.

Drahtlose Sensornetze bieten jedoch weitere, realistische, Angriffsmöglichkeiten, die im Dolev-Yao-Angreifermodell nicht berücksichtigt werden, weswegen eine Erweiterung dieses Modells notwendig ist. Insbesondere ist es in einem Sensornetz möglich, dass ein Angreifer physikalischen Zugriff auf einzelne Sensorknoten hat, da sich Sensornetze oft im öffentlichen Raum befinden. Diese Annahme über die Fähigkeit eines Angreifers könnte so im Internet nicht getroffen werden, da der Angreifer bei heute im Internet üblichen Angriffen räumlich vom angegriffenen System getrennt ist, so dass kein physikalischer Zugang möglich ist. Bedingt durch den kostensparenden Aufbau eines Sensorknotens ist damit zu rechnen, dass die Sensorknoten nicht besonders gegen Manipulationen gesichert sind (engl. tamper-proof). Hat ein Angreifer also erst einmal physikalischen Zugriff auf einen Sensorknoten bekommen, so ist damit zu rechnen, dass es dem Angreifer möglich ist, diesen Knoten zu übernehmen, siehe z. B. [13]. In [97] wird dies als „Big stick prinicple“ bezeichnet („Whoever

has physical control of the device is allowed to take it over“). Hat ein Angreifer physikalischen Zugriff auf einen Knoten, so ist es ihm möglich, den Speicher eines Sensorknotens auszulesen und so an die Geheimnisse (z. B. Schlüssel) des Sensorknotens zu kommen. Weiterhin ist es einem Angreifer mit physikalischem Zugang auch möglich, den Sensorknoten zu reprogrammieren, also mit einem neuen Programm zu versehen. Ein solcher Art manipulierter Knoten wird als *korrumpierter Knoten* bezeichnet. Wie einfach herkömmliche Sensorknoten zu korrumpieren sind wird z. B. in [13] gezeigt.

Um diesem, im Vergleich zum Angreifer des Dolev-Yao-Angreifermodell, mächtigeren Angreifer gerecht zu werden, wird das Angreifermodell analog zu [26] folgendermaßen erweitert:

- *Aktiver Angreifer:* Von Angreifern korrumpierte Sensorknoten verhalten sich aktiv, d. h. sie versenden Nachrichten und können Nachrichten, die für sie bestimmt sind, entschlüsseln. Ein aktiver Angreifer kann sich auch die meiste Zeit über protokollkonform verhalten und erst ab einem gewissen Zeitpunkt ein bösartiges Verhalten an den Tag legen. Deshalb wird davon ausgegangen, dass es legitimen Knoten nicht möglich ist, zwischen legitimen und korrumpierten Knoten zu unterscheiden.

- *Zusammenarbeitende Angreifer:* Da es einem Angreifer prinzipiell möglich ist, nicht nur einen, sondern eine ganze Menge von Sensorknoten zu übernehmen, wird in dieser Arbeit davon ausgegangen, dass korrumpierte Sensorknoten zusammenarbeiten können, um das Ziel eines Angreifers zu erreichen. Weiterhin wird davon ausgegangen, dass die korrumpierten Sensorknoten untereinander Daten austauschen können. Die daraus resultierende Kommunikation muss nicht zwangsläufig den im Sensornetz verwendeten Protokollmechanismen genügen oder die gleiche Kommunikationsschnittstelle verwenden. So ist es z. B. möglich, dass in einem Sensornetz ZigBee [3] zur Kommunikation eingesetzt wird, korrumpierte Knoten aber über einen der IEEE 802.11 Standards miteinander kommunizieren.

- *Globaler Angreifer:* Zusätzlich zur Zusammenarbeit der korrumpierten Sensorknoten wird davon ausgegangen, dass ein Angreifer nicht auf einen Ort im Sensornetz beschränkt ist, sondern global im Sensornetz agieren kann. Der globale Angreifer kann also z. B. Nachrichten zwischen beliebigen Sensorknoten abhören. Diese Eigenschaft des Angreifers ist der Tatsache geschuldet, dass drahtlose Kommunikation in Sensornetzen relativ einfach abgehört werden kann, was wiederum auf die einfache Bauweise der Sensorknoten und deren relativ einfache Kommunikationstechnologien zurückgeht. Ein Angreifer kann nicht nur Nachrichten abhören, sondern auch neue Nachrichten versenden.

- *Statischer Angreifer:* In dieser Arbeit wird angenommen, dass sich ein Angreifer nicht adaptiv, sondern statisch verhält. Dies bedeutet, dass der Angreifer zufällig und wahllos Sensorknoten des Sensornetzes korrumpiert. Insbesondere

bedeutet dies, dass der Angreifer nicht anhand des bisherigen Protokollverlaufs „wichtige“ Knoten gezielt korrumpiert. Ein Angreifer, der „wichtige“ Knoten global erkennen, und diese auch gezielt übernehmen kann, wäre allmächtig. Kein Sicherheitsprotokoll könnte gegen solch einen Angreifer sinnvoll schützen.

Der Entwurf des Verfahrens zur sicheren Dienste-Suche in Sensornetzen und die Simulation desselben beruhen auf dieser Erweiterung des Dolev-Yao-Angreifermodells. Bevor nun auf Angriffe eingegangen wird, die durch das Verfahren zur sicheren Dienste-Suche verhindert werden sollen, werden hier noch zwei Bezeichnungen für Angreifer eingeführt, welche im Weiteren verwendet werden:

- *Insider-Angreifer:* Als Insider-Angreifer wird ein korrumpierter Sensorknoten bezeichnet, der Mitglied des Verfahrens zur sicheren Dienste-Suche ist. Er gehört also dem Overlay an.
- *Outsider-Angreifer:* Als Outsider-Angreifer wird ein korrumpierter Sensorknoten bezeichnet, der zwar dem Sensornetz angehört, nicht jedoch dem Verfahren zur sicheren Dienste-Suche. Er gehört also nicht dem Overlay-Netz an. Allerdings können über einen Outsider-Angreifer sehr wohl Nachrichten zwischen zum Overlay gehörenden Knoten weitergeleitet werden.

3.2 Betrachtete Angriffe

Im Folgenden werden, basierend auf dem gerade entwickelten Angreifermodell, verschiedene Angriffe auf Overlay-Netze vorgestellt, die verhindert oder zumindest erschwert werden müssen, um ein Overlay-Netz wie CAN zur sicheren Dienstesuche in Sensornetzen einzusetzen. Keiner der Angriffe ist dabei CAN-spezifisch, sondern es handelt sich ausnahmslos um Angriffe, die auf eine Vielzahl von Overlay-Netzen möglich sind.

Dies sind im einzelnen folgende Angriffe:

- Sybil-Angriff
- Churn-Angriff/Rapid-Join-and-Leave Angriff
- Manipulation der Overlay-Routing-Tabellen
- Eclipse-Angriff
- Man-in-the-Middle und Feindliches Overlay
- Angriffe auf im Overlay gespeicherte Werte

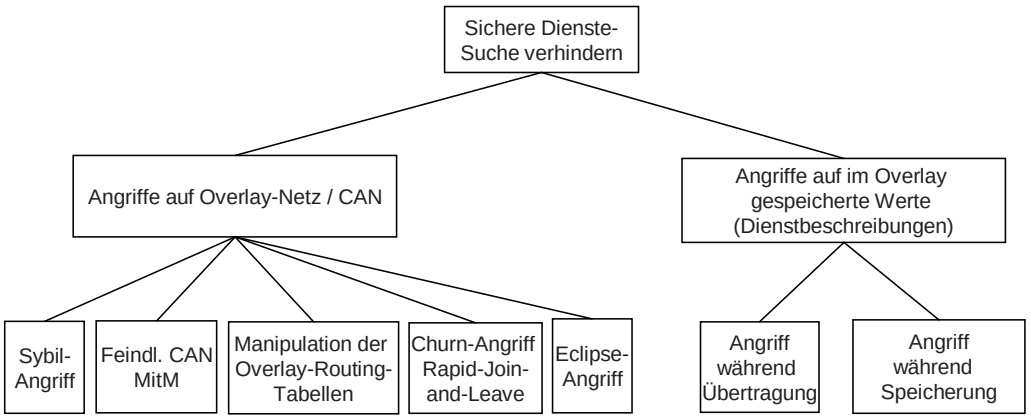


Abbildung 3.1: Angriffsbaum

Abbildung 3.1 zeigt einen Angriffsbaum nach [89, 90], welcher die Angriffe zusammenfasst.

Explizit von der Betrachtung ausgeschlossen sind so genannte Denial-of-Service-Angriffe. Nach [27] liegt ein Denial-of-Service-Angriff vor, wenn es einem legitimierten Nutzer nicht möglich ist, wegen eines vorsätzlichen Angriffs auf eine Ressource zuzugreifen. Dabei können zwei Klassen von Angriffen unterschieden werden: Angriffe auf Basis von Implementierungsfehlern und Angriffe durch Verbrauchen bzw. Auslasten von Ressourcen. Während sich Implementierungsfehler durch sorgfältige Implementierungen, Verifikation und Testen unter Umständen, zumindest zum Teil, verhindern lassen, stellt die zweite Klasse eine wesentlich größere Gefahr für Sensornetze dar. Da Sensornetze drahtlose Kommunikation einsetzen, können z. B. die zur Kommunikation verwendeten Funksignale gestört werden. Man spricht in diesem Fall von *Jamming*. In [113] wird ein Überblick über gebräuchliche Jamming-Methoden gegeben. Als Gegenmaßnahmen schlägt [113] vor, dem Angriff auszuweichen, z. B. durch Verwendung anderer Frequenzbänder oder durch Wegbewegen vom Angreifer. Weiterhin könnten bei einem Angriff leistungsfähigere Fehlerkorrekturverfahren verwendet werden. Verfügt ein Angreifer jedoch über ausreichend Ressourcen, wovon man ausgehen muss, so sind die vorgeschlagenen Verteidigungsstrategien nutzlos. Zusammenfassend kann man sagen, dass sich schon mit relativ geringem Aufwand die Ressource „Funk“ komplett belegen lässt, so dass Kommunikation nur noch in höchst eingeschränktem Maße möglich ist. Ein weiterer Angriff durch Auslastung von Ressourcen ist der *Sleep-Deprivation-Torture-Angriff* [98], bei dem ein Angreifer ständig von einem Sensorknoten angebotene Dienste aufruft, um den Sensorknoten daran zu hindern, in einen Schlafmodus zu wechseln, um Energie zu sparen. So wird der Energievorrat des Sensorknotens schnell erschöpft, was zum Ausfall des Knotens führt. Ein weiterer Denial-of-Service-Angriff ist die physikalische Zerstörung von Sensorknoten. Da Sensornetze oft im öffentlichen Raum installiert sind und Sensorknoten damit auch physikalisch zugänglich sind, ist es einem Angreifer ein leichtes, zumindest einen Teil dieser Sensorknoten zu zerstören.

Zusammenfassend ist zu vermerken, dass Sensornetze sehr anfällig gegen Denial-of-Service-Angriffe sind, insbesondere gegen Jamming, das einfach durchzuführen ist. Da schon gegen diese grundlegenden Angriffe keine wirksamen Verteidigungsstrategien bestehen, werden Denial-of-Service-Angriffe in dieser Arbeit nicht weiter betrachtet.

Im Folgenden sollen nun die eingangs erwähnten Angriffe beschrieben werden.

3.2.1 Sybil-Angriff

Ein grundlegender Angriff auf Overlay-Netze ist der so genannte Sybil-Angriff [32]. Bei einem solchen Angriff tritt ein und derselbe Knoten unter mehreren verschiedenen Identitäten auf. Gelingt es einem Knoten, sich mit jeder dieser Identitäten ins Overlay zu integrieren, so kann er einen großen Einfluss auf das Routing im Overlay ausüben. Mittels eines Sybil-Angriffs kann ein Angreifer also eine Menge von zusammenarbeitenden Insider-Angreifern erzeugen. Dieser Angriff dient als Basis für viele weitere Angriffe, für die zusammenarbeitende Angreifer notwendig sind.

Auf CAN bezogen bedeutet ein Sybil-Angriff, dass ein Angreifer einen Sensorknoten immer wieder über die Join-Operation in CAN integriert. Da der hinzukommende Knoten seinen Join-Point selbst auswählen darf, kann der Angreifer jeweils andere Join-Points wählen, so dass er mit jeder Join-Operation einen Teil des CAN-Raums übernimmt. Abbildung 3.2 verdeutlicht diesen Angriff: der Knoten K_1 führt die Join-Operation 4 mal durch, wählt dabei jeweils den i-ten Join-Point J_i $(1 \leq i \leq 4)$ und übernimmt damit die grau hinterlegten Zonen. In der Abbildung sind die Join-Points durch kleine Sterne symbolisiert. Fortan kann der Angreifer, der eigentlich nur einen einzigen Sensorknoten korrumpiert hat, als Insider-Angreifer das Overlay-Routing immer dann beeinflussen, wenn ein Overlay-Routing-Pfad über eine der vier grauen Zonen führt.

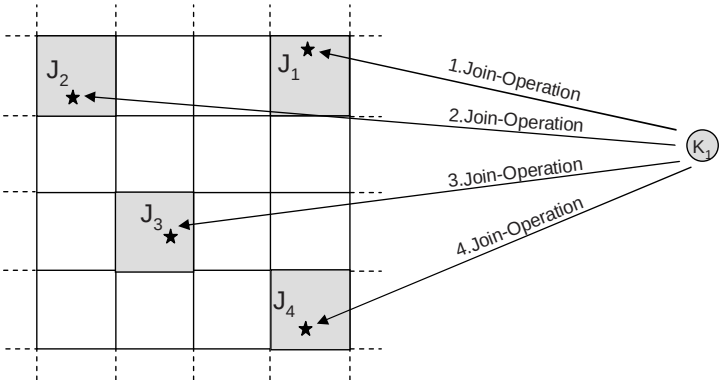


Abbildung 3.2: Sybil-Angriff

3.2.2 Churn-Angriff

Tritt ein Knoten in schneller Frequenz mehrfach einem Overlay bei und verlässt dieses sofort wieder, so spricht man von einem Rapid-Join-and-Leave-Angriff [95]

bzw. einem Churn-Angriff [60]. Dieses Verhalten bewirkt eine Belastung des Netzes, da mit jedem Beitritt eines Knotens zu einem Overlay ein gewisser Kommunikationsaufwand verbunden ist und insbesondere das Verlassen des Overlays weiteren Kommunikationsaufwand auslöst. Zudem kann es zur Störung des Overlay-Routings kommen und Werte, die der Knoten, der das Overlay verlässt, verwaltet, sind unter Umständen verloren. Der Erfolg des Churn-Angriffs hängt maßgeblich damit zusammen, mit welcher Frequenz ein Angreifer dem Overlay beitreten kann.

In CAN bewirkt ein Beitritt beispielsweise, dass eine Zone geteilt wird und eine Zonenhälfte dem neu hinzukommenden Knoten übergeben wird. Die Join-Operation von CAN beinhaltet unter anderem eine Benachrichtigung aller Zonenverwalter der benachbarten Zonen sowie die Übertragung aller in der übergebenen Zonenhälfte liegenden Werte an den neuen hinzukommenden Knoten. Verlässt ein Knoten das CAN, so muss seine Zone von anderen Knoten übernommen werden, was wiederum Kommunikationsaufwand erzeugt. In der Zeit, in der diese Zonenübergabe noch nicht abgeschlossen ist, kann es zu Beeinträchtigung des Overlay-Routings kommen, falls ein Overlay-Routing-Pfad über die verlassene Zone führt. Darüber hinaus sind alle Werte verloren, die in der verlassenen Zone gespeichert wurden.

3.2.3 Manipulation der Overlay-Routing-Tabellen

Knoten verwalten eine Overlay-Routing-Tabelle, die Informationen, wie z. B. die Vermittlungsschichtadresse, über andere Knoten im Overlay beinhaltet und für Weiterleitungsentscheidungen im Overlay herangezogen wird. Durch eine Manipulation dieser Tabellen werden einige weitere Angriffe möglich, z. B. der Eclipse Angriff.

In CAN beinhaltet die Overlay-Routing-Tabelle eines Overlay-Knotens Einträge für die Zonenverwalter der benachbarten Zonen. Eine Manipulation der Routing-Tabelle kann folgendermaßen geschehen:

- Wahl eines „geschickten“ Join-Points: Möchte sich ein Angreifer in der Overlay-Routing-Tabelle eines Opfers platzieren, und ist ihm die vom Opfer verwaltete Zone bekannt, so kann der Angreifer eine Join-Operation ausführen, in deren Ablauf er den Join-Point so wählt, dass er auf jeden Fall eine zum Opfer benachbarte Zone übernimmt. Dadurch wird er ganz legitim durch die Join-Operation in die Overlay-Routing-Tabelle des Opfers integriert.

- Ausnutzen des Reparatur-Algorithmus: Ein Angreifer kann auch den Reparatur-Algorithmus für ausgefallene Zonen verwenden, um eine zum Opfer benachbarte Zone zu übernehmen und so in die Overlay-Routing-Tabelle des Opfers integriert zu werden. Da im Reparatur-Algorithmus basierend auf einer vom Angreifer angegebenen Größe entschieden wird, ob der Angreifer die zum Opfer benachbarte Zone übernimmt, kann der Angreifer diese Größe so wählen, dass er auf jeden Fall Zonenverwalter der zum Opfer benachbarten Zone wird.

3.2.4 Eclipse Angriff

Der Eclipse-Angriff [94] ist eine Spezialform des Angriffs zur Manipulation der Overlay-Routing-Tabellen. Eine Menge von zusammenarbeitenden Angreifern versucht dabei, durch Manipulation der Overlay-Routing-Tabelle des Opfers möglichst viele Einträge in der Overlay-Routing-Tabelle einzunehmen. Dadurch wird es dem Opfer mehr und mehr unmöglich, über das Overlay-Routing mit nicht-korrumpierten Knoten zu kommunizieren.

In CAN wird ein Eclipse-Angriff ausgeführt, indem zusammenarbeitende Angreifer versuchen, mit den oben beschriebenen Manipulationsmöglichkeiten alle direkt zum Opfer benachbarten Zonen zu übernehmen. Abbildung 3.3 zeigt einen Eclipse-Angriff: Der Angreifer K_1 hat alle zur Zone Z benachbarten Zonen übernommen. Jedwede Nachricht, die über das Overlay-Routing-Protokoll an Z geschickt wird oder von Z an einen beliebigen anderen Overlay-Knoten gerichtet wird, kann vom Angreifer manipuliert oder abgefangen werden.

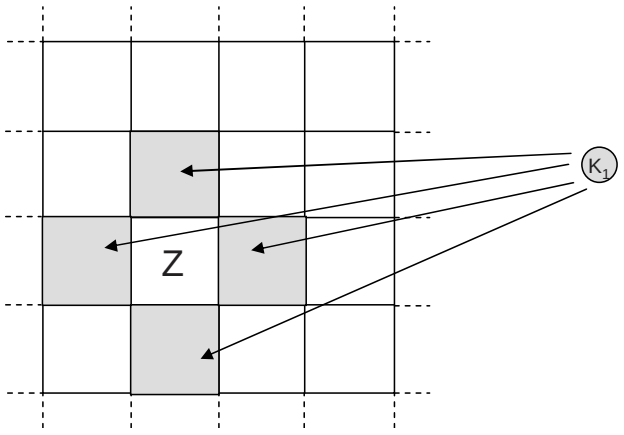


Abbildung 3.3: Eclipse Angriff

3.2.5 Feindliches Overlay und Man in the Middle

Bevor ein Knoten einem Overlay-Netz beitreten kann, muss er während dem Bootstrapping üblicherweise einen oder mehrere bereits im Overlay befindliche Knoten finden. Gelingt es einem Angreifer, einem neu hinzukommenden Knoten vorzuspiegeln, einer dieser Bootstrapping-Knoten zu sein, so kann er dem hinzukommenden Knoten gegenüber vortäuschen, ein Overlay-Netz zu sein. Tritt der hinzukommende Knoten diesem feindlichen Overlay bei, so kann der Angreifer, da er den Rest des vermeintlichen Overlay-Netzes kontrolliert, dem hinzukommenden Knoten beliebige Sachverhalte vortäuschen. Ist es einem Angreifer erst einmal gelungen, einem Knoten vorzutäuschen, ein Overlay-Netz zu sein, so kann er auch als Vermittler, als so genannter „Man in the Middle“, auftreten und Nachrichten des Opfers an ein echtes Overlay weiterleiten bzw. die Antwortnachrichten entsprechend aus dem echten Overlay an das Opfer weiterzugeben. Dieser Angriff wird in allgemeiner Form in [95] vorgestellt.

Im Fall von CAN werden Knoten während dem Bootstrapping durch Broadcast aufgefunden, d. h. es ist einem Angreifer ein leichtes, ein CAN vorzutäuschen. Abbildung 3.4 zeigt, wie in diesem Fall ein „Man in the Middle“ Angriff aussieht: Knoten K_1 ist das Opfer, dem durch den Knoten K_2, dem Angreifer, ein CAN vorgetäuscht wird. K_2 ist Mitglied im CAN. Sendet nun K_1 eine Nachricht, so erhält K_2 diese und kann sie beliebig manipulieren. Dann sendet er die Nachricht über das Overlay-Routing ans Ziel. Entsprechend wird umgekehrt für die Antwortnachricht vorgegangen. Ähnlich wie bei dem Eclipse Angriff ist es durch diesen Angriff also möglich zu kontrollieren, welche Overlay-Nachrichten ein Knoten erhält und welche nicht.

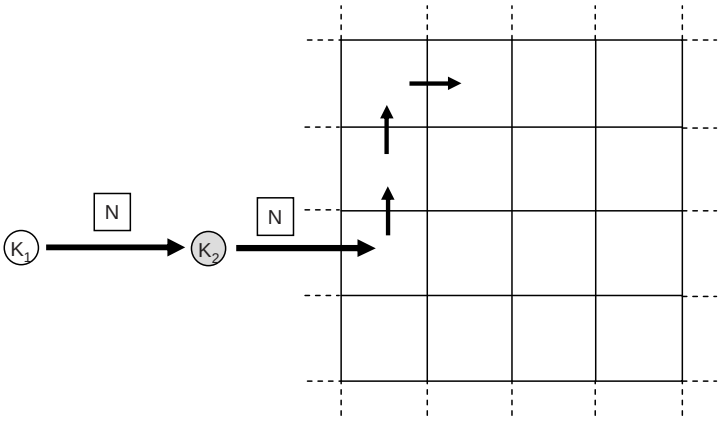


Abbildung 3.4: Man in the Middle

3.2.6 Angriffe auf im Overlay gespeicherte Werte

Wird ein Overlay dazu genutzt um Werte unter einem Schlüssel zu speichern, wie dies z. B. bei verteilten Hash-Tabellen der Fall ist, so mag nicht nur die Struktur des Overlays ein Angriffsziel sein, sondern auch der gespeicherte Wert an sich.

Da CAN als verteiltes Diensteverzeichnis zum Auffinden von Diensten eingesetzt werden soll, sind die Werte, die im Overlay gespeichert werden, in diesem Fall Dienstbeschreibungen. Dienstbeschreibungen beinhalten Informationen über einen Dienst wie z. B. die Vermittlungsschichtadresse des Dienstanbieters. Ein Outsider-Angreifer könnte z. B. diese Vermittlungsschichtadressen manipulieren, um selbst als Dienstanbieter aufzutreten. Er könnte auch Dienstbeschreibungen erstellen und diese einfügen. Dazu genügt es, die Kommunikation zwischen zwei Overlay-Nachbarn zu manipulieren bzw. vorzugeben, einer der Overlay-Nachbarn zu sein. Dies ist möglich, weil weder Authentizität, Integrität noch Vertraulichkeit von Nachricht zwischen im Overlay benachbarten Zonenverwaltern geschützt sind.

Auch Insider-Angreifer können Dienstbeschreibungen manipulieren: Befindet sich ein Insider-Angreifer auf dem Overlay-Pfad zwischen dem Dienstanbieter, der die Dienstbeschreibung im CAN hinterlegen will, und dem speichernden Knoten, so kann der Insider-Angreifer die Nachricht beliebig manipulieren oder unterdrücken, sobald sie über ihn gesendet wird. Weiterhin kann jeder Insider-Angreifer eine Nachricht, die

eine Dienstbeschreibung beinhaltet, erzeugen. Dadurch ist es einem Insider-Angreifer auch möglich, auf eine Dienstsuche-Anfrage mit einer Dienstbeschreibung zu antworten, obwohl er überhaupt nicht für die entsprechende Antwort zuständig ist. Weiterhin kann ein Insider-Angreifer auch Dienstbeschreibungen manipulieren, die er selbst speichert. In CAN bestimmen die Koordinaten einer Zone, welche Dienstbeschreibungen ein Knoten verwaltet. Ein Insider-Angreifer kann einen der Angriffe zur Manipulation der Overlay-Routing-Tabellen nutzen, um gezielt eine Zone zu übernehmen, in der die Dienstbeschreibungen zu einem festgelegten Dienstnamen liegen. Als Zonenverwalter dieser Zone kann er dann alle Dienstbeschreibungen zu dem entsprechenden Dienstnamen manipulieren.

Weiterhin ermöglicht es der in Abschnitt 3.2 beschriebene Rapid-Join-and-Leave-Angriff einem Angreifer, gezielt Dienstbeschreibungen zu verändern. Dazu übernimmt er die Verwaltung einer Zone, manipuliert die darin enthaltenen Daten und verlässt das CAN wieder, indem er die von ihm verwaltete Zone einschließlich der manipulierten Dienstbeschreibungen an einen anderen Knoten übergibt. Anschließend fährt der Angreifer an andere Stelle ebenso weiter fort. Damit kann es einem Angreifer innerhalb kurzer Zeit gelingen, einen Großteil der im CAN gespeicherten Dienstbeschreibungen zu manipulieren. Dieser Angriff wird in allgemeiner Form in [96] beschrieben.

3.3 Eignung kryptographischer Bausteine

Ver- und Entschlüsselung, Digitale Signaturen, Hash-Funktionen und Message Authentication Codes sind kryptographische Bausteine, die oft in sicheren Protokollen eingesetzt werden. Wegen der besonderen Bedingungen in Sensornetzen, insbesondere wegen den beschränkten Ressourcen der eingesetzten Sensorknoten, muss deren Einsatzfähigkeit in Sensornetzen überdacht werden. Für eine Einsatzfähigkeit in Sensornetzen spielen unter anderem die Faktoren Rechenzeit, Energieeffizienz und Speicherverbrauch eine große Rolle. Um eine Bewertung vornehmen zu können, wurden Forschungsarbeiten betrachtet, die eine Aussage über die entsprechenden Bausteine auf für Sensornetzen typischen Hardware-Plattformen vornehmen [37, 38, 59, 64, 75, 109–111]. Die angegebenen Werte beziehen sich immer auf eine konkrete Implementierung der Verfahren. Deshalb wurden aus den Arbeiten lediglich allgemeine Aussage bezüglich Größenordnungen hergeleitet. Anhang C gibt eine vollständige Diskussion der Erkenntnisse wieder. Hier werden die Ergebnisse kurz zusammengefasst:

- *Verschlüsselungs-Algorithmen:* Auf typischen Sensorknoten benötigt die Verschlüsselung eines Bytes zwischen $0.77\mu J$ und $2.9\mu J$ Energie und dauert zwischen $32.5\mu s$ und $287.5\mu s$. Für asymmetrische Verschlüsselung werden um Größenordnungen mehr Energie und deutlich mehr Zeit benötigt. Dies deckt sich mit den Annahmen früher Arbeiten wie z. B. [23, 73, 74].
- *Digitale Signaturen:* Auf typischen Sensorknoten benötigt die Erstellung einer Signatur zwischen $26.96mJ$ und $2725.66mJ$. Die Überprüfung einer Signatur

benötigt zwischen $14.05mJ$ und $144.40mJ$. Dabei dauert die Signaturerstellung zwischen $890ms$ und $12040ms$ und die Signaturüberprüfung zwischen $470ms$ und $4780ms$.

- *Hash-Funktionen:* Die untersuchten Implementierungen von Hash-Funktion haben eine Laufzeit zwischen $427.5\mu s$ und $3812\mu s$ pro Eingabebyte bei einem Energieverbrauch von $5.9\mu J$.
- *Message Authentication Codes:* Pro Byte Eingabe benötigen die Implementierungen zur Erzeugung von Message Authentication Codes ca. $2.5\mu J$ Energie und brauchen für die Berechnung $105\mu s$.

Basierend auf diesen Erkenntnissen wurde für diese Arbeit festgelegt, dass beim Entwurf von Secure Content Addressable Networks auf asymmetrische Verschlüsselung und Digitale Signaturen verzichtet werden soll. Symmetrische Verschlüsselung, Hash-Funktionen und Message Authentication Codes können hingegen eingesetzt werden.

4. Stand der Forschung sichere Dienste-Suche

Die Verwendung einer Dienstabstraktion stellt nicht nur für Sensornetze einen vielversprechenden Ansatz dar, sondern kann auch in vielen anderen Netzen, unter anderem in Ad-hoc-Netzen und dem Internet, benutzt werden. Deshalb haben sich schon einige Arbeiten dem Thema Dienste-Suche gewidmet. In [10, 35, 36, 79, 119] wird ein guter Überblick über Verfahren zur Dienste-Suche gegeben.

Allerdings sind die bisherigen Lösungen – wie sich im Folgenden zeigen wird – für Sensornetze nicht befriedigend. Die Bewertung erfolgte dabei anhand folgender Entwurfsziele, die ein Verfahren zur sicheren Dienste-Suche in Sensornetzen erfüllen muss. Diese Entwurfsziele werden später noch weiter verfeinert werden.

- *Funktionalität*: Das Verfahren soll es einem Dienstnutzer ermöglichen, alle gesuchten Dienste aufzufinden, die von den Dienstanbietern im Sensornetz zur Verfügung gestellt werden.
- *Sicherheit:* Es soll einem Angreifer nicht möglich sein, das Ergebnis der Dienste-Suche zu beeinflussen.
- *Effizienz:* Ein Verfahren zur Dienste-Suche soll auf Sensorknoten effizient arbeiten. Insbesondere bedeutet dies, dass asymmetrische Verschlüsselungsverfahren und digitale Signaturen vermieden werden sollten, da heute gebräuchliche Implementierungen dieser Verfahren relativ viel Rechenaufwand auf Sensorknoten erfordern (siehe Kapitel 3.3). Im Folgenden werden asymmetrische Verschlüsselungsverfahren und digitale Signaturen unter dem Oberbegriff Public-Key-Verfahren zusammengefasst.
- *Robustheit:* Ein Verfahren zur Dienste-Suche soll robust sein gegen den Ausfall von Sensorknoten.

- *Skalierbarkeit:* Da Sensornetze aus tausenden von Knoten bestehen können, soll ein Verfahren zur Dienste-Suche mit steigender Knotenanzahl gut skalieren.
- *Dezentralität:* Verfahren zur Dienste-Suche sollen nicht auf jederzeit online-erreichbare zentrale Komponenten angewiesen sein.

Die Verfahren werden nach ihrem Einsatzzweck gegliedert präsentiert. Zuerst werden Infrastruktur-basierte Verfahren vorgestellt, die z. B. im Internet verwendet werden. Daran schließen Single-Hop-Verfahren an, die oft im Umfeld von Ubiquitous Computing eingesetzt werden, z. B. im Kontext von ortsbasierten Diensten. Ausführlich werden Verfahren für drahtlose Multi-Hop Ad-hoc-Netze sowie für Sensornetze vorgestellt.

4.1 Infrastruktur-basierte Verfahren

Die meisten *Infrastruktur-basierte Verfahren* verwenden zur Dienste-Suche zentrale Infrastrukturkomponenten wie z. B. Server. Beispiele für Infrastruktur-basierte Verfahren sind Jini [103], das Service Location Protocol (SLP)[39], UPnP [68], Salutation [79], Ninja Service Discovery Service [29], Splendor [118] und das in [106] vorgestellte Verfahren.

Während viele Sensornetze der ersten Stunde (z. B. das Great Duck Island Projekt [63]) eine Basisstation verwendeten, an die sämtliche Messwerte gesendet wurden, werden in dienstorientierten Sensornetzen nicht zwingend solche zentralen Komponenten eingesetzt. Einerseits ist der Datenfluss in einem dienstorientierten Sensornetz üblicherweise anders, da Sensorknoten untereinander kommunizieren, um die Ziele einer verteilten Anwendung zu erreichen. Andererseits ist bei Benutzung von Basisstationen zu beobachten, dass die Sensorknoten, die sich in der Nähe der Basisstationen befinden, schneller ausfallen können, da deren Energievorrat leichter erschöpft ist. Der reduzierte Energievorrat geht darauf zurück, dass die Knoten in der Nähe der Basisstationen sehr viele Nachrichten weiterleiten müssen, da alle Knoten Nachrichten lediglich an die Basisstationen senden. Dieses Problem wird z. B. in [61] beschrieben.

Es hat sich also herausgestellt, dass zentrale Infrastrukturkomponenten wie Basisstationen und eben auch die oben erwähnten Server das Entwurfsziel „Dezentralität“ verletzen und in dienstorientierten Sensornetzen vielfach nicht ohne weiteres eingesetzt werden können.

Viele der vorgestellten Verfahren verwenden darüber hinaus Multicast, z. B. Splendor. In [24] werden die Schwierigkeiten dargestellt, die Implementierungen von Multicast-Protokollen auf Sensorknoten mit sich bringen. Die Ergebnisse zeigen großen Forschungsbedarf im Bereich Multicast-Protokolle für Sensornetze auf und legen den Schluss nahe, dass Verfahren, die Multicast verwenden, zumindest im Moment, das Entwurfsziel „Effizienz“ und „Funktionalität“ verletzen.

4.2 Single-Hop-Verfahren

Es existieren einige Dienste-Suche-Verfahren, die Dienste in der unmittelbaren Umgebung eines Knotens suchen, also innerhalb eines einzigen Hops. Solche Verfahren eignen sich besonders gut für Szenarien, in denen ortsbasierte Dienste angeboten werden, d. h. Dienste, die nur innerhalb eines eng begrenzten räumlichen Kontextes aufgerufen werden, so z. B. wenn ein Handy und ein Computer miteinander verbunden werden und der Computer einen Dienst zur Übertragung von elektronischen Visitenkarten bietet. Gerade im Bereich Consumer Electronics wird für solche und ähnliche Aufgaben oft Bluetooth verwendet. Obwohl mit Bluetooth auch Multi-Hop-Netze, so genannte Scatternetze, aufgebaut werden können, ermöglicht es das Bluetooth Service Discovery Protocol [19] lediglich, Dienste innerhalb eines einzigen Hops aufzufinden. Ein weiteres Beispiel für ein Single-Hop-Verfahren zur Dienste-Suche ist das Secure Pervasive Discovery Protocol [5]. In dienstorientierten Sensornetzen kann jedoch nicht davon ausgegangen werden, dass auf Dienste immer nur aus dem direkten Umfeld zugegriffen wird, sondern Dienste sollen vielmehr für Sensorknoten aus dem gesamten Netz auffindbar sein. Die beschriebenen Single-Hop-Verfahren verletzten zwar das Entwurfsziel „Funktionalität“ nicht, aber sie schränken den Anwendungsbereich doch sehr stark ein.

4.3 Verfahren für drahtlose Multi-Hop Ad-hoc-Netze

In den letzten Jahren wurden einige Verfahren zur Dienste-Suche für drahtlose Ad-hoc-Netze entwickelt. Da drahtlose Ad-hoc-Netze, zumindest zum Teil, ähnliche Eigenschaften wie drahtlose Sensornetze aufweisen, sind diese Arbeiten von besonderer Relevanz. Insbesondere stellt sich die Frage, ob nicht eines der Verfahren auch in drahtlosen Sensornetzen verwendet werden kann. Die Verfahren zur Dienste-Suche in drahtlosen Ad-hoc-Netzen kann man in folgende Klassen unterteilen:

- Push-basierte Verfahren
- Pull-basierte Verfahren
- Hybride Verfahren
- Verteilte Diensteverzeichnisse

Push-basierte Verfahren zeichnen sich dadurch aus, dass Dienstanbieter ihre Dienste durch Dienstankündigungen periodisch im Netz bekannt machen. Beispiele für Push-basierte Verfahren sind [42, 67]. Um zu erreichen, dass alle Knoten im Netz eine Dienstankündigung erhalten, ist es notwendig, dass eine empfangene Dienstankündigung weitergeleitet wird, bis alle Knoten Kenntnis von der Dienstankündigung haben. Reine Push-basierte Verfahren skalieren aus diesem Grund schlecht mit der

Größe der Netze. Die Entwurfsziele „Skalierbarkeit“ und „Effizienz“ sind damit verletzt.

Pull-basierte Verfahren zeichnen sich dadurch aus, dass ein Dienstnutzer eine Such-Anfrage versendet, die von anderen Knoten weitergegeben wird und schließlich einen Dienstanbieter erreicht, der die Such-Anfrage mit einer Dienst-Nachricht beantwortet, die den Zugriff auf den Dienst ermöglicht. Das bereits oben erwähnte Service Location Protocol [39] bietet, neben einem Infrastruktur-basierten Verfahren, auch ein Pull-Verfahren zur Dienste-Suche an. Auch in [115] wird ein Pull-basiertes Verfahren vorgestellt.

Bei Pull-basierten Verfahren besteht das Problem darin sicherzustellen, dass eine Such-Anfrage auch wirklich einen Dienstanbieter erreicht. Dazu kann das Netz mit Such-Anfragen geflutet werden, was aber in Sensornetzen wegen dem dabei erzeugten Kommunikationsaufwand vermieden werden sollte, insbesondere wenn Such-Anfragen periodisch gesendet werden oder wenn viele Sensoren Such-Anfragen senden. Bei der Verwendung von Fluten ist das Entwurfsziel „Effizienz“ nicht erfüllt. Eine weitere Möglichkeit besteht darin, Such-Anfragen nur über wenige Hops weiterzuleiten. Dieses Vorgehen kann z. B. in Szenarien eingesetzt werden, wenn ortsgebundene Dienste angeboten werden, die nur innerhalb eines eng begrenzten räumlichen Kontextes aufgerufen werden. In diesem Fall wird das Entwurfsziel „Funktionalität“ zwar nicht verletzt, aber der Anwendungsbereich ist stark eingeschränkt. Das Service Location Protocol setzt Multicast ein, um Dienstanbieter zu finden, die auf eine bekannte Multicast-Adresse hören. Multicast wird in Sensornetzen wegen der oben angeführten Schwierigkeiten üblicherweise nicht eingesetzt, weswegen sich das Verfahren nicht für dienstorientierte Sensornetze eignet.

Wegen den Nachteilen von reinen Push- und Pull-basierten Verfahren existieren einige *hybride Verfahren*, welche Techniken von Push- und Pull-basierte Verfahren kombinieren, z. B. Konark [41], SANDMAN [87], das Pervasive Discovery Protocol [22] sowie [108]. In hybriden Verfahren versenden Dienstanbieter Dienstankündigungen, die von anderen Knoten zwischengespeichert werden. Dienstnutzer benutzen eine Such-Anfrage, um nach Diensten zu suchen. Die Such-Anfrage wird weitergegeben, bis ein Knoten erreicht wird, der eine entsprechende Dienstankündigung zwischenspeichert. Die oben erwähnten Verfahren verwenden verschiedene Strategien, welche Knoten Dienstankündigungen zwischenspeichern: So werden bei SANDMAN die Knoten in Cluster eingeteilt, in denen ein, in periodischen Abständen wechselnder, Clusterhead für die Speicherung zuständig ist. Andere Verfahren wie z. B. das Pervasive Discovery Protocol wählen die speichernden Knoten anhand der Leistungsfähigkeit von Knoten, um zu erreichen, dass leistungsfähigere Knoten mehr Aufwand der Dienste-Suche tragen als leistungsschwache Knoten. Weiterhin können auch alle Knoten Dienstankündigungen speichern, aber die Dienstankündigungen werden nur eine gewisse Anzahl von Hops weitergegeben. Hybride Verfahren skalieren üblicherweise mit steigender Knotenanzahl besser als reine Push- oder Pull-basierte Verfahren, da der Kommunikationsaufwand geringer ist. Allerdings ist es schwierig sicherzustellen, dass eine Such-Anfrage einen Dienstanbieter oder einen speichernden Knoten erreicht, weswegen Entwurfsziel „Funktionalität“ nur zum Teil erfüllt ist.

Ein *Diensteverzeichnis* bietet einem Dienstanbieter die Möglichkeit, seine Dienste beim Diensteverzeichnis zu registrieren. Dienstnutzer können verfügbare Dienste beim Diensteverzeichnis erfragen. Im Kontext von Ad-hoc-Netzen werden vor allem verteilte Diensteverzeichnisse betrachtet, z. B. [55, 86, 117]. Man spricht von einem *verteilten Diensteverzeichnis* wenn die Aufgabe eines Diensteverzeichnis von mehreren Knoten in Zusammenarbeit erbracht wird. Ein verteiltes Diensteverzeichnis unterscheidet sich von einem hybriden Verfahren vor allem dadurch, dass die Sensorknoten, die ein verteiltes Diensteverzeichnis bilden, in einer Struktur angeordnet sind und mit Hilfe dieser Struktur die Dienste-Suche organisieren, während bei einem hybriden Verfahren keine solche Struktur existiert und die Sensorknoten die Dienste-Suche unabhängig voneinander erbringen. Außerdem werden die Registrierungs-Nachrichten von Dienstanbietern bei verteilten Diensteverzeichnissen per Unicast direkt an einen Diensteverzeichnisknoten geschickt, während bei hybriden Verfahren dafür oft ein Broadcast verwendet wird.

Bei verteilten Diensteverzeichnissen ist die Frage wesentlich, welche Knoten das Diensteverzeichnis bilden und wie diese Knoten organisiert werden. Die erste Frage, welche Knoten als speichernde Knoten verwendet werden, stellte sich bereits bei den zuvor beschriebenen hybriden Verfahren. Um solche Diensteverzeichnisknoten zu finden, werden z. B. Cluster gebildet, in denen der Clusterhead der Diensteverzeichnisknoten ist. Weiterhin ist von Interesse, wie Daten zwischen den einzelnen Diensteverzeichnisknoten ausgetauscht werden. Dazu können z. B. hierarchische Verfahren oder auch Overlay-Netze eingesetzt werden. Verteilte Diensteverzeichnisse sind für dienstorientierte Sensornetze sehr gut geeignet: Die Aufgabe des Diensteverzeichnisses kann auf mehrere oder alle Knoten verteilt werden, um das Entwurfsziel „Skalierbarkeit“ zu erreichen. Verteilte Diensteverzeichnisse erfordern keine zentralen Komponenten und genügen deshalb der Anforderung „Dezentralität“.

Hybride Verfahren und die eben angesprochenen verteilten Diensteverzeichnisse stellen also vielversprechende Ansätze für Sensornetze dar. Da Sicherheit ein zentrales Entwurfsziel ist, soll hier untersucht werden, welche Verfahren eingesetzt werden und ob sich diese Verfahren für Sensornetze eignen.

Folgende Sicherheits-Aspekte werden in den bisherigen Arbeiten behandelt:

- *Schutz von Integrität und Vertraulichkeit:* Einige Verfahren schützten Integrität und Vertraulichkeit von Nachrichten, die während der Dienste-Suche ausgetauscht werden, so z. B. PrudentExposure [117]. In den Verfahren werden üblicherweise die für den Schutz von Integrität und Vertraulichkeit benötigten Schlüssel mittels Public-Key-Verfahrens ausgetauscht, was in einem hohen Rechenaufwand resultiert und sich deshalb nur sehr begrenzt für Sensornetze eignet. Das Entwurfsziel „Effizienz“ ist also nicht erfüllt.

- *Reputation:* Verfahren wie z. B. [115] setzen Reputationssysteme ein, um andere Knoten, die am System teilnehmen, anhand ihres bisherigen Verhaltens zu bewerten und entsprechend der Bewertung gewisse Aktionen durchzuführen. In

Reputationssystemen werden jedoch üblicherweise Bewertungen über Knoten weitergegeben. Diese Bewertungen sind digital signiert und verletzen, wegen Einsatz eines Public-Key-Verfahrens, deshalb das Entwurfsziel „Effizienz“.

Insbesondere die Verwendung von aufwendigen Public-Key-Verfahren verhindern also den Einsatz von Dienste-Suche-Verfahren aus Ad-hoc-Netzen in Sensornetzen.

4.4 Verfahren für Sensornetze

Für Sensornetze haben sich noch keine Klassen von Verfahren für die Dienste-Suche etabliert, wie dies bei den gerade angesprochenen Ad-hoc-Netzen der Fall ist. Dies liegt unter anderem daran, dass viele Sensornetze auf eine spezielle Aufgabe zugeschnitten wurden, und weil sich noch keine konsistente Sicht auf Sensornetze herausgebildet hat. Verfahren für die Dienste-Suche in Sensornetzen versuchen, den besonderen Eigenschaften von Sensornetzen gerecht zu werden. So konzentriert sich z. B. Directed Diffusion [50] auf die Daten-Orientierung von Sensornetzen. Directed Diffusion kann im weitesten Sinne als Verfahren zur Dienste-Suche gesehen werden, da es einem Sensorknoten die Möglichkeit gibt, eine Datenquelle für vom Knoten spezifizierte Daten zu finden. Das Verfahren ist allerdings speziell auf datenzentrische Sensornetze zugeschnitten. Insbesondere die Integration von Aktoren in ein solches datenzentrisches Sensornetz gestaltet sich schwierig, da eine Ansteuerung von Aktoren nicht vorgesehen ist. Für dienstorientierte Sensornetze eignet sich Directed Diffusion deshalb nicht. Das Entwurfsziel „Funktionalität“ ist zwar erfüllt, das Anwendungsgebiet ist jedoch stark eingeschränkt.

Ein weiterer Ansatz für Sensornetze aus [108] konzentriert sich auf geographische Aspekte von Sensornetzen und geht im Besonderen davon aus, dass ein geographisches Routing-Protokoll auf Vermittlungsschicht eingesetzt wird und alle Knoten Wissen über ihre geographische Position haben. Ein geographisches Routing-Protokoll trifft seine Weiterleitungsentscheidung auf Basis von geographischen Positionen, z. B. GPS-Koordinaten. Das Dienste-Suche-Verfahren aus [108] ermöglicht es, Dienste in der Nähe einer geographischen Position aufzufinden. Für dienstorientierte Sensornetze eignet sich dieser Entwurf nur bedingt, da einerseits nicht davon ausgegangen werden kann, dass dort ein geographisches Routing eingesetzt wird und andererseits auch nicht davon ausgegangen werden kann, dass dem Dienstnutzer die ungefähre Position eines Dienstes bekannt ist. Auch hier ist das Entwurfsziel „Funktionalität“ zwar erfüllt, aber das Anwendungsgebiet des Verfahrens ist stark eingeschränkt.

ServiceCast [57, 102] stellt einen zu [108] ähnlichen Ansatz dar, abstrahiert allerdings von der Verwendung des Zonen-Routing-Protokolls und setzt ein allgemeineres Kontext-Routing voraus, das Nachrichten, basierend auf einem Kontext, transportiert. Allerdings ist es auch für ServiceCast, zumindest um nicht zu viel Kommunikationsaufwand zu verursachen, notwendig zu wissen, in welchem Kontext ein Dienst ausgeführt wird, was nicht für alle Sensornetz-Anwendungen selbstverständlich ist,

deshalb gilt auch hier für das Entwurfsziel „Funktionalität“: erfüllt, aber nur für ein sehr kleines Anwendungsgebiet.

Keines der Dienste-Suche-Verfahren für Sensornetze erfüllt das Entwurfsziel „Sicherheit“, da Sicherheitsaspekte nicht berücksichtigt wurden.

4.5 Überblick über das in dieser Arbeit entworfene sichere, verteilte Diensteverzeichnis

In diesem Abschnitt wird ein Überblick über das in dieser Arbeit entworfenen sichere, verteilte Diensteverzeichnis Secure Content Addressable Network (SCAN) gegeben. SCAN basiert auf CAN. CAN bietet als verteilte Hash-Tabelle die Möglichkeit, unter einem Schlüssel einen Wert abzulegen. Dies soll für die Dienstesuche genutzt werden. Dazu kommen *Dienstbeschreibungen* zum Einsatz, die in der Hash-Tabelle unter einem Dienstnamen hinterlegt werden, d. h. der Name eines Dienstes wird als Schlüssel für die Hash-Tabelle verwendet und die Dienstbeschreibung ist der Wert, der hinterlegt wird. Dabei besteht eine Dienstbeschreibung aus Attributen, denen Werte zugeordnet werden können.

Attribut	Wert
Adresse	243
Raum	364
Genauigkeit	0.5 ℃

Abbildung 4.1: Beispiel einer Dienstbeschreibung mit drei Attributen

Abbildung 4.1 zeigt ein Beispiel: Der Dienst, welcher durch diese Dienstbeschreibung beschrieben wird, wird durch den Knoten mit der Vermittlungsschichtadresse 243 angeboten (Attribut „Adresse“), liefert Werte mit einer Genauigkeit von 0.5 Grad Celsius zurück (Attribut „Genauigkeit“), und befindet sich in Raum 364 (Attribut „Raum“). Allen Dienstbeschreibungen ist gemeinsam, dass sie über das Attribut „Adresse“ verfügen müssen, das die Vermittlungsschichtadresse desjenigen Knotens beinhaltet, unter welcher der beschriebene Dienst erreichbar ist. Da die Vermittlungsschichtadresse zur Adressierung im Underlay verwendet wird, kann somit der Dienstanbieter im Underlay erreicht werden. Alle weiteren Attribute, wie „Raum“ und „Genauigkeit“ im Beispiel, sind anwendungsspezifisch.

Dienstanbieter erstellen Dienstbeschreibungen und veröffentlichen diese mittels der Insert-Operation. Ein *Dienstnutzer* verwendet die Lookup-Operation, um eine Liste von Dienstbeschreibungen zu erhalten, die unter einem bestimmten Dienstnamen abgespeichert wurden. Die Join-Operation wird verwendet, um neue Knoten in das CAN zu integrieren, die fortan für die Speicherung von Dienstbeschreibungen zuständig sind.

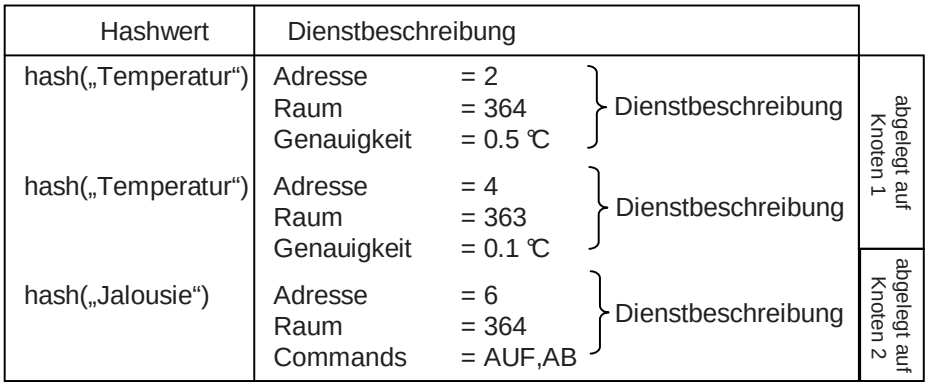


Abbildung 4.2: Dienstbeschreibungen in einer verteilten Hash-Tabelle

Abbildung 4.2 zeigt beispielhaft eine verteilte Hash-Tabelle, die von einem zweidimensionalen CAN implementiert wird: in der Hash-Tabelle sind Dienstbeschreibungen zu zwei Dienste mit dem Namen „Temperatur“ unter dem entsprechenden Hash-Wert hash(„Temperatur“) gespeichert, wobei hash() eine beliebige Hash-Funktion bezeichnet. Weiterhin ist eine Dienstbeschreibung zum Dienst „Jalousie“ vorhanden. Abbildung 4.3 zeigt, wie die Hash-Tabelle aus Abbildung 4.2 realisiert wird: Die Hash-Tabelle ist auf Knoten 1 und Knoten 2 verteilt. Abbildung 4.3 zeigt die Zonen der Knoten 1 und 2, in denen die entsprechenden Dienstbeschreibungen auf Grund ihres Dienstnamens gespeichert werden. Im CAN sind Knoten 1 und Knoten 2 zueinander benachbart, da deren Zonen aneinander angrenzen. In Abbildung 4.3 wird neben der logischen Anordnung der Knoten im CAN auch noch die auf der physikalischen Anordnung basierende Netzwerktopologie gezeigt. Hier wird sichtbar, dass Knoten 1 und Knoten 2 nicht zueinander benachbart sind. Vielmehr erreicht Knoten 1 Knoten 2 z. B. über Knoten 3 und Knoten 7. Nachbarbeziehungen im CAN sind also nicht automatisch auch Nachbarbeziehungen auf Vermittlungsschicht. Schließlich wird in Abbildung 4.3 auch noch einmal die lokale Sicht der einzelnen Knoten auf die verteilte Hash-Tabelle sowie die verteilte Speicherung der Dienstbeschreibungen gezeigt: Knoten 1 speichert alle Dienstbeschreibungen zum Dienst „Temperatur“, während Knoten 2 alle Dienstbeschreibungen zum Dienst „Jalousie“ speichert.

CAN wurde aus folgenden Gründen als Grundlage für ein Verfahren zur sicheren Dienstesuche in Sensornetzen gewählt:

- In CAN bestimmt der Parameter d die Größe der Overlay-Routing-Tabelle, da jeder Knoten im Schnitt $2d$ Nachbarn besitzt. Insbesondere kann der Parameter d frei gewählt werden, d. h. die Größe der Overlay-Routing-Tabelle hängt nicht von der Gesamtanzahl von Knoten im Netz ab, wie dies bei vielen anderen Overlay-Netzen der Fall ist (z. B. bei Chord [101], Pastry [85] oder Tapestry [43]). Dies kommt den begrenzten Ressourcen der Sensorknoten zugute.
- Die relativ starre Struktur des CAN, die durch die Nachbarbeziehungen zwischen den Zonen definiert wird, ist als Ansatzpunkt für die Sicherheitserweite-

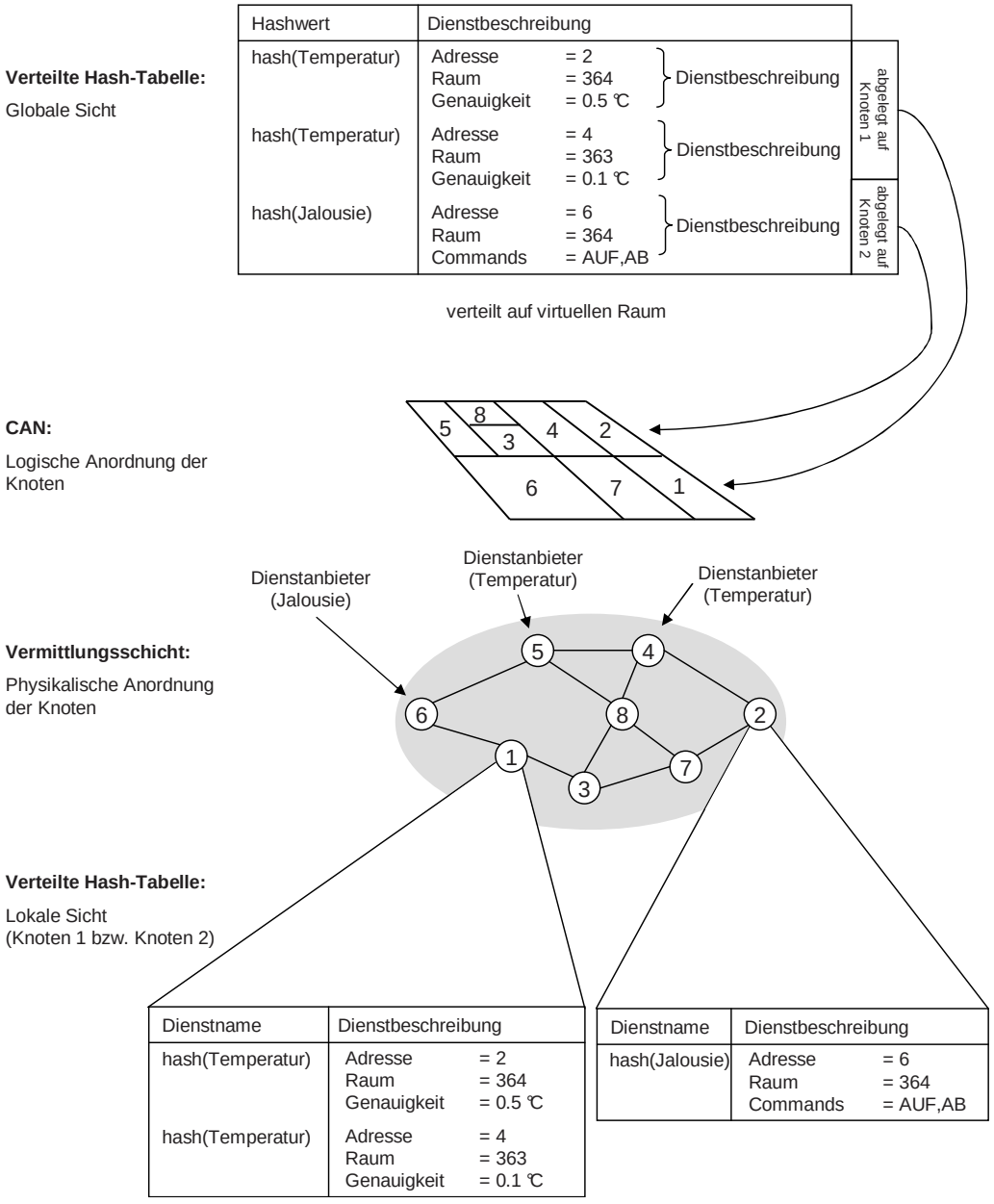


Abbildung 4.3: CAN zur Dienstesuche: Überblick

rungen zu CAN, die im Rahmen dieser Arbeit entwickelt wurden, sehr nützlich. Insbesondere die Möglichkeit, mittels des Overlay-Routings einfach zonendisjunkte Pfade konstruieren zu können, stellte sich als sehr wichtig heraus. Details zu diesem Vorteil von CAN finden sich in Kapitel 5.

- CAN skaliert mit steigender Knotenanzahl gut und erfüllt damit die Anforderung Skalierbarkeit.

- CAN ist dezentral und erfüllt damit die Anforderung Dezentralität.

- CAN ist robust und kann mit dem Ausfall von Knoten umgehen (Anforderung Robustheit).
- CAN eignet sich, alle verfügbaren Dienste mit geringem Aufwand aufzufinden und erfüllt deshalb die Anforderungen Funktionalität und Effizienz.

CAN erfüllt allerdings die Anforderung Sicherheit nicht. SCAN erweitert CAN, so dass auch die Anforderung Sicherheit erfüllt ist. Dazu werden die Knoten, die das verteilte Dienstverzeichnis SCAN bilden, vor dem Beitritt zu SCAN *authentifiziert* und zum Beitritt *autorisiert*. Die *Integrität* und *Vertraulichkeit* von Dienstbeschreibungen wird durch SCAN sowohl während der Übertragung durchs Netz als auch während der Speicherung sichergestellt. SCAN bietet also eine umfassenden Schutz für Dienstbeschreibungen, der über das hinaus geht, was die in den Abschnitten 4.1 bis 4.4 beschriebenen Verfahren leisten. SCAN setzt dabei nur symmetrische kryptographische Verfahren ein und erfüllt damit die Anforderung „Effizienz“. Der Entwurf von SCAN wird im nächsten Kapitel ausführlich beschrieben.

4.6 Zusammenfassung

In diesem Kapitel wurde klar, dass verteilte Diensteverzeichnisse einen vielversprechenden Ansatz für die Dienste-Suche in Sensornetzen darstellen. Obwohl in diesem Bereich schon Vorarbeiten existieren, erfüllt keine alle Entwurfsziele. Insbesondere die Entwurfsziele „Effizienz“ und „Sicherheit“ werden von vielen Verfahren nicht erfüllt. Verfahren, die das Entwurfsziel „Sicherheit“ erfüllen, erfüllen meist nicht gleichzeitig das Entwurfsziel „Effizienz“, da die verwendeten Public-Key-Verfahren für Sensornetze zu aufwendig sind. Tabelle 4.1 fasst den Stand der Forschung noch einmal kurz zusammen.

Es besteht also Bedarf für ein Verfahren zur sicheren Dienste-Suche in Sensornetzen, welches die Entwurfsziele erfüllt und insbesondere nur effiziente, symmetrische Verschlüsselungsverfahren verwendet. SCAN verwendet keinerlei asymmetrischen Verfahren, da diese der Anforderung „Effizienz“ entgegenstehen. Vielmehr beruht SCAN lediglich auf symmetrischen kryptographischen Verfahren und erfüllt somit die Anforderung „Effizienz“. Dabei bietet SCAN einen umfassenden Schutz von Dienstbeschreibungen. SCAN erfüllt alle Entwurfsziele.

	Infrastruktur-basierte Verfahren [29, 39, 68, 79, 103, 106, 118]	Single-Hop-Verfahren [5, 19]	Push-basierte Verfahren [42, 67]	Pull-basierte Verfahren [39, 115]	Hybride Verfahren [22, 41, 87, 108]	Dienste-verzeichnis [55, 86, 117]	Verfahren für Sensornetze [50, 57, 102]	SCAN
Funktionalität	**Nein**, falls Multicast verwendet	**Stark eingeschränkt**	**Ja**	**Nein**,falls Multicast verwendet oder beschränkte Reichweite Such-Anfragen	**Eingeschränkt**, da unklar ob alle Dienstanbieter erreicht werden	**Ja**	**Ja**	**Ja**
Effizienz	**Nein**, falls Multicast verwendet	**Ja**	**Nein**	**Nein**, falls Multicast oder Fluten verwendet	**Ja**	**Ja**, falls keine asymmetrischen kryptographischen Verfahren verwendet werden	**Ja**	**Ja**
Robustheit	**Ja**, falls zentrale Komponenten redundant ausgelegt	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**
Skalierbarkeit	**Ja**, falls Lastverteilung zwischen zentralen Komponenten	**Nein**	**Nein**	**Nein**	**Ja**	**Ja**	**Ja**	**Ja**
Dezentralität	**Nein**	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**	**Ja**
Sicherheit	**Ja**	**Ja**,[19] schützt Integrität und Vertraulichkeit, symmetrische Kryptographie mit vorverteilten Passwörtern	**Nein**	**Ja**,[115] bietet Reputationssystem	**Nein**	**Ja**,[117] bietet Authentifizierung sowie Schutz von Integrität und Vertraulichkeit. Es werden asymmetrische kryptographische Verfahren verwendet ⇒eingeschränkte Effizienz.	**Nein**	**Ja**,Integrität und Vertraulichkeit von Dienstbeschreibungen während Übertragung und Speicherung, Authentifizierung und Autorisation, basiert ausschließlich auf symmetrischer Kryptographie ⇒effizient.

Tabelle 4.1: Stand der Forschung Dienste-Suche

5. Secure Content Addressable Networks: Ein sicheres verteiltes Diensteverzeichnis für dienstorientierte Sensornetze

In Kapitel 3 hat sich gezeigt, dass einige Angriffe auf Content Addressable Networks (CANs) existieren, die einer Anwendung von CAN zur Dienste-Suche in Sensornetzen mit sicherheitskritischen verteilten Anwendungen entgegenstehen. In diesem Kapitel wird der Entwurf von *Secure Content Addressable Networks (SCAN)* [16, 45, 46, 48] vorgestellt. SCAN ist eine sichere Version von CAN zur Dienste-Suche in Sensornetzen. Genauer gesagt realisiert SCAN ein sichere, verteiltes Diensteverzeichnis für Sensornetze.

Zunächst werden in Abschnitt 5.1 die Anforderungen an SCAN sowie die Randbedingungen und Ziele von SCAN vorgestellt. Insbesondere werden die Sicherheitsanforderungen beschrieben. Anschließend wird in Abschnitt 5.2 ein Überblick über SCAN und seine Eigenschaften gegeben. Hier wird auch beschrieben, warum die Angriffe auf CAN in SCAN nicht funktionieren. Die folgenden Abschnitte beschreiben den Entwurf von SCAN im Detail: In Abschnitt 5.3 wird eine Notation für die Beschreibung der Protokolle eingeführt. Daran schließt die Beschreibung der drei Operationen Join (Abschnitt 5.4), Insert (Abschnitt 5.5) und Lookup (Abschnitt 5.6) an. Diese Operationen stellen die Funktionalität von SCAN zur Verfügung: Über die Join-Operation werden neue Knoten in SCAN integriert. Die Insert-Operation dient zur Speicherung von Dienstbeschreibungen unter einem Dienstnamen. Die Dienstbeschreibungen zu einem Dienstnamen werden mittels der Lookup-Operation wieder abgefragt. In Abschnitt 5.7 werden die Schlüsselaustauschprotokolle Single-Path-Key-Exchange und Multi-Path-Key-Exchange vorgestellt. Der folgende Abschnitt 5.8 beschäftigt sich mit den Folgen eines Knotenausfalls. Abschnitt 5.9 erläutert, wie Dienstbeschrei-

bungen während der Speicherung gegen Manipulationen geschützt werden können. Abschnitt 5.10 fasst schließlich die Ergebnisse des Kapitels zusammen.

5.1 Anforderungen, Ziele und Randbedingungen

SCAN wurde entworfen, um das sichere Auffinden von Diensten in drahtlosen Sensornetzen zu ermöglichen, weswegen SCAN folgende Funktionalität bieten soll:

- *Veröffentlichung von Dienstbeschreibungen:* Ein Dienstanbieter soll unter einem Dienstnamen eine Dienstbeschreibung veröffentlichen können. Die Dienstbeschreibung soll Informationen enthalten, die den Zugriff auf den Dienst ermöglichen.

- *Dezentrale Speicherung von Dienstbeschreibungen:* Zur Speicherung der Dienstbeschreibungen sollen keine Server oder andere zentrale Instanzen zum Einsatz kommen. Vielmehr soll die Speicherung verteilt erfolgen, u.a. um einen Flaschenhals im Netz zu verhindern und um höhere Ausfallsicherheit zu erreichen.

- *Suche nach Dienstbeschreibungen:* Ein Dienstnutzer sollen anhand eines Dienstnamens von SCAN eine Liste aller Dienstbeschreibungen vorhandener Dienste mit diesem Namen erhalten. Die Liste soll nicht nur lokal sein, sondern sämtliche im Netz vorhandenen Dienstbeschreibungen erfassen, falls vom Dienstnehmer nicht weitere Einschränkungen gemacht werden. Dadurch sollen insbesondere verteilte, aus Diensten zusammengesetzte, Anwendungen möglich werden, die nicht nur Dienste nutzen, die in ihrer unmittelbaren Umgebung angeboten werden.

Um der zunehmenden Bedeutung von Sicherheit in Sensornetzen gerecht zu werden, waren folgende Sicherheitsziele zentral bei der Entwicklung von SCAN:

- *Authentifizierung und Autorisierung von Knoten vor dem Beitritt:* Knoten, die in ein bestehendes Secure Content Addressable Network (SCAN) integriert werden sollen, müssen authentifiziert und zum Beitritt zum SCAN autorisiert werden.

- *Integrität und Vertraulichkeit von SCAN-Nachrichten:* Die Integrität und Vertraulichkeit von Nachrichten, die mittels des Overlay-Routings im SCAN verschickt werden, sollen geschützt sein, um die Struktur von SCAN aufrecht zu erhalten.

- *Integrität und Vertraulichkeit von Dienstbeschreibungen:* Integrität und Vertraulichkeit von Dienstbeschreibungen sollen während dem Transport durch das Netz und während der Speicherung auf einem Knoten gesichert sein.

- *Authentifizierung des speichernden Knotens:* Jeder Dienstanbieter und Dienstnutzer soll überprüfen können, ob er während der Insert-Operation (Veröffentlichen einer Dienstbeschreibung) bzw. während der Lookup-Operation (Abfragen einer Liste von Dienstbeschreibungen) mit demjenigen Sensorknoten kommuniziert, der die Zone verwaltet, in welcher die entsprechenden Dienstbeschreibungen gespeichert werden. Im allgemeineren Fall soll es jedem Sensorknoten des SCAN möglich sein zu überprüfen, ob er mit einem Sensorknoten im SCAN kommuniziert, der einen bestimmten Punkt im SCAN-Raum verwaltet.

SCAN soll in drahtlosen Sensornetzen eingesetzt werden, weswegen an SCAN folgende Anforderungen gestellt werden:

- *Skalierbarkeit:* Sensornetze bestehen aus einer, aus Sicht von anderen drahtlosen dezentralen Netzen, großen Anzahl von Sensorknoten. In dieser Arbeit wird davon ausgegangen, dass ein Sensornetz aus einigen hundert bis zu tausenden von Knoten bestehen kann. SCAN soll deshalb mit steigender Knotenanzahl gut skalieren, insbesondere im Hinblick auf den Kommunikations- und Speicheraufwand.
- *Effizienz:* Die eingesetzten Verfahren sollen Rücksicht auf die leistungsschwachen Sensorknoten und deren stark beschränkten Ressourcen nehmen. Insbesondere sollen aufwendige asymmetrische Verschlüsselungsverfahren vermieden werden, da diese auf Sensorknoten nur mit unverhältnismäßig hohem Aufwand benutzt werden können.
- *Dezentralität:* Um dem Fehlen einer Infrastruktur Rechnung zu tragen, soll SCAN verteilt arbeiten und auf die Hilfe einer Basisstation oder Ähnlichem verzichten.
- *Robustheit:* SCAN soll robust sein gegenüber dem Ausfall von Diensten und Knoten.

5.2 Überblick über SCAN

SCAN basiert auf dem Overlay CAN und wird zur dezentralen Speicherung von Dienstbeschreibungen verwendet. CAN bietet gute Skalierbarkeit mit steigender Knotenanzahl und funktioniert dezentral.

Eine zentrale Problematik stellt die Authentifizierung und Autorisierung von Knoten dar. Hierzu kommt in SCAN ein spezielles Gerät, das so genannte Master Device, zum Einsatz. Das Master Device dient für das ganze Sensornetz als Vertrauensanker. Es hat, neben der herausragenden Vertrauensstellung, folgende Eigenschaften:

- Das Master Device ist unter ständiger Kontrolle des Benutzers.

- Das Master Device kann mit einem Sensorknoten einen Location Limited Channel aufbauen.

Ein *Location Limited Channel* (LLC) hat nach [11, 98] folgende Eigenschaften:

- *Seitenkanal:* Der Location Limited Channel ist vom eigentlichen Kommunikationsmedium getrennt. So kann z. B. in einem Sensornetz über Funk kommuniziert werden und als Location Limited Channel kommt Kommunikation über Infrarot zum Einsatz.
- *Demonstrative Identification [11]:* Der Location Limited Channel ermöglicht durch seine physikalischen Eigenschaften „Demonstrative Identification", d.h. es ist auf Grund des physikalischen Kontextes klar, wer der Kommunikationspartner ist. Deshalb sind für die Realisierung eines Location Limited Channels Medien gut geeignet, die durch ihre physikalischen Eigenschaften Einschränkungen bei der Übertragungsreichweite unterliegen (z. B. Infrarot, Schall, physikalischer Kontakt) und gerichtet sind (z. B. Infrarot).
- *Authentizität:* Weiterhin ist ein Location-Limited Channel authentisch, d.h. es ist für einen Angreifer nicht, oder nur schwer, möglich, in den Kanal zu senden, bzw. es ist zumindest erkennbar, wann ein Angreifer in den Kanal sendet.

Mit den in [11, 98] beschriebenen Verfahren wird die Integrität und Vertraulichkeit von Nachrichten geschützt, die über den Location Limited Channel übertragen werden. Im Folgenden wird Kommunikation über den Location Limited Channel als sicher angesehen. Der Location Limited Channel wird in SCAN benutzt, um Knoten mittels der Join-Operation sicher ins SCAN zu integrieren. Dazu stößt ein Benutzer die Integration eines neuen Knotens ins SCAN an, indem er einen Location Limited Channel zwischen dem Master Device und dem neuen Knoten aufbaut. Das Master Device kommt zur Authentifizierung und Autorisierung lediglich während dieser Integration eines neuen Knotens (im Rahmen der Join-Operation) in das SCAN zum Einsatz und kommuniziert sonst nie mit dem Sensornetz. Es handelt sich bei dem Master Device also nicht um einen Server oder eine Basisstation, die ständig erreichbar ist. Das Master Device widerspricht also nicht der Anforderung „Dezentrales Netz".

Die Sicherheit von SCAN basiert nun darauf, dass zwischen benachbarten Zonen in SCAN Nachbarschlüssel eingerichtet werden. Diese Nachbarschlüssel werden verwendet, um Nachrichten, die über das Overlay-Routing von Zone zu Zone weitergegeben werden, zu schützen. Dazu wird die Nachricht zum Schutz der Vertraulichkeit verschlüsselt und zum Schutz der Integrität mit einem Message Authentication Code versehen. So können z. B. Dienstbeschreibungen während der Insert-Operation und der Lookup-Operation geschützt werden. Da sich die Nachbarbeziehungen von Zonen nur ändern, wenn eine Zone geteilt wird, also während dem Beitritt eines neuen Knotens während der Join-Operation, wird das Master Device dazu verwendet, den

Knoten die Nachbarschlüssel zuzuweisen, bzw. alte Nachbarschlüssel durch neue zu ersetzen.

Basierend auf den Nachbarschlüsseln wurden die beiden Schlüsselaustauschprotokolle Single-Path-Key-Exchange (SPX) und Multi-Path-Key-Exchange (MPX) entwickelt, die es ermöglichen, Schlüssel zwischen beliebigen Knoten des SCAN, und nicht nur zwischen benachbarten Knoten, einzurichten. Dabei ist das Master Device nicht mehr nötig. SPX und MPX basieren beide darauf, einen Schlüssel oder Teile eines Schlüssels über einen oder mehrere Overlay-Routing-Pfade zu verschicken. Auf Basis von MPX wird das Reparaturverfahren RMPX entwickelt, mit dem der Ausfall einer Zone repariert werden kann. Damit wird Robustheit gegen Knotenausfall erreicht: Mittels RMPX wird ein Knoten ermittelt, der die Zone, die der ausgefallene Knoten verwaltet hatte, übernimmt, und es werden die Nachbarschlüssel zwischen der ausgefallenen Zone und den benachbarten Zonen mittels MPX neu etabliert. RMPX benötigt dazu nicht die Unterstützung des Master Device.

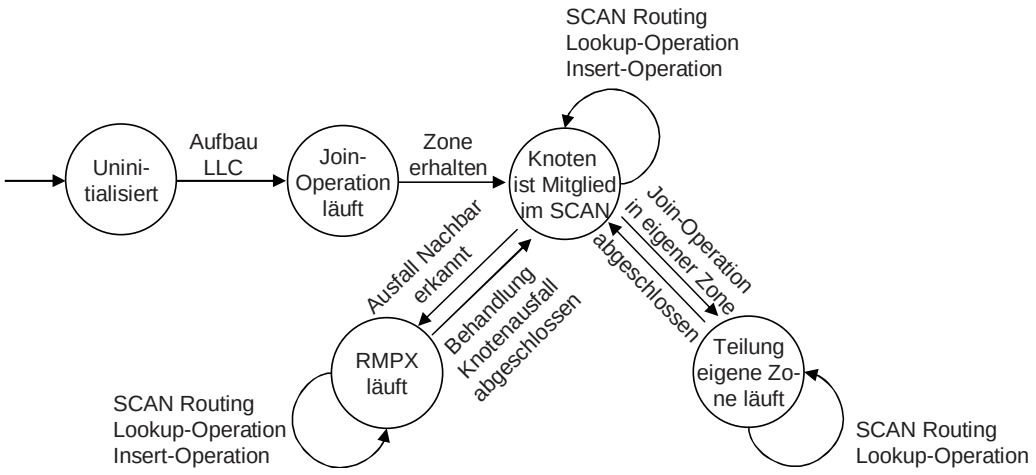


Abbildung 5.1: Protokollautomat von SCAN

Abbildung 5.1 zeigt den Protokollautomaten von SCAN aus der Sicht eines Sensorknotens. Ein Sensorknoten ist zu Beginn uninitialisiert. Durch den Aufbau des Location Limited Channel (LLC) beginnt die Join-Operation, die für den Knoten abgeschlossen ist, sobald der Knoten eine eigene Zone erhalten hat. Der Knoten ist dann Mitglied im SCAN und bearbeitet oder erzeugt in dieser Rolle Nachrichten der Lookup- und Insert-Operation und nimmt am Overlay-Routing von SCAN teil. Fällt ein Nachbar aus, so wird dieser Knotenausfall mit RMPX bearbeitet. Während der Reparatur eines Knotenausfalls nimmt der Knoten an keinen Join-Operationen teil. Sollte von einem Sensorknoten aktuell kein Knotenausfall behandelt werden, so führt eine Join-Operation in die Zone eines Knotens dazu, dass dieser seine Zone teilt. Während dieses Vorgangs werden keine Insert-Operationen für diese Zone bearbeitet. Details zu dem Entwurf der einzelnen Verfahren werden in den folgenden Abschnitten erläutert. Hier werden als Übersicht die Eigenschaften von SCAN zusammengefasst. Wie die einzelnen Eigenschaften durch den Entwurf von SCAN zu

Stande kommen wird dann das Thema der folgenden Abschnitte sein. SCAN hat folgende Eigenschaften:

- *E1 – Authentifizierung:* Sensorknoten werden vor dem Beitritt zu SCAN durch den Benutzer mittels des Master Device authentifiziert.
- *E2 – Autorisierung:* Sensorknoten werden zum Beitritt zu SCAN autorisiert.
- *E3 – Integrität und Vertraulichkeit zwischen Nachbarn:* In SCAN benachbarte Knoten verfügen über einen gemeinsamen Schlüssel, den so genannten Nachbarschlüssel, mit dem sie Integrität und Vertraulichkeit von Nachrichten untereinander schützen.
- *E4 – Integrität und Vertraulichkeit mit dem Master Device:* Jeder Knoten verfügt über einen Schlüssel, mit dem er die Integrität und Vertraulichkeit von Nachrichten zum Master Device schützt.
- *E5 – Integrität und Vertraulichkeit zwischen beliebigen Knoten:* Zwischen beliebigen Knoten können Schlüssel ausgetauscht werden. Diese Schlüssel werden zum Schutz von Integrität und Vertraulichkeit eingesetzt.
- *E6 – Keine Angriffe durch Wiedereinspielen:* Angriffe durch Wiedereinspielen (engl. Replay Attack) werden verhindert. Bei einem Angriff durch Wiedereinspielen hört ein Angreifer Nachrichten ab, die er später wieder einspielt.
- *E7 – Keine gezielte Zonenübernahme:* Sensorknoten können nicht bestimmen, welche Zonen sie bei der Integration ins SCAN übernehmen.
- *E8 – Keine Manipulation der eigenen Zone:* Sensorknoten können die eigene Zone nicht vergrößern.
- *E9 – Zonenverifikation:* Das Master Device kann verifizieren, welche Zone ein Knoten in SCAN verwaltet.
- *E10 – Authentifizierung Zonenverwalter:* Ein beliebiger Sensorknoten, der am SCAN teilnimmt, kann verifizieren, ob ein Knoten eine Zone verwaltet, in der ein gewisser Punkt liegt.
- *E11 – Schutz von Dienstbeschreibungen während der Übertragung:* Integrität und Vertraulichkeit von Dienstbeschreibungen sind während der Übertragung gegen Angriffe von Insider- und Outsider-Angreifern geschützt.
- *E12 – Schutz von Dienstbeschreibungen während der Speicherung:* Integrität und Vertraulichkeit von Dienstbeschreibungen werden während der Speicherung gesichert.

Eigenschaft	Beschrieben in Abschnitt...
E1	5.4
E2	5.4
E3	5.4 und 5.8
E4	5.4 und 5.8
E5	5.7
E6	alle
E7	5.4
E8	5.4 und 5.8
E9	5.4 und 5.8
E10	5.7
E11	5.5, 5.6 und 5.7
E12	5.9

Tabelle 5.1: Übersicht, in welchen Kapiteln beschreiben wird, wie die einzelnen Eigenschaften von SCAN zustande kommen

Mit diesen Eigenschaften erfüllt SCAN also die Sicherheitsziele aus Abschnitt 5.1: „Authentifizierung und Autorisierung von Knoten vor dem Beitritt“ wird durch E1 (Authentifizierung) und E2 (Autorisierung) erfüllt. „Integrität und Vertraulichkeit von SCAN-Nachrichten“ wird durch E3 (Integrität und Vertraulichkeit zwischen Nachbarn), E4 (Integrität und Vertraulichkeit mit dem Master Device) und E5 (Integrität und Vertraulichkeit zwischen beliebigen Knoten) erfüllt. Das Sicherheitsziel „Integrität und Vertraulichkeit von Dienstbeschreibungen“ wird durch E11 (Schutz von Dienstbeschreibungen während der Übertragung) und E12 (Schutz von Dienstbeschreibungen während der Speicherung) erfüllt. „Authentifizierung des speichernden Knotens“ wird durch E9 (Zonenverifikation) und E10 (Authentifizierung Zonenverwalter) erfüllt.

Tabelle 5.1 gibt eine Übersicht, in welchem Abschnitt beschrieben wird, wie die jeweilige Eigenschaft von SCAN zustande kommt.

Mit diesen Eigenschaften von SCAN werden die in Kapitel 3.2 beschriebenen Angriffe verhindert:

- *Verhinderung von Sybil-Angriffen:* Ein Sybil-Angriff setzt voraus, dass es einem Angreifer gelingt, für einen Sensorknoten mehrere Identitäten zu generieren unter denen er im SCAN auftritt. Dies kann einem Angreifer in SCAN nicht gelingen, da die Identität eines Knotens durch den Benutzer authentifiziert wird (E1 – Authentifizierung) und er ansonsten nicht zum Beitritt in das SCAN autorisiert wird (E2 – Autorisierung).

- *Verhinderung von Rapid-Join-and-Leave-Angriffen:* In SCAN wird jede Integration eines neuen Sensorknotens durch den Benutzer angestoßen. Da der Benutzer den Sensorknoten authentifiziert (E1 – Authentifizierung) ist damit

zu rechnen, dass ein Rapid-Join-and-Leave-Angriff auffallen würde. Darüber hinaus ist ein Rapid-Join-and-Leave-Angriff dann am effizientesten, wenn Beitritt und Verlassen des SCAN sehr schnell geschehen. Bedingt durch die nötige Benutzer-Interaktion ist dieser Angriff auf SCAN deshalb nicht praktikabel.

- *Verhinderung von Manipulationen der Overlay-Routing-Tabellen:* Die Integrität und Vertraulichkeit von Nachrichten zwischen Nachbarn in SCAN sind geschützt (E3 – Integrität und Vertraulichkeit zwischen Nachbarn), weswegen es einem Angreifer zur Manipulation der Overlay-Routing Tabelle nicht möglich ist, Nachrichten zwischen Nachbarn zu manipulieren oder neue Nachrichten einzuschleusen. Auch die Join-Operation kann nicht zur Manipulation der Overlay-Routing-Tabelle benutzt werden, da der neu hinzukommende Knoten in SCAN nicht bestimmen kann, wo er in das SCAN integriert wird (E7 – Keine gezielte Zonenübernahme). Auch E8 (Keine Manipulation der eigenen Zone) verhindert eine Manipulation der Overlay-Routing-Tabellen.

- *Verhinderung einer Eclipse Attacke:* Da eine Manipulation der Overlay-Routing-Tabellen nicht möglich ist, kann es einem Angreifer auch nicht gelingen, möglichst viele Plätze in der Overlay-Routing-Tabelle eines Opfers zu besetzen. Verantwortlich sind wieder die Eigenschaften E3 (Integrität und Vertraulichkeit zwischen Nachbarn), E7 (keine gezielte Zonenübernahme) und E8 (keine Manipulation der eigenen Zone)

- *Verhinderung eines feindlichen SCANs:* Da neue Knoten durch das Master Device in das SCAN integriert werden, das Master Device sicher mit jedem Sensorknoten des SCAN kommunizieren kann (E4 – Integrität und Vertraulichkeit mit dem Master Device) und dessen Zonengröße überprüfen kann (E9 – Zonenverifikation), ist es einem Angreifer, der nicht zum SCAN gehört, nicht möglich vorzugeben, ein SCAN zu sein. Ein Angreifer kann entsprechend auch nicht als „Man in the Middle“ auftreten. Insbesondere ist es einem Benutzer bewusst, ob er den ersten Knoten in ein SCAN integriert, also ein neues SCAN eröffnet, oder Knoten in ein bereits vorhandenes SCAN integriert.

- *Verhinderung von Manipulationen der Dienstbeschreibung:* Die Eigenschaften E11 (Schutz von Dienstbeschreibungen während der Übertragung) und E12 (Schutz von Dienstbeschreibungen während der Speicherung) verhindern die Manipulation von Dienstbeschreibungen.

5.3 Notation

Um Protokollabläufe zu beschreiben werden Notationen wie z. B. in [88] verwendet. Allerdings werden durch diese Notation meist Protokolle beschrieben, die sich lediglich auf einer Abstraktionsebene abspielen. Für die vorliegende Arbeit ergibt sich das Problem, dass die Protokolle auf zwei Abstraktionsebenen ablaufen: Einerseits werden Nachrichten über das Vermittlungsschicht-Routing des Sensornetzes versendet, andererseits wird das Overlay-Routing verwendet, um eine Nachricht zu versenden.

Schon die Adressierung ist in beiden Fällen unterschiedlich: Während in der Vermittlungsschicht Vermittlungsschichtadressen zum Einsatz kommen, werden beim Routing im Overlay Punkte im virtuellen Koordinatenraum von SCAN als Adressen verwendet. Natürlich kann das Overlay-Routing auf die entsprechenden Kommunikationsvorgänge auf Vermittlungsschicht abgebildet werden, da jeder Overlay-Hop aus einem oder mehreren Underlay-Hops besteht, allerdings verkompliziert sich dadurch die Darstellung der Protokolle unnötig, weswegen zur Beschreibung der in SCAN verwendeten Protokolle eine Notation zum Einsatz kommt, welche beide Abstraktionsebenen berücksichtigt:

Sendet ein Sender S eine Nachricht mit dem Typ `MsgType` und dem Inhalt Msg an einen Empfänger R, so wird dies folgendermaßen dargestellt:

$$\boxed{\mathrm{S} \overset{\text{Layer}}{\rightarrow} \mathrm{R} : \textbf{MsgType}[\mathrm{Msg}]}$$

Tabelle 5.2 listet beispielsweise Sender und Empfänger von Nachrichten auf, die im Rahmen der Join-Operation vorkommen. *Layer* gibt an, wie eine Nachricht verschickt wird. Mögliche Werte für *Layer* sind OR, NR oder LLC: Einerseits kann eine Nachricht mittels des Overlay-Routings OR versendet werden. In diesem Fall sind die Quell- und Zieladresse jeweils Koordinaten im SCAN-Raum. Anderseits kann eine Nachricht mittels Vermittlungsschicht-Routing versendet werden, was mit NR bezeichnet wird. In diesem Fall werden Vermittlungsschichtadressen verwendet. Außerdem kann ein Sensorknoten auch noch über den Location Limited Channel LLC mit dem Master Device kommunizieren. Hier bezeichnen der Einheitlichkeit wegen S und R jeweils den Knoten und das Master Device die direkt miteinander verbunden sind. Ob Adressen verwendet werden hängt vom konkreten Location Limited Channel ab.

Neben der oben beschriebenen Notation kommen die in der Kryptographie übliche Notationen zum Einsatz, die bereits im Grundlagenkapitel in Abschnitt 2.2 beschrieben wurden. Für die Benennung von Schlüssel, die z. B. zur Verschlüsselung oder für Message Authentication Codes zum Einsatz kommen, wird in dieser Arbeit folgende Konvention verwendet: Schlüssel haben zwei Indizes, welche die Kommunikationspartner bezeichnen, zwischen denen sie verwendet werden. So ist $key_{A,B}$ z. B. der

Name	Beschreibung
MD	Das Master Device
JN	Der neu hinzukommende Sensorknoten (engl. Joining Node)
ZO	Ein Zonenverwalter (engl. Zone Owner)
JP	Zonenverwalter, in dessen Zone der Join Point liegt
N_i	Einer der Nachbarn des ZO

Tabelle 5.2: Überblick über einige in der Join-Operation benutzte Bezeichnungen von Knoten

Name	Beschreibung
$key_{MD,ZO}$	Schlüssel zwischen dem Master Device (MD) und dem Zonenverwalter (ZO)
$key_{MD,JN}$	Schlüssel zwischen Master Device (MD) und dem neu hinzukommenden Knoten (JN)
key_{MD}	Der Master Key des Master Device
key_{grpJN}	Gruppenschlüssel für die Gruppe der Nachbarn von JN

Tabelle 5.3: Überblick über einige in der Join-Operation benutzte Schlüssel

Schlüssel, den Knoten A und Knoten B gemeinsam haben und der für die Kommunikation zwischen A und B verwendet wird. Kommt ein Schlüssel nicht zur Kommunikation zum Einsatz, so wird lediglich ein einziger Index verwendet, welcher den Name des Knotens bezeichnet, der den Schlüssel einsetzt. Schlüssel, die von einer Gruppe von Knoten verwendet werden haben ebenfalls nur einen Index, den Namen der Gruppe. Tabelle 5.3 gibt einen beispielhaften Überblick über einige im Rahmen der Join-Operation verwendete Schlüssel.

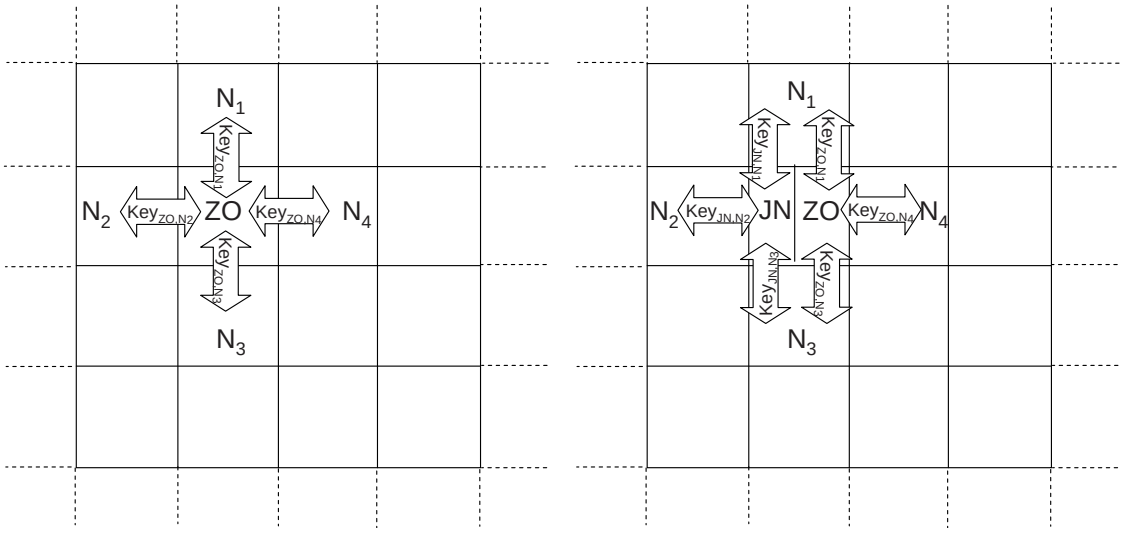
5.4 Die Join-Operation

Die Join-Operation nimmt in SCAN eine wichtige Stellung ein, da sie dazu dient, neue Knoten in das SCAN zu integrieren. Im Rahmen dieser Integration werden zwei wichtige Aufgaben erledigt:

- *Zuteilung einer Zone:* Einem neu hinzukommenden Knoten wird ein Teil des SCAN-Raums, eine so genannte Zone, zugeteilt. Der neu hinzukommende Knoten wird Zonenverwalter für diese Zone und speichert alle Dienstbeschreibungen, die für diese Zone vorgesehen sind.
- *Nachbarschlüssel einrichten:* Eine weitere wichtige Aufgabe während der Integration ist es, zwischen in SCAN benachbarten Knoten symmetrische Schlüssel, so genannte Nachbarschlüssel, einzurichten.

Abbildung 5.2(a) zeigt einen Ausschnitt eines zweidimensionalen SCAN-Raum, der bereits in Zonen aufgeteilt ist. In diesem SCAN-Raum ist der Zonenverwalter ZO zu den Knoten N_1, N_2, N_3 und N_4 benachbart und unterhält mit diesen die Nachbarschlüssel key_{ZO,N_1}, key_{ZO,N_2}, key_{ZO,N_3} und key_{ZO,N_4}. Abbildung 5.2(b) zeigt denselben SCAN-Raum nach Beitritt des Knotens JN: Insbesondere wurden zwischen JN und den Nachbarn die Nachbarschlüssel key_{JN,N_1}, key_{JN,N_2} und key_{JN,N_3} eingerichtet. Der Übersicht halber wurde der Nachbarschlüssel $key_{JN,ZO}$ zwischen JN und ZO, der ebenfalls eingerichtet wurde, in dieser Abbildung weggelassen.

Die Join-Operation teilt sich auf in 4 Phasen, die in Abbildung 5.3 gezeigt werden, wobei die Pfeile jeweils angeben, ab wann Vertraulichkeit und Integrität, abgekürzt als „V+I“ der Kommunikation zwischen den einzelnen Protokollteilnehmern gesichert sind.



(a) vor Integration (b) nach Integration

Abbildung 5.2: Integration eines Knotens JN in das SCAN mittels der Join-Operation

- *Phase 1:* Ein neuer Knoten wird authentifiziert und zum Beitritt ins SCAN autorisiert. Nach Ende dieser Phase ist die Integrität und Vertraulichkeit der Kommunikation zwischen dem Master Device (MD) und dem neu hinzukommenden Knoten (JN) für den restlichen Protokollablauf sichergestellt.

- *Phase 2:* Es wird eine Zone bestimmt, welche später geteilt werden soll, um den Knoten ins SCAN zu integrieren. In dieser Phase wird außerdem der Verwalter dieser Zone ermittelt. Am Ende dieser Phase ist die Integrität und Vertraulichkeit der Kommunikation zwischen dem Master Device und dem Zonenverwalter (ZO) für den restlichen Protokollablauf sichergestellt.

- *Phase 3:* Die Zone aus Phase zwei wird aufgeteilt. Eine Zonenhälfte wird an den neuen Knoten übergeben, während die andere Hälfte beim Zonenverwalter verbleibt. Außerdem werden Dienstbeschreibungen, die in der übergebenen Zone liegen und deshalb fortan vom neuen Knoten verwaltet werden, vom Zonenverwalter zum neu hinzukommenden Knoten übertragen. Nach Abschluss dieser Phase ist die Integrität und Vertraulichkeit der Kommunikation zwischen dem neu hinzukommenden Knoten und dem Zonenverwalter gesichert.

- *Phase 4:* Die Nachbarn der geteilten Zone werden über Änderungen in ihrer Umgebung benachrichtigt. Weiterhin werden Nachbarschlüssel zwischen dem neu hinzugekommenen Knoten und seinen neuen Nachbarn eingerichtet. Nach Abschluss dieser Phase ist die Integrität und Vertraulichkeit der Kommunikation zwischen jedem der Nachbarn und dem hinzukommenden Knoten gesichert. Nach Phase 4 ist der neu hinzukommende Knoten vollständig ins SCAN integriert und die Join-Operation ist beendet.

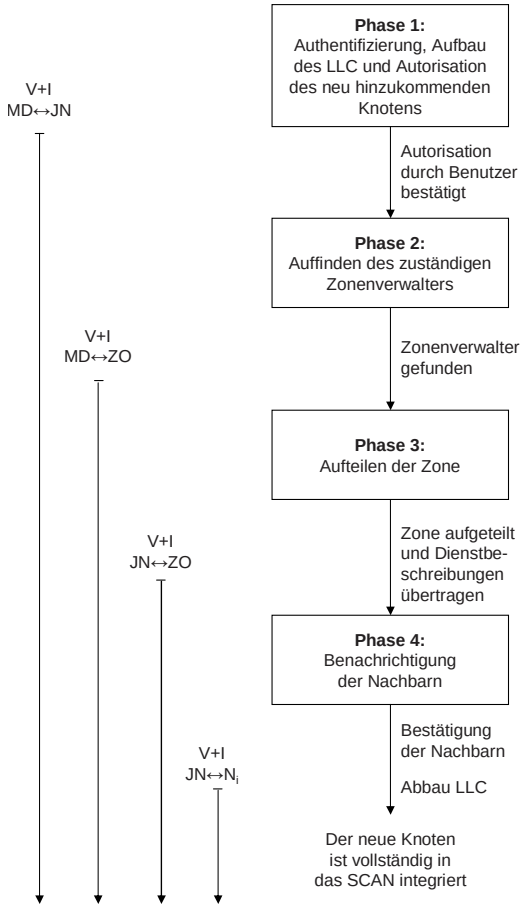


Abbildung 5.3: Überblick über den Ablauf der Join-Operation

Im Folgenden werden die einzelnen *Phasen* der Join-Operation beschrieben, wobei jede *Phase* aus einem oder mehreren *Protokollschritten* besteht. Die Protokollschritte sind phasenübergreifend durchgehend nummeriert, um in späteren Kapiteln problemlos auf die entsprechenden Protokollschritte verweisen zu können. Am Ende dieses Abschnitts erfolgt dann noch eine ausführliche Sicherheitsbetrachtung.

5.4.1 Phase 1: Authentifizierung und Autorisierung des neuen Knotens

Zu Beginn des Integrationsvorgangs authentifiziert der Benutzer den neu hinzukommenden Knoten. Die Authentifizierung nimmt der Benutzer visuell vor, z. B. dadurch, dass er einen fabrikneuen Sensor aus seiner Packung auspackt. Zu diesem Zeitpunkt kann der Benutzer davon ausgehen, dass der Sensorknoten noch vertrauenswürdig ist, d. h. es wird davon ausgegangen, dass ein Sensorknoten zur Zeit der Ausbringung vertrauenswürdig ist. Anschließend baut der Benutzer den Location Limited Channel zwischen dem Master Device und dem neu hinzukommenden Knoten auf. Das Master Device fragt den Benutzer, ob der Knoten zum Beitritt ins SCAN autorisiert werden soll. Falls dies der Fall ist leitet das Master Device den weiteren

Integrationsvorgang ein. Durch die Eigenschaften des Location Limited Channel ist sichergestellt, dass im weiteren Verlauf die Integrität und Vertraulichkeit der Kommunikation zwischen Master Device und neu hinzukommendem Knoten gesichert ist, solange der Location Limited Channel verwendet wird und in der Zwischenzeit nicht abgebaut wurde.

5.4.2 Phase 2: Auffinden des zuständigen Zonenverwalters

In der zweiten Phase der Join-Operation bestimmt das Master Device, welche Zone der neu hinzukommende Knoten JN übernehmen soll. Dazu wählt das Master Device einen Zonenverwalter ZO, dessen Zone im weiteren Verlauf der Join-Operation geteilt wird. Das Master Device nimmt diese Auswahl vor, indem es zufällig eine Koordinate im SCAN-Raum bestimmt, den so genannten Join Point (JP). Dieser Join Point liegt in genau einer Zone. Es ist die Aufgabe der zweiten Phase, zu dieser Zone den zugehörigen Zonenverwalter zu finden. Dazu wird folgendermaßen vorgegangen:

$$\boxed{1.)\text{MD} \overset{\text{LLC}}{\rightarrow} \text{JN} : \textbf{ASSIGN\_JP}[\text{JP}]}$$

Mit der ersten Nachricht des Join-Kommunikationsprotokolls wird JN der JP mitgeteilt. Die `ASSIGN_JP`-Nachricht wird über den (sicheren) Location Limited Channel (LLC) verschickt. Wegen den Eigenschaften des LLC sind Integrität und Vertraulichkeit der Nachricht geschützt.

$$\boxed{2.)\text{MD} \overset{\text{OR}}{\rightarrow} \text{JP} : \textbf{JOIN\_REQ}[\text{addr}_{\text{MD}}]}$$

$$\boxed{3.)\text{ZO} \overset{\text{NR}}{\rightarrow} \text{MD} : \textbf{JOIN\_REP}[zone_{ZO}, E_{key_{MD,ZO}}(uid_{JP}, uid_{grpZO}, NL), mac]}$$

In den Protokollschritten 2 und 3 wird der zum JP gehörige Zonenverwalter ZO ermittelt, indem eine an JP adressierte `JOIN_REQ` Nachricht mittels Overlay-Routing verschickt wird. Das Overlay-Routing stellt sicher, dass diese Nachricht den Zonenverwalter erreicht.

Um Integrität und Vertraulichkeit von Nachrichten, die mittels des Overlay-Routings versendet werden, zu schützen, kommen bei der Übertragung einer Nachricht von einem Knoten aus dem Overlay zu einem seiner Nachbarn im Overlay Verschlüsselung und ein Message Authentication Code zum Einsatz. Beide Verfahren können eingesetzt werden, weil zwischen Nachbarn die Nachbarschlüssel existieren, die für Verschlüsselung und zur Erstellung des Message Authentication Code herangezogen werden. Abbildung 5.4 zeigt beispielhaft, wie eine Nachricht von Knoten A zu Knoten D über die Knoten B und C geschickt wird. Die Overlay-Nachricht

$$\boxed{\text{A} \overset{\text{OR}}{\rightarrow} \text{D} : \textbf{NACHRICHT}[\text{Inhalt}]}$$

zieht auf Vermittlungsschicht folgende Nachrichten nach sich:

$$\boxed{\mathrm{addr_A} \overset{\mathrm{NR}}{\rightarrow} \mathrm{addr_B} : \textbf{NACHRICHT}[\mathrm{E}_{\mathrm{key_{A,B}}}(\mathrm{Inhalt}), \mathrm{mac}]}$$

$$\boxed{\mathrm{addr_B} \overset{\mathrm{NR}}{\rightarrow} \mathrm{addr_C} : \textbf{NACHRICHT}[\mathrm{E}_{\mathrm{key_{B,C}}}(\mathrm{Inhalt}), \mathrm{mac}]}$$

$$\boxed{\mathrm{addr_C} \overset{\mathrm{NR}}{\rightarrow} \mathrm{addr_D} : \textbf{NACHRICHT}[\mathrm{E}_{\mathrm{key_{C,D}}}(\mathrm{Inhalt}), \mathrm{mac}]}$$

Dabei steht $addr_A$ für die Vermittlungsschichtadresse von A, $key_{A,B}$ ist der Nachbarschlüssel zwischen dem Knoten A und dem Knoten B und mac ist der Message Authentication Code. Abbildung 5.4 zeigt diesen Vorgang. Ein Outsider-Angreifer kann die Nachricht zwischen A und D nicht unbemerkt manipulieren oder Informationen über den Nachrichteninhalt erhalten. Als Insider-Angreifer können von allen Knoten des SCAN nur B und C die Nachricht manipulieren oder den Inhalt abhören, also alle auf dem Overlay-Routing-Pfad zwischen A und D liegende Knoten.

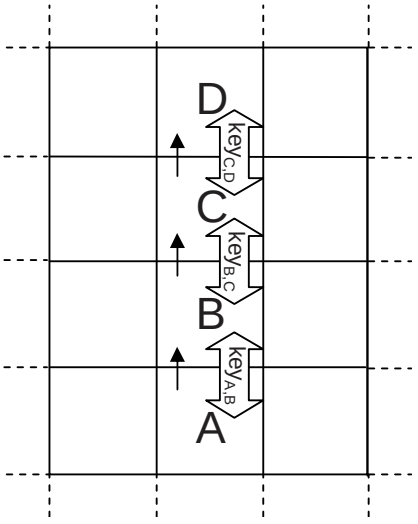


Abbildung 5.4: Beispiel Overlay-Routing

Da die `JOIN_REQ`-Nachricht wie beschrieben mittels Overlay-Routing versendet wird, ist die `JOIN_REQ`-Nachricht gegen Angriffe von Outsider-Angreifern geschützt. Die `JOIN_REQ`-Nachricht enthält die Vermittlungsschichtadresse $addr_{MD}$ des Master Device, weswegen der Zonenverwalter mit einer `JOIN_REP`-Nachricht auf Vermittlungsschicht antworten kann.

Die `JOIN_REP`-Nachricht beinhaltet eine Beschreibung der Zone ($zone_{ZO}$) des Zonenverwalters. Unter anderem beinhaltet diese Zonenbeschreibung die Größe der Zone, beschrieben durch zwei Eckpunkte der Zone, welche die Zone aufspannen. Eine beispielhafte Zonenbeschreibung wird in Tabelle 5.4 gezeigt.

Inhalt	Beschreibung
$X = (x_1, x_2, ..., x_d)$	Der Punkt X im SCAN-Raum bezeichnet die niederwertigste Ecke der Zone.
$Y = (y_1, y_2, ..., y_d)$	Der Punkt Y im SCAN-Raum bezeichnet die höchstwertige Ecke der Zone. Für einen beliebigen Punkt $Z = (z_1, z_2, ..., z_d)$ in der durch diese Beschreibung bestimmten Zone gilt also : $\forall i \in [1, 2, ..., d] : x_i \leq z_i \leq y_i$

Tabelle 5.4: Zonenbeschreibung $zone_A$ eines Knotens A in einem SCAN mit d Dimensionen

Neben der Zonenbeschreibung enthält die `JOIN_REP`-Nachricht in der Nachbarliste NL Informationen über die Nachbarn der Zone. Die Nachbarliste enthält für jeden Nachbarn N_i der Zone eine Vermittlungsschichtadresse $addr_i$ sowie die Beschreibung $zone_i$ der benachbarten Zone. Tabelle 5.5 zeigt eine beispielhafte Nachbarliste. Nach Phase 2 hat das Master Device also Wissen über die zu ZO benachbarten Zonen. Insbesondere kann das Master Device mit allen Nachbarn von ZO kommunizieren, da die entsprechenden Vermittlungsschichtadressen bekannt sind. Abbildung 5.5 zeigt die Sicht, die das Master Device nach Phase 2 auf die Umgebung der Zone des Zonenverwalters ZO hat.

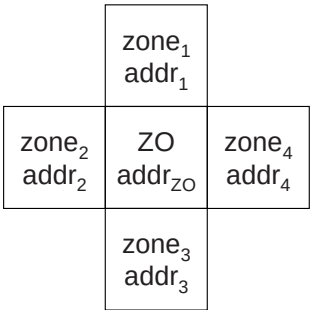


Abbildung 5.5: Sicht des Master Device auf die Umgebung des Zonenverwalters nach Abschluss von Phase 2.

Die Vertraulichkeit der `JOIN_REP`-Nachricht wird durch Verschlüsselung mit dem Schlüssel zwischen dem Master Device und dem Zonenverwalter ($key_{MD,ZO}$) geschützt. Dieser Schlüssel wurde dem Zonenverwalter während dessen Beitritt zum SCAN in einer früheren Join-Operation übergeben. Dabei wurde der Schlüssel durch dynamische Schlüsselberechnung (siehe Kapitel 2.2.7) vom Master Device unter Verwendung dessen Schlüssel key_{MD} erstellt. Der Schlüssel $key_{MD,ZO}$ wurde folgendermaßen berechnet:

$$key_{MD,ZO} = E_{key_{MD}}(hash(zone_{ZO}|addr_{ZO}))$$

Also wurde der Schlüssel an die Zoneninformation $zone_{ZO}$ sowie an die Vermittlungsschichtadresse $addr_{ZO}$ des Zonenverwalters gebunden. Da das Master Device

Inhalt	Beschreibung
$addr_1$	Vermittlungsschichtadresse von Nachbar 1
$zone_1$	Zonenbeschreibung der Zone von Nachbar 1
$addr_2$	Vermittlungsschichtadresse von Nachbar 2
$zone_2$	Zonenbeschreibung der Zone von Nachbar 2
...	...

Tabelle 5.5: Beispielhafte Darstellung einer Nachbarliste

nun die Informationen $zone_{ZO}$ und $addr_{ZO}$ aus der `JOIN_REP`-Nachricht besitzt, kann es den Schlüssel $key_{MD,ZO}$ zur sicheren Kommunikation mit dem Zonenverwalter berechnen. Der Integritätsschutz der `JOIN_REP`-Nachricht erfolgt über einen Message Authentication Code mac mit dem Schlüssel $key_{MD,ZO}$. Nach Überprüfung des mac kann das Master Device nicht nur sicher sein, dass die Integrität der `JOIN_REP`-Nachricht nicht verletzt wurde, sondern das Master Device weiß auch, dass der Zonenverwalter die Zone verwaltet, die in $zone_{ZO}$ beschrieben ist, da der für den Message Authentication Code verwendete Schlüssel wie oben beschrieben an die Zoneninformation gebunden ist, und der Schlüssel in einer vorherigen Join-Operation vom Master Device erstellt wurde.

Weiterhin sind in der `JOIN_REP`-Nachricht zwei Nonces (uid_{JP} und uid_{grpZO}) enthalten, die zum Schutz gegen Angriffe durch Wiedereinspielen dienen. Als Nonce bezeichnet man eine üblicherweise wenige Byte lange Zeichenkette, die nur ein einziges Mal verwendet wird. Nonces kommen in Kommunikations-Protokollen oft zum Einsatz, um Angriffe durch Wiedereinspielen zu verhindern. Im Rahmen von SCAN werden die Nonces folgendermaßen gebildet: Die Nonce uid_{JP} ist der Hash-Wert des Join Points. Da das Master Device den Join Point kennt, kann es überprüfen, ob die empfangene `JOIN_REP`-Nachricht zur aktuellen Join-Operation gehört, und somit kann das Master Device einen Angriff durch Wiedereinspielen erkennen. Die Nonce uid_{grpZO} ist der Hash-Wert des Gruppenschlüssels der Nachbargruppe des Zonenverwalters (siehe unten). Diese Nonce wird vom Master Device nicht überprüft, sondern in den Protokollschritten 5 und 8 verwendet. Der uid_{grpZO}-Wert ändert sich bei jeder erfolgreichen Join-Operation (siehe unten), weswegen sich uid_{grpZO} als Nonce eignet. Da sich JP bei jeder Join-Operation ändert, eignet sich uid_{JP} ebenfalls als Nonce.

5.4.3 Phase 3: Aufteilen der Zone

Ziel der dritten Phase ist es, dem neu hinzukommenden Knoten und dem Zonenverwalter jeweils eine Hälfte der ursprünglichen Zone zuzuweisen, sowie die in der Zone des neu hinzukommenden Knotens gespeicherten Dienstbeschreibungen vom alten Zonenverwalter zum neu hinzukommenden Knoten zu transferieren.

Das Master Device sendet dazu dem neu hinzukommenden Knoten JN respektive dem Zonenverwalter ZO eine `ZONE_SPLIT` Nachricht:

$$4.)\text{MD} \overset{\text{LLC}}{\rightarrow} \text{JN} : \textbf{ZONE\_SPLIT}[uid_{JP}, zone_{JN}, key_{MD,JN}, key_{grpJN}, NKL]$$

Inhalt	Beschreibung
$addr_{N_1}$	Vermittlungsschichtadresse von Nachbar N_1
$zone_{N_1}$	Zoneninformation der Zone von Nachbar N_1
key_{A,N_1}	Schlüssel für die Kommunikation zwischen Knoten A und Nachbar N_1
$addr_{N_2}$	Vermittlungsschichtadresse von Nachbar N_2
$zone_{N_2}$	Zoneninformation der Zone von Nachbar N_2
key_{A,N_2}	Schlüssel für die Kommunikation zwischen Knoten A und Nachbar N_2
...	...

Tabelle 5.6: Beispielhafte Darstellung der Nachbarschlüsselliste (NKL) eines Knotens A

5.)MD $\overset{\text{NR}}{\rightarrow}$ ZO :
ZONE_SPLIT$[E_{key_{MD,ZO}}(uid_{grpZO}, zone_{ZO_{neu}}, key_{MD,ZO}, key_{grpZO}, NKL), mac]$

Die Zonenbeschreibung $zone_{JN}$ der Zone des neu hinzukommenden Knotens sowie seine Vermittlungsschichtadresse $addr_{JN}$ fließen in die Generierung des Schlüssels $key_{MD,JN}$ ein:

$$key_{MD,JN} = E_{key_{MD}}(hash(zone_{JN}|addr_{JN}))$$

Weiterhin erzeugt das Master Device einen Gruppenschlüssel key_{grpJN}, der an JN und, in späteren Schritten, an alle Nachbarn verteilt wird. Außerdem erzeugt das Master Device für jeden Nachbarn N_i von JN einen Nachbarschlüssel key_{JN,N_i}. Diese Schlüssel übergibt das Master Device, zusammen mit den Informationen aus der Nachbarliste aus Phase 2, an JN in der Nachbarschlüsselliste NKL. Der Aufbau einer Nachbarschlüsselliste ist beispielhaft in Tabelle 5.6 zu sehen. Mittels der Nachbarschlüsselliste erstellt der JN seine Overlay-Routing-Tabelle.

Der Inhalt der `ZONE_SPLIT`-Nachricht zwischen dem Master Device und dem Zonenverwalter wird analog gebildet. Da diese Nachricht jedoch über das Vermittlungsschicht-Routing versendet wird, ist die Nachricht verschlüsselt und mit einem Message Authentication Code versehen. In den beiden `ZONE_SPLIT`-Nachrichten werden die Nonces uid_{JP} respektive uid_{grpZO} verwendet, die bereits aus Phase 2 bekannt sind. Die verwendeten Nonces eignen sich aus den bereits in Phase 2 besprochenen Gründen zum Schutz gegen Angriffe durch Wiedereinspielen.

Integrität und Vertraulichkeit der `ZONE_SPLIT`-Nachricht zwischen MD und JN werden über den Location Limited Channel geschützt. Integrität und Vertraulichkeit der `ZONE_SPLIT`-Nachricht zwischen MD und ZO werden durch Verschlüsselung und den Message Authentication Code mac geschützt. Der dazu verwendete Schlüssel $key_{MD,ZO}$ ist dem Master Device aus Phase 2 bekannt.

6.)ZO $\overset{\text{NR}}{\rightarrow}$ JN : **ZONE_TRANSFER**[$\text{E}_{\text{key}_{\text{ZO,JN}}}(\text{SDRs}), \text{mac}$]

7.)JN $\overset{\text{LLC}}{\rightarrow}$ MD : **ZONE_SPLIT_ACK**[]

Der Zonenverwalter hat nun die Möglichkeit, mittels der `ZONE_TRANSFER`-Nachricht Dienstbeschreibungen ($SDRs$), die nicht mehr in seine Zuständigkeit fallen, an den JN zu übergeben.

Integrität und Vertraulichkeit der `ZONE_TRANSFER`-Nachricht werden über den Message Authentication Code mac sowie Verschlüsselung geschützt. Dazu kommt der Nachbarschlüssel $key_{ZO,JN}$ zum Einsatz, der dem ZO und dem JN durch die Nachbarschlüsselliste NKL aus der `ZONE_SPLIT`-Nachricht bekannt ist.

Wurden alle Dienstbeschreibungen übertragen oder ist keine weitere Übertragung gewünscht, so zeigt der neu hinzukommende Knoten dem Master Device durch die `ZONE_SPLIT_ACK`-Nachricht an, dass die Aufteilung der Zone abgeschlossen ist und nunmehr die Nachbarn über die Änderungen informiert werden können. Wegen den Eigenschaften des LLC sind Integrität und Vertraulichkeit der `ZONE_SPLIT_ACK`-Nachricht geschützt.

5.4.4 Phase 4: Benachrichtigung der Nachbarn

Nachdem die eigentliche Übergabe der Zone nach Phase 3 abgeschlossen ist, müssen in der letzten Phase noch die Nachbarn der alten Zone über die Änderungen in ihrer Nachbarschaft informiert werden. Insbesondere müssen den Nachbarn neue Nachbarschlüssel zugewiesen werden, sowie der Gruppenschlüssel aufgefrischt werden. Dies geschieht mittels der `NEIGHBOR_UPDATE`-Nachricht. Ein Nachbar bestätigt den erfolgreichen Abschluss der Integration mittels einer `NEIGHBOR_UPDATE_ACK`-Nachricht.

8.)MD $\overset{\text{NR}}{\rightarrow}$ N_i : **NEIGHBOR_UPDATE**
$[E_{key_{MD,N_i}}(uid_{JP}, uid_{grpZO}, addr_{NN}, zone_{NN}, key_{N_i,NN}, key_{grpNN}), mac]$

9.)N_i $\overset{\text{NR}}{\rightarrow}$ MD : **NEIGHBOR_UPDATE_ACK**[$\text{E}_{\text{key}_{\text{MD,N}_\text{i}}}(\text{uid}_{\text{JP}}), \text{mac}$]

Mit der `NEIGHBOR_UPDATE`-Nachricht teilt das Master Device jedem Nachbarn der geteilten Zone die Adresse $addr_{NN}$ seines neuen Nachbarn (also $addr_{NN} = addr_{ZO}$ oder $addr_{NN} = addr_{JN}$) mit. Außerdem sind in der `NEIGHBOR_UPDATE`-Nachricht die Zonenbeschreibung $zone_{NN}$ für ZO oder JN enthalten, also entweder $zone_{NN} = zone_{ZO}$ oder $zone_{NN} = zone_{JN}$. Darüber hinaus sind noch der entsprechende Nachbarschlüssel $key_{N_i,NN}$ sowie der entsprechende Gruppenschlüssel key_{grpNN} enthalten, also $key_{N_i,NN} = key_{N_i,ZO}$ und $key_{grpNN} = key_{grpZO}$ oder $key_{N_i,NN} = key_{N_i,JN}$ und $key_{grpNN} = key_{grpJN}$.

Die Nachbarn quittieren den Empfang der NEIGHBOR_UPDATE-Nachricht mit einer NEIGHBOR_UPDATE_ACK-Nachricht.

Integrität und Vertraulichkeit sowohl der NEIGHBOR_UPDATE-Nachricht als auch der NEIGHBOR_UPDATE_ACK-Nachricht werden durch den Message Authentication Code *mac* sowie Verschlüsselung gesichert. Die dafür benötigten Schlüssel kann das Master Device mittels dynamischer Schlüsselberechnung (siehe Kapitel 2.2.7) aus den Informationen in der Nachbarliste herleiten (Zoneninformation und Vermittlungsschichtadresse). Diese Nachbarliste wurde in Phase 2 übergeben. Da auch die Zonenbeschreibungen der Nachbarn in die Schlüsselerzeugung einfließen, kann nach Empfang der NEIGHBOR_UPDATE_ACK-Nachricht davon ausgegangen werden, dass die entsprechenden Informationen in der Nachbarliste korrekt waren. Dadurch wird Eigenschaft E8 (Keine Manipulation der eigenen Zone) und E9 (Zonenverifikation) erreicht.

Als Nonces gegen Angriffe durch Wiedereinspielen kommen erneut uid_{JP} sowie uid_{grpZO} zum Einsatz, deren Eignung bereits oben beschrieben wurde. In der NEIGHBOR_UPDATE-Nachricht sind beide Nonces enthalten. Jeder Nachbar kann die uid_{grpZO} überprüfen. Die übergebene uid_{JP} kommt dann für die NEIGHBOR_UPDATE_ACK-Nachricht zum Einsatz. Das Master Device kann diese Nonce überprüfen. Durch den Einsatz der Nonces wird Eigenschaft E6 (Keine Angriffe durch Wiedereinspielen) erreicht.

5.4.5 Timer

Im Rahmen der Join-Operation werden die Timer JOIN_REQ_TIMER, ZONE_SPLIT_TIMER und NEIGHBOR_UPDATE_TIMER verwendet, um das Ausbleiben von Antwortnachrichten zu erkennen und somit Übertragungswiederholungen anstoßen zu können.

Das Master Device startet den Timer JOIN_REQ_TIMER mit Versenden der JOIN_REQ-Nachricht und stoppt ihn, wenn die Nachricht JOIN_REP empfangen wird. Der Timer läuft nach der Zeitspanne JOIN_REQ_TIMEOUT ab. In diesem Fall sendet das Master Device die JOIN_REQ-Nachricht erneut und startet den Timer wieder. Das Master Device unternimmt maximal MAX_RETRIES Übertragungswiederholungen. Läuft der Timer danach wieder ab, wählt das Master Device einen neuen Join Point und beginnt die Join-Operation von vorne, da vom Ausfall des Zonenverwalters, in dessen Zone der alte Join Point liegt, ausgegangen wird. Der neu hinzukommende Knoten wird über den Neubeginn der Join-Operation durch Empfang der ASSIGN_JP-Nachricht informiert.

Der ZONE_SPLIT_TIMER wird vom Master Device mit Versenden der ZONE_SPLIT-Nachricht gestartet und mit Erhalt der Nachricht ZONE_SPLIT_ACK gestoppt. Der Timer laufen nach der Zeitdauer ZONE_SPLIT_TIMEOUT ab. In diesem Fall sendet das Master Device die ZONE_SPILT-Nachricht erneut und startet den Timer wieder. Das Master Device wiederholt die Übertragung maximal MAX_RETRIES Versuche. Danach startet das Master Device die Join-Operation wie oben beschrieben erneut.

Der Timer `NEIGHBOR_UPDATE_TIMER` wird für jeden Nachbarn, der über die Join-Operation benachrichtigt wird, vom Master Device mit Versenden der Nachrichten `NEIGHBOR_UPDATE` gestartet und bei Empfang der Nachricht `NEIGHBOR_UPDATE_ACK` gestoppt. Sollte der Timer ablaufen, wird die Nachricht erneut gesendet, wobei maximal `MAX_RETRIES` Versuche unternommen werden. Sollte der Timer erneut auslaufen gibt das Master Device auf. In diesem Fall ist ein Ausfall des entsprechenden Nachbarn wahrscheinlich. Ein solcher Ausfall wird durch die in Abschnitt 5.8 beschriebenen Verfahren behandelt.

5.4.6 Sicherheitsbetrachtung

Die Join-Operation setzt in allen geschützten Nachrichten, die nicht über den *LLC* versendet werden, Nonces zum Schutz vor Angriffen durch Wiedereinspielen ein. Die einzige Ausnahme stellt die `ZONE_TRANSFER`-Nachricht dar, die über keine Nonce verfügt. Diese Nachricht kann jedoch nur ein einziges Mal, nämlich während der Zonenübergabe, zwischen zwei Knoten versendet werden, weswegen ein Angriff durch Wiedereinspielen keinen Sinn machen würde, da ein Knoten am Kontext erkennt, ob solch ein Angriff vorliegt. Angriffe durch Wiedereinspielen werden also wirksam verhindert, weswegen Eigenschaft E6 (Keine Angriffe durch Wiedereinspielen) von SCAN durch die Join-Operation nicht verletzt wird.

Im Folgenden wird gezeigt, dass die Join-Operation für die Eigenschaften E1-E4 sowie E7-E9 von SCAN verantwortlich ist. Abbildung 5.6 gibt einen Überblick. In der Abbildung und der weiteren Beschreibung wird folgende Notation verwendet: gilt eine Eigenschaft Ex, so wird dies als Ex geschrieben; gilt die Eigenschaft nicht, so wird dies durch $\neg Ex$ dargestellt. Der Übersichtlichkeit halber werden alle Eigenschaften zusammen in einer Mengenklammer geschrieben. Beispielsweise bedeutet $\{E1, E2, \neg E3, E4, E7, \neg E8, E9\}$, dass die Eigenschaften E1,E2,E4,E7 und E9 gelten und die Eigenschaften E3 und E8 nicht.

Zu zeigen: In einem SCAN gelten die Eigenschaften E1 (Authentifizerung), E2 (Autorisierung), E3 (Integrität und Vertraulichkeit zwischen Nachbarn), E4 (Integrität und Vertraulichkeit mit dem Master Device), E7 (Keine gezielte Zonenübernahme), E8 (Keine Manipulation der eigenen Zone) und E9 (Zonenverifikation).

Beweis:

Induktion über die Anzahl N der Knoten, die zu SCAN gehören.

Induktions-Annahme:

In einem SCAN mit N Knoten gelten die Eigenschaften E1 (Authentifizierung), E2 (Autorisierung), E3 (Integrität und Vertraulichkeit zwischen Nachbarn), E4 (Integrität und Vertraulichkeit mit dem Master Device), E7 (Keine gezielte Zonenübernahme), E8 (Keine Manipulation der eigenen Zone) und E9 (Zonenverifikation).

Induktionsanfang: $N = 1$:

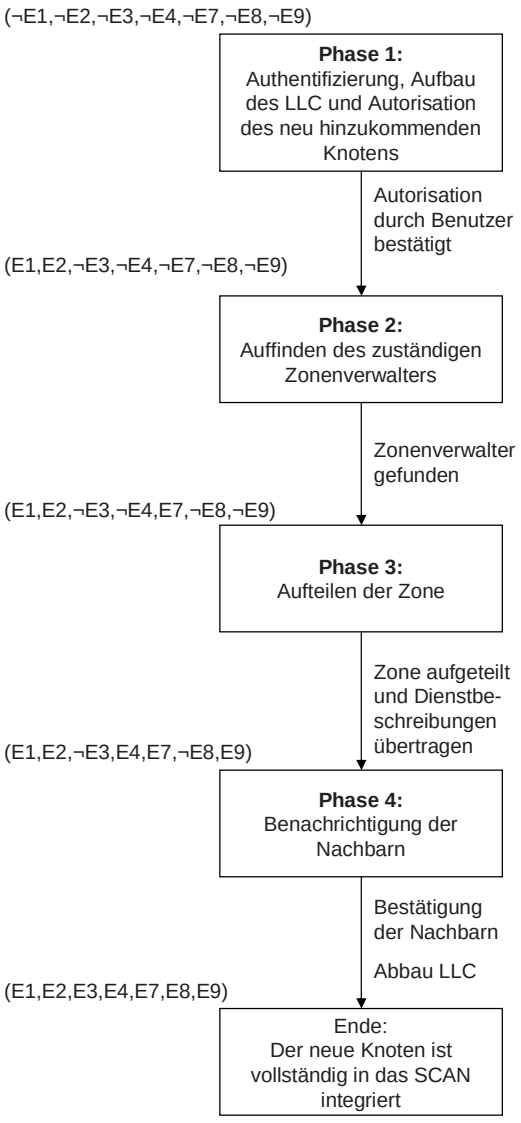


Abbildung 5.6: Überblick über die Eigenschaften E1-E4 und E7-E9 während der Join-Operation des Knoten K_{N+1}

Sei K_i der i-te Knoten, der ins SCAN integriert wird. Der Knoten K_1 bildet ein neues SCAN, da er der erste Knoten ist. Per Definition des SCAN-Raums als d-dimensionaler Torus ist K_1 in allen Dimensionen zu sich selbst benachbart, weswegen trivialerweise E3 (Integrität und Vertraulichkeit zwischen Nachbarn) gilt. Weiterhin sind E7 (Keine gezielte Zonenübernahme) und E8 (Keine Manipulation der eigenen Zone) erfüllt, weil vor dem Beitritt von K_1 keine Zone existiert hat, die übernommen werden könnte. Da K_1 in allen Dimensionen zu sich selbst benachbart ist, erfolgt während der Join-Operation keinerlei Kommunikation über Vermittlungsschicht oder Overlay sondern lediglich über den Location Limited Channel, der als sicher angesehen wird. Im Rahmen der Join-Operation bekommt K_1 in seiner Rolle als neu hinzukommender Knoten den gesamten SCAN-Raum als Zone zugewiesen, sowie einen Schlüssel zur Kommunikation mit dem Master Device, der an diese Zone gebunden ist, d. h. E4 (Integrität und Vertraulichkeit mit dem Master Device) sowie E9 (Zonenverifikation) sind erfüllt. Da der Benutzer das neue SCAN mit dem Master Device erzeugt gelten E1 (Authentifizierung) und E2 (Autorisierung).

Es gilt also: $\{E1, E2, E3, E4, E7, E8, E9\}$.

Induktionsschritt: $N \rightarrow N+1$:

Gelte die Induktions-Annahme für ein SCAN mit N Knoten, d. h. $\{E1, E2, E3, E4, E7, E8, E9\}$. Nun wird der N+1-te Knoten K_{N+1} durch die Join-Operation in das SCAN integriert:

Phase 1:

Zu Beginn von Phase 1 der Join-Operation gilt keine der Eigenschaften, da der neue Knoten noch nicht in das SCAN integriert ist, also $\{\neg E1, \neg E2, \neg E3, \neg E4, \neg E7, \neg E8, \neg E9\}$. Während Phase 1 nimmt der Benutzer die Authentifizierung des Sensorknotens vor und autorisiert den Knoten zum Beitritt. Also gilt nach Phase 1: $\{E1, E2, \neg E3, \neg E4, \neg E7, \neg E8, \neg E9\}$

Phase 2:

Vor Phase 2 der Join-Operation gilt $\{E1, E2, \neg E3, \neg E4, \neg E7, \neg E8, \neg E9\}$

1.)MD $\overset{\text{LLC}}{\rightarrow}$ JN : **ASSIGN_JP**[JP]

2.)MD $\overset{\text{OR}}{\rightarrow}$ JP : **JOIN_REQ**[addr_{MD}]

3.)ZO $\overset{\text{NR}}{\rightarrow}$ MD :**JOIN_REP**[$zone_{ZO}, E_{key_{MD,ZO}}(uid_{JP}, uid_{grpZO}, NL), mac$]

Durch Zuweisung des JP in Protokollschritt 1 kann der JN sich die Zone, die er durch die Join-Operation erhält, nicht mehr durch geschickte Wahl des JP selbst

aussuchen, d. h. E7 (Keine gezielte Zonenübernahme) ist erfüllt. Es kann zwar überprüft werden, ob der ZO zu einem vorigen Zeitpunkt die Zone $zone_{ZO}$ besessen hat, aber ob dies immer noch der Fall ist, kann erst überprüft werden, wenn die entsprechenden Nachbarn dies bestätigt haben, d. h. E8 (Keine Manipulation der eigenen Zone) ist noch nicht erfüllt.

Nach Phase 2 gilt $\{E1, E2, \neg E3, \neg E4, E7, \neg E8, \neg E9\}$

Phase 3:

Vor Phase 3 der Join-Operation gilt $\{E1, E2, \neg E3, \neg E4, E7, \neg E8, \neg E9\}$

4.)MD $\overset{\text{LLC}}{\rightarrow}$ JN :**ZONE_SPLIT**$[uid_{JP}, zone_{JN}, key_{MD,JN}, key_{grpJN}, NKL]$

5.)MD $\overset{\text{NR}}{\rightarrow}$ ZO :
ZONE_SPLIT$[E_{key_{MD,ZO}}(uid_{grpZO}, zone_{ZO_{neu}}, key_{MD,ZO}, key_{grpZO}, NKL), mac]$

In der `ZONE_SPLIT`-Nachricht erhält der JN den Schlüssel $key_{MD,JN}$, den er für die weitere Kommunikation mit dem Master Device, auch zu einem späteren Zeitpunkt nach Abbau des LLC, verwendet. Da in die Erzeugung des Schlüssels die Zoneninformation $zone_{JN}$ einfließt, diese Information also an den Schlüssel gebunden ist, gilt Eigenschaft E9 (Zonenverifikation) für die neue Zone von JN. Analog gilt dies für die neue Zone des ZO, denn auch dessen neuer Schlüssel $key_{MD,ZO}$ wird neu berechnet. Zwar könnte ein Angreifer nach einem Zone Split in einer späteren Join-Operation nach wie vor den alten Schlüssel verwenden und damit die alte Zone beanspruchen, allerdings wird dies durch die Protokollschritt 8 und 9 der Join-Operation unterbunden (siehe unten). Es gilt also auch für ZO Eigenschaft E9 (Zonenverifikation). Da im Rahmen der Join-Operation keine weiteren Zonen verändert werden, gilt also E9 (Zonenverifikation) für das ganze SCAN. Die beiden Knoten JN und ZO haben vom Master Device den Schlüssel $key_{MD,JN}$ bzw. $key_{MD,ZO}$ zugewiesen bekommen, mit dem Integrität und Vertraulichkeit der Kommunikation mit dem Master Device gesichert werden kann, weswegen Eigenschaft E4 (Integrität und Vertraulichkeit mit dem Master Device) gilt.

6.)ZO $\overset{\text{NR}}{\rightarrow}$ JN : **ZONE_TRANSFER**$[\text{E}_{\text{key}_{\text{ZO,JN}}}(\text{SDRs}), \text{mac}]$

7.)JN $\overset{\text{LLC}}{\rightarrow}$ MD : **ZONE_SPLIT_ACK**$[]$

Nach Phase 3 gilt $\{E1, E2, \neg E3, E4, E7, \neg E8, E9\}$

Phase 4:

Vor Phase 4 der Join-Operation gilt $\{E1, E2, \neg E3, E4, E7, \neg E8, E9\}$

8.)MD $\overset{\text{NR}}{\rightarrow}$ N_i : **NEIGHBOR_UPDATE**
$[E_{key_{MD,N_i}}(uid_{JP}, uid_{grpZO}, addr_{NN}, zone_{NN}, key_{N_i,NN}, key_{grpNN}), mac]$

9.)N_i $\overset{\text{NR}}{\rightarrow}$ MD : **NEIGHBOR_UPDATE_ACK**$[\text{E}_{\text{key}_{\text{MD,N}_\text{i}}}(\text{uid}_{\text{JP}}), \text{mac}]$

Durch die Antwort der Nachbarn mittels der `NEIGBOR_UPDATE_ACK` wird bestätigt, dass ZO auch aktuell noch die Zone verwaltet, die er vorgibt. Dadurch wird verhindert, dass ein Zonenverwalter vorgibt, eine Zone, die er früher verwaltet hat, die aber inzwischen geteilt wurde, immer noch zu verwalten. Es gilt also Eigenschaft E8 (Keine Manipulation der eigenen Zone). Durch Benachrichtigung der Nachbarn über die Änderungen und insbesondere durch Einrichten der Nachbarschlüssel gilt für zukünftige Nachrichten E3 (Integrität und Vertraulichkeit zwischen Nachbarn).

Nach Phase 4 gilt also $\{E1, E2, E3, E4, E7, E8, E9\}$.

q.e.d.

5.5 Die Insert-Operation

Die Insert-Operation dient einem Dienstanbieter dazu, Dienstbeschreibungen unter einem Dienstnamen im SCAN zu hinterlegen, d. h. über die Insert-Operation wird die Funktionalität *Veröffentlichung von Dienstbeschreibungen* realisiert. Die Dienstbeschreibungen werden *verteilt gespeichert*. Eine Dienstbeschreibung enthält genauere Informationen über den angebotenen Dienst und ermöglicht es einem Sensorknoten, unter mehreren Diensten mit gleichem Namen einen passenden auszusuchen. Der Aufbau und Inhalt einer Dienstbeschreibung hängt von der Anwendung ab, allerdings ist für alle Dienstbeschreibungen ein Eintrag verpflichtend, der die Vermittlungsschichtadresse des Dienstanbieters enthält. So wird der *Zugriff auf Dienste* ermöglicht.

Die Insert-Operation schützt Integrität und Vertraulichkeit von Dienstbeschreibungen während des Einfügens gegen Angriffe von Outsider-Angreifern und ist damit zum Teil für Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) verantwortlich. In einem späteren Abschnitt wird die Insert-Operation erweitert, um auch Schutz gegen Angriffe durch Insider-Angreifern gewährleisten zu können, und somit Eigenschaft E11 für die Insert-Operation vollständig zu erfüllen.

Die Insert-Operation wird von einem Dienstanbieter durch folgenden Funktionsaufruf angestoßen:

$$insert(Dienstbeschreibung, Dienstname)$$

Die Insert-Operation berechnet den Hash-Wert des Dienstnamens:

$$h_1 = hash(Dienstname)$$

Anschließend wird eine `INSERT`-Nachricht erzeugt, welche die Dienstbeschreibung DB beinhaltet und vom Dienstanbieter A an den Hash-Wert h_1 adressiert ist. Die `INSERT`-Nachricht wird anschließend über das Overlay verschickt:

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT}[\mathrm{DB}]}$$

Die Nachricht erreicht den Zonenverwalter der Zone, in welcher h_1 liegt. Dieser speichert das Paar (h_1, DB) in seinem Teil der verteilten Hash-Tabelle ab. Das bedeutet insbesondere, dass Knoten, die Dienstbeschreibungen speichern, zwar die Dienstbeschreibung und den Hash-Wert des Dienstnamens kennen, aber nicht den Dienstnamen selbst, da dieser nicht mit übermittelt wird und wegen der Eigenschaften von Hash-Funktionen auch nicht einfach errechnet werden kann.

5.5.1 Sicherheitsbetrachtung

Integrität und Vertraulichkeit der `INSERT`-Nachricht sind gegen Angriffe von Outsider-Angreifern geschützt, da die Nachbarschlüssel, wie bei der Join-Operation beschrieben, beim Overlay-Routing dazu verwendet werden, die Nachricht bei der Weitergabe zu verschlüsseln und mit einem Message Authentication Code zu versehen. Ein Insider kann die Dienstbeschreibung manipulieren, falls er sich auf dem Overlay-Routing-Pfad zwischen dem Dienstanbieter A und dem Zonenverwalter ZO, in dessen Zone h_1 liegt, befindet. Abbildung 5.7 zeigt ein Beispiel: zwischen A und ZO befinden sich die beiden Sensorknoten K_1 und K_2. Ist einer von diesen beiden Knoten feindlich, so kann er die entsprechende Dienstbeschreibung unbemerkt manipulieren. Die Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) wird also von der unmodifizierten Insert-Operation nicht erfüllt.

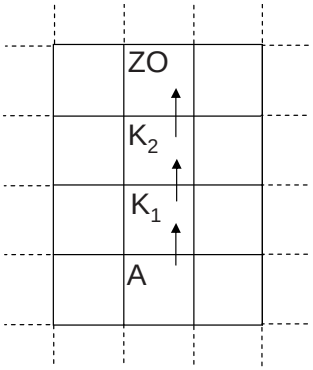


Abbildung 5.7: Mögliche Insider-Angreifer während der Insert-Operation

5.5.2 Replikation

Um den oben beschrieben Angriff durch Insider-Angreifer zu verhindern, ist es möglich, die Dienstbeschreibung *redundant* im SCAN zu speichern und später, während der Lookup-Operation, mehrere dieser Replikate abzufragen und über einen

Mehrheitsentscheid eine Dienstbeschreibung auszusuchen. Details zum Mehrheitsentscheid finden sich im Abschnitt über die Lookup-Operation, hier wird beschrieben, wie r Replikate einer Dienstbeschreibung erzeugt und an verschiedenen Punkten im SCAN-Raum gespeichert werden. Um die Koordinaten von r verschiedene Punkten zu bestimmen, werden während der Insert-Operation folgende Hash-Werte berechnet:

$$h_1 = hash(1|Dienstname)$$
$$h_2 = hash(2|Dienstname)$$
$$...$$
$$h_r = hash(r|Dienstname)$$

Dabei steht | für die Konkatenation von Werten. Die Eigenschaften einer Hash-Funktion bewirken, dass h_1 bis h_r im Werteraum der Hash-Funktion gleichverteilt sind. Dadurch ist die Wahrscheinlichkeit groß, dass die Punkte in Zonen verschiedener Zonenverwalter liegen.

Anschließend werden die entsprechenden Insert-Nachrichten an h_1, h_2, ..., h_r erzeugt:

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h_1} : \textbf{INSERT}[\mathrm{DB}]}$

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h_2} : \textbf{INSERT}[\mathrm{DB}]}$

...

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h_r} : \textbf{INSERT}[\mathrm{DB}]}$

Die Nachrichten werden wie oben beschrieben über das Overlay verschickt. Durch die redundante Speicherung genügt es später beim Abfragen der Dienstbeschreibungen, wenn im SCAN noch eine einzige Replik existiert. Die Sicherheit gegen Insider-Angriffe wird dadurch erhöht, dass mehrere Replikate einer Dienstbeschreibung abgefragt werden und über einen Mehrheitsentscheid festgestellt wird, welche Dienstbeschreibung die richtige ist. Bei einem Mehrheitsentscheid, bei dem r der r Replikate abgefragt werden, wird für jede zurückgelieferte Dienstbeschreibung ermittelt, ob sie $\lfloor \frac{r}{2} \rfloor + 1$ mal oder öfters zurückgeliefert wurde. Falls ja, wird diese Dienstbeschreibung als authentisch angesehen.

Allerdings erhöht die Anzahl der gewünschten Replikate den durchschnittlichen Speicheraufwand auf den Knoten sowie den Kommunikationsaufwand, da für jedes Replikat eine eigene `INSERT`-Nachricht und ein eigener Eintrag in der entsprechenden Hash-Tabelle erzeugt wird. Es handelt sich also hier um einen Trade-off zwischen

einerseits Fehlertoleranz und Sicherheit und andererseits Speicher- und Kommunikationsaufwand. Allerdings wird auch mit Replikation keine vollständige Sicherheit gegen Angriffe von Insider-Angreifern gewährleistet, sondern Angriffe werden nur deutlich erschwert.

5.6 Die Lookup-Operation

Die Lookup-Operation dient einem Dienstnutzer zur *Suche nach Dienstbeschreibungen*. Sie lieft zu einem gegebenen Dienstnamen eine Liste von Dienstbeschreibungen zurück, die unter dem Hash-Wert des entsprechenden Dienstnamens im SCAN gespeichert sind, wobei Wert gelegt wird auf eine möglichst *vollständige Liste von Diensten*.

Die Lookup-Operation schützt Integrität und Vertraulichkeit von Dienstbeschreibungen während des Abfragens gegen Angriffe von Outsider-Angreifern und ist damit zum Teil für Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) verantwortlich. In einem späteren Abschnitt wird die Lookup-Operation erweitert, um auch Schutz gegen Angriffe durch Insider-Angreifer gewährleisten zu können, und somit Eigenschaft E11 für die Lookup-Operation vollständig zu erfüllen.

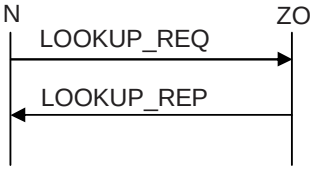


Abbildung 5.8: Ablauf der Lookup-Operation

Abbildung 5.8 zeigt den Ablauf der Lookup-Operation im Überblick. Ein Dienstnutzer N, der einen Dienst benötigt, stößt die Lookup-Operation durch den Aufruf der Funktion

$$lookup(Dienstname)$$

an. Die Funktion berechnet den Hash-Wert des Dienstnamens:

$$h_1 = hash(Dienstname)$$

Dies ist dieselbe Berechnung wie auch schon bei der Insert-Operation. Anschließend wird die `LOOKUP`-Nachricht erzeugt, die an h_1 adressiert wird. Ein Dienstnutzer N, der nach einem Dienst sucht, erzeugt also folgende `LOOKUP_REQ`-Nachricht und versendet sie mittels des Overlay-Routings:

$$\boxed{1)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_REQ}[]}$$

Die Nachricht erreicht den Zonenverwalter ZO, in dessen Zone h_1 liegt. Die `LOOKUP_REQ`-Nachricht wird über das Overlay-Routing versendet wie bei der Join-Operation beschrieben, d. h. Integrität und Vertraulichkeit der Nachricht sind gegen Angriffe von Outsider-Angreifern geschützt.

Der Zonenverwalter prüft, ob in seinem Teil der verteilten Hash-Tabelle Dienstbeschreibungen unter dem Hash-Wert h_1 abgelegt sind. Er antwortet mit folgender Nachricht über das Overlay:

$$\boxed{2)\text{ZO} \overset{\text{OR}}{\rightarrow} \text{N} : \textbf{LOOKUP\_REP}[\text{DB\_Liste}]}$$

Dabei ist *DB_Liste* eine Liste von Dienstbeschreibungen, die ZO unter dem Hash-Wert h_1 in seinem Teil der verteilten Hash-Tabelle gespeichert hat. Auch bei dieser Nachricht werden Integrität und Vertraulichkeit durch das Overlay-Routing gegen Angriffe von Outsider-Angreifern geschützt.

5.6.1 Timer

Der Dienstnutzer N startet mit dem Versenden der `LOOKUP_REQ`-Nachricht den Timer `LOOKUP_REQ_TIMER`, der nach der Zeitdauer `LOOKUP_REQ_TIMEOUT` abläuft, falls nicht zuvor die `LOOKUP_REP`-Nachricht empfangen wurde. Falls der Timer abläuft, wird die `LOOKUP_REQ`-Nachricht erneut versendet.

5.6.2 Sicherheitsbetrachtung

Angriffe durch Insider-Angreifer sind, ebenso wie bei der Insert-Operation, dann möglich, wenn sich ein oder mehrere Insider-Angreifer auf dem Overlay-Routing-Pfad zwischen dem Dienstnutzer N und dem Zonenverwalter ZO befinden. Die Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) wird also nur zum Teil erzeugt.

5.6.3 Mehrheitsentscheid bei Verwendung von Replikaten

Auch bei der Lookup-Operation kann die Anfälligkeit gegen Angriffe von Outsider-Angreifern dadurch gesenkt werden, dass nicht nur eine Dienstbeschreibung, sondern mehrere Replikate der Dienstbeschreibung abgefragt werden und über einen Mehrheitsentscheid die authentische Dienstbeschreibung ermittelt wird. Dies setzt voraus, dass zuvor bei der Insert-Operation auch Replikate eingespeichert werden.

Zum Auffinden der Replikate werden wiederum die Koordinaten von r Punkte im SCAN-Raum nach folgender Rechenvorschrift berechnet:

$$h_1 = hash(1|Dienstname)$$

$$h_2 = hash(2|Dienstname)$$

...

$$h_r = hash(r|Dienstname)$$

Dabei steht| für die Konkatenation von Werten. Anschließend wird für jeden Hash-Wert eine Lookup-Nachricht erzeugt:

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_REQ}[]}$

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_2 : \textbf{LOOKUP\_REQ}[]}$

...

$\boxed{\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_\mathrm{r} : \textbf{LOOKUP\_REQ}[]}$

Der Dienstnutzer erhält dann jede der r Replikate der Dienstbeschreibung zurück. Der Knoten entscheidet mittels eines Mehrheitsentscheids, welches Replikat er als nicht manipuliert ansieht. Für einen erfolgreichen Angriff muss ein Angreifer also bei r Replikaten mindestens $\left\lfloor \frac{r}{2} \right\rfloor + 1$ dieser Replikate übernehmen.

Im Folgenden soll die Erfolgswahrscheinlichkeit p_{rep} einer Lookup-Operation bei Verwendung von r Replikaten berechnet werden, wenn die Erfolgswahrscheinlichkeit p_{in} einer Insert-Operation ohne Replikation bzw. die Erfolgswahrscheinlichkeit p_{lo} einer Lookup-Operation ohne Replikation gegeben sind:

Sei X eine Zufallsvariable die angibt, wie viele der r Dienstbeschreibungen erfolgreich durch die Insert-Operation im SCAN gespeichert werden. Die Wahrscheinlichkeit, dass i der r ($i \leq r$) Dienstbeschreibungen erfolgreich gespeichert wurden ist:

$$P(X = i) = \binom{r}{i} p_{in}^{i} (1 - p_{in})^{(r-i)}$$

Sei Y eine Zufallsvariable, die angibt, wie viele der i erfolgreich abgelegten Dienstbeschreibungen durch die Lookup-Operation erfolgreich abgefragt werden. Die Wahrscheinlichkeit, dass j ($j \leq i \leq r$) Dienstbeschreibungen erfolgreich zurückgeliefert werden ist dann:

$$P(Y = j) = \binom{i}{j} p_{lo}^{j} (1 - p_{lo})^{(i-j)}$$

Damit ist die Wahrscheinlichkeit p_{rep}, dass mindestens $\left\lfloor \frac{r}{2} \right\rfloor + 1$ von eingespeisten r Replikaten zurückgeliefert wurden:

$$
\begin{aligned}
p_{rep} &= \begin{cases} 0 & , falls \quad \lfloor \frac{r}{2} \rfloor + 1 > i \\ P(i \geq Y \geq \lfloor \frac{r}{2} \rfloor + 1) & , sonst \end{cases} \\
&= \begin{cases} 0 & , falls \quad \lfloor \frac{r}{2} \rfloor + 1 > i \\ \sum_{k=\lfloor \frac{r}{2} \rfloor + 1}^{i} P(Y = k) & , sonst \end{cases} \\
&= \begin{cases} 0 & , falls \quad \lfloor \frac{r}{2} \rfloor + 1 > i \\ \sum_{k=\lfloor \frac{r}{2} \rfloor + 1}^{i} \binom{i}{k} p_{lo}^{k} (1 - p_{lo})^{(i-k)} & , sonst \end{cases}
\end{aligned}
$$

Unter Verwendung von $P(X = i)$ und unter Berücksichtigung der Tatsache, dass der Mehrheitsentscheid nur erfolgreich sein kann, wenn gilt $i \geq \lfloor \frac{r}{2} \rfloor + 1$ ergibt sich für p_{rep} folgende Formel:

$$
\begin{aligned}
p_{rep} &= \sum_{m=\lfloor \frac{r}{2} \rfloor + 1}^{r} (P(X = m) * (\sum_{k=\lfloor \frac{r}{2} \rfloor + 1}^{m} \binom{m}{k} p_{lo}^{k} (1 - p_{lo})^{(m-k)})) \\
&= \sum_{m=\lfloor \frac{r}{2} \rfloor + 1}^{r} (\binom{r}{m} p_{in}^{m} (1 - p_{in})^{r-m} * (\sum_{k=\lfloor \frac{r}{2} \rfloor + 1}^{m} \binom{m}{k} p_{lo}^{k} (1 - p_{lo})^{(m-k)}))
\end{aligned}
$$

Die Verbesserung der Erfolgswahrscheinlichkeit p_{verb} einer Lookup-Operation bei Verwendung von Replikation kann damit folgendermaßen berechnet werden:

$$p_{verb} = p_{rep} - p_{lo} * p_{in}$$

Abbildung 5.9 zeigt für $r = 3$ und $p_{lo} = p_{in}$ die Verbesserung p_{verb} der Erfolgswahrscheinlichkeit einer Lookup-Operation. Es wird deutlich, dass erst ab einer gewissen Wahrscheinlichkeit p_{lo} bzw. p_{in} Replikation zu einer Verbesserung führt. Ist die Wahrscheinlichkeit einer erfolgreichen Lookup-Operation zu klein, so werden zu wenig Replikate erfolgreich zurückgeliefert, so dass der Mehrheitsentscheid fehlschlägt.

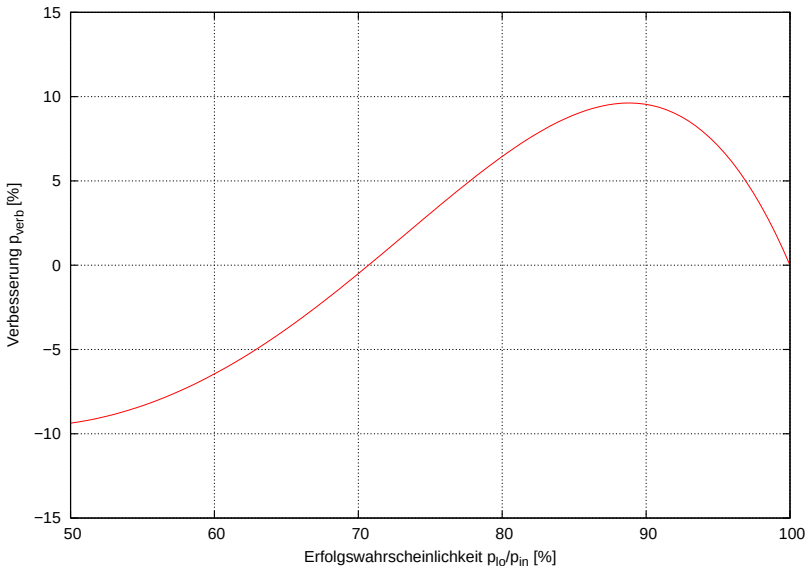


Abbildung 5.9: Verbesserung, die durch den Einsatz von Replikation (3 Replikate) erreicht wird

5.7 Schlüsselaustauschprotokoll

Die Lookup- und Insert-Operation verwenden beide das Overlay-Routing, um die Integrität und Vertraulichkeit der ausgetauschten Nachrichten gegen Angriffe von Outsider-Angreifern zu schützten. Allerdings bietet dieses Verfahren keinen Schutz gegen Angriffe durch Insider-Angreifer, die sich auf dem Overlay-Routing-Pfad befinden. Durch die Verwendung von Replikaten kann ein gewisser Schutz gegen Insider-Angreifer erreicht werden. In diesem Abschnitt werden zwei Verfahren vorgestellt, mit Hilfe derer ein Schlüssel zwischen zwei beliebigen Knoten im Overlay ausgetauscht werden kann: *Single-Path-Key-Exchange (SPX)* und *Multi-Path-Key-Exchange (MPX)*. Existiert ein solcher Schlüssel zwischen zwei Knoten A und B, so können Integrität und Vertraulichkeit von Nachrichten zwischen A und B durch Verwendung eines Message Authentication Codes und Verschlüsselung der Nachricht erreicht werden, d. h. die hier vorgestellten Schlüsselaustauschprotokolle sind für die Eigenschaft E5 (Integrität und Vertraulichkeit zwischen beliebigen Knoten) verantwortlich. Ist Eigenschaft E5 erfüllt, so kann damit auch Eigenschaft E11 (Schutz von Dienstbeschreibungen gegen Angriffe von Outsider-Angreifern und Insider-Angreifern) erfüllt werden, indem der entsprechenden Lookup- bzw. Insert-Operation ein Schlüsselaustausch vorausgeht.

Die Schlüsselaustauschprotokolle SPX und MPX werden im Folgenden beschrieben. Sowohl SPX als auch MPX setzen auf den Nachbarschlüsseln auf. SPX basiert auf der Idee, einen symmetrischen Schlüssel zwischen einem Knoten A und einem Knoten B einzurichten, indem ein Schlüssel mittels Overlay-Routing über einen Overlay-Routing-Pfad vom Knoten A zum Knoten B geschickt wird. MPX verwendet hierzu mehrere Overlay-Routing-Pfade und sendet über jeden Pfad nur einen Teil des

Schlüssels. Mittels MPX kann, als positiver Nebeneffekt, verifiziert werden, ob ein Knoten eine Zone verwaltet, in der ein gewisser Punkt liegt. Mit MPX wird also die Eigenschaft E10 (Authentifizierung Zonenverwalter) erreicht.

5.7.1 Single-Path-Key-Exchange (SPX)

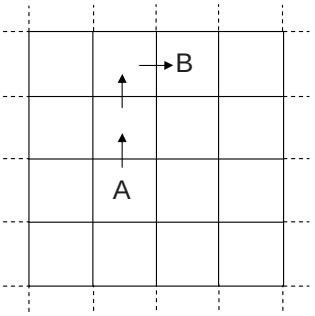


Abbildung 5.10: Schematische Darstellung eines SPX Schlüsselaustauschs

Abbildung 5.10 zeigt einen Schlüsselaustausch mittels Single-Path-Key-Exchange (SPX) zwischen einem Knoten A und einem Knoten B.

Ein Knoten A erzeugt einen Schlüssel *key* zufällig. Anschließend sendet A diesen Schlüssel in einer `SPX`-Nachricht mittels des Overlay-Routings an den Knoten B:

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{SPX}[\mathrm{key}]}$$

Um eine Nachricht an B senden zu können, muss A mindestens einen Punkt in der Zone von B kennen, da dieser Punkt für die Adressierung im Overlay-Routing notwendig ist. Ein solcher Punkt ist z. B. bekannt, wenn der Schlüsselaustausch einer Insert- oder Lookup-Operation vorausgeht: in diesem Fall wird der Hash-Wert des Dienstnamens verwendet.

Die `SPX`-Nachricht wird mittels des Overlay-Routings versendet, d. h. Integrität und Vertraulichkeit werden jeweils beim Versenden der Nachricht zwischen zwei im Overlay benachbarten Knoten sichergestellt. Einem Outsider-Angreifer ist es also nicht möglich, den Schlüssel mitzulesen. Ein Insider-Angreifer kennt den Schlüssel, falls er auf dem Overlay-Routing-Pfad zwischen A und B liegt. Auf dem Overlay-Routing-Pfad zwischen zwei beliebigen Knoten befinden sich im Durchschnitt h Knoten, wobei h die durchschnittliche Pfadlänge in SCAN ist. Die durchschnittliche Pfadlänge h hängt dabei von der Gesamtknotenanzahl N sowie der Dimensionalität d des SCAN ab. Nach [77] kann die durchschnittliche Pfadlänge h in einem Content Addressable Network (CAN) nach folgender Formel abgeschätzt werden:

$$h = \frac{d}{4} N^{\frac{1}{d}} \tag{5.1}$$

Da der CAN-Raum die gleiche Struktur wie der SCAN-Raum hat, gilt diese Formel auch für SCAN.

Bei einem Anteil von m $(0 \leq m \leq 1)$ bösartigen Knoten von allen Knoten des SCANs beträgt die Wahrscheinlichkeit p_{spx} $(0 \leq p_{spx} \leq 1)$, dass alle Zonen, über die ein Overlay-Routing-Pfad führt, von gutartigen Knoten verwaltet werden:

$$p_{spx} = (1 - m)^h \tag{5.2}$$

Dabei ist h die Länge des Overlay-Routing-Pfads. Die durchschnittliche Wahrscheinlichkeit p_{spx} berechnet sich mit der durchschnittlichen Overlay-Routing-Pfadlänge aus Gleichung 5.1 zu:

$$p_{spx} = (1 - m)^{\frac{d}{4}N^{\frac{1}{d}}} \tag{5.3}$$

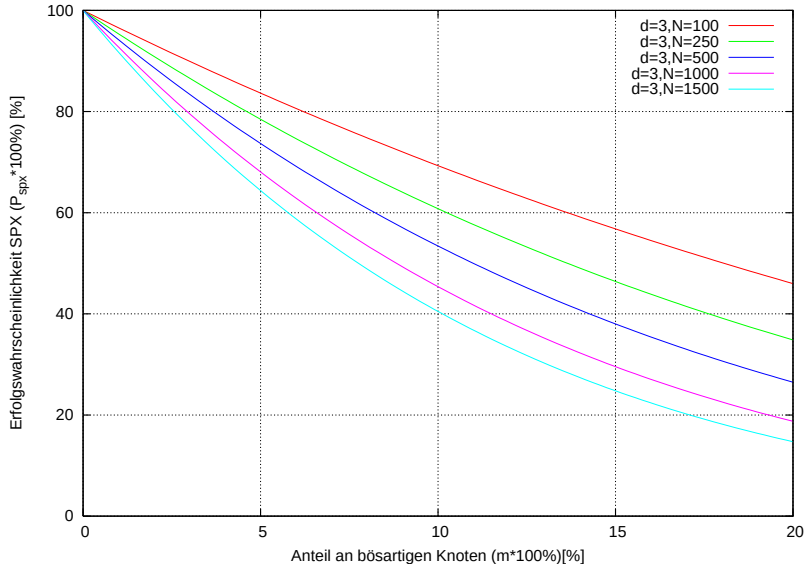


Abbildung 5.11: SPX-Schlüsselaustauschwahrscheinlichkeit für ein SCAN mit N Knoten und Dimension d=3

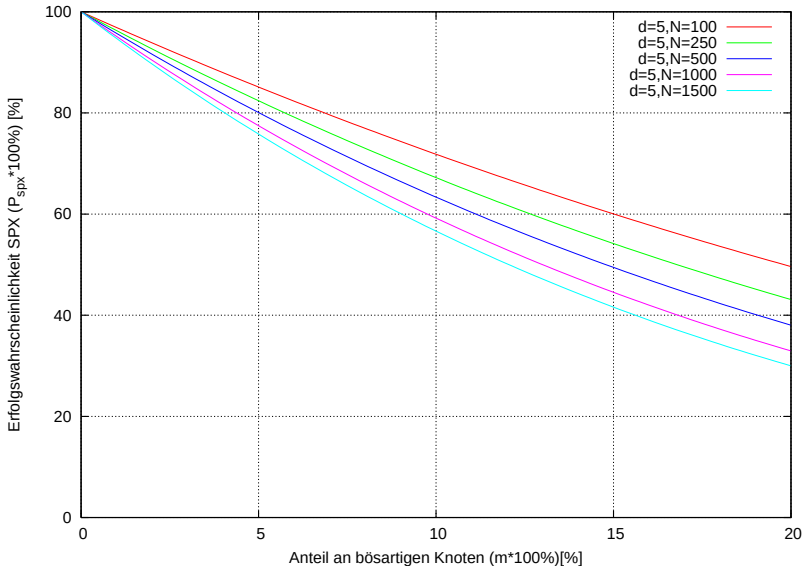


Abbildung 5.12: SPX-Schlüsselaustauschwahrscheinlichkeit für ein SCAN mit N Knoten und Dimension d=5

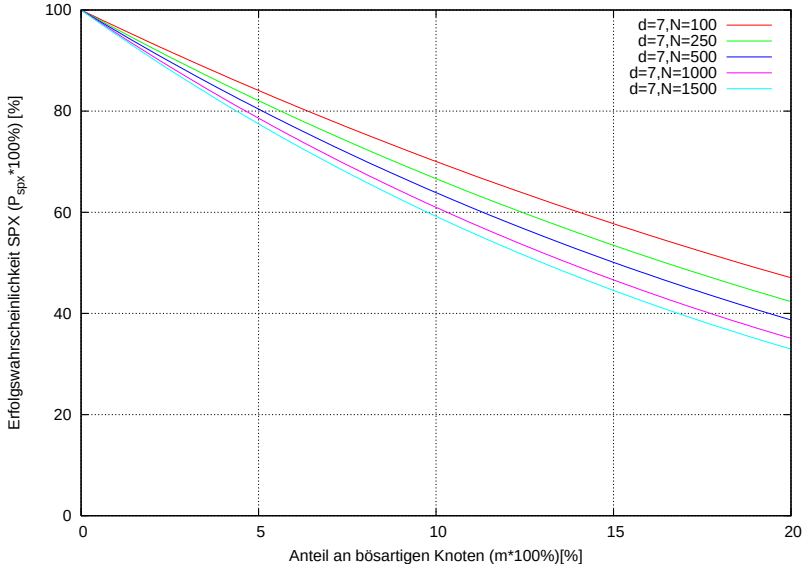


Abbildung 5.13: SPX-Schlüsselaustauschwahrscheinlichkeit für ein SCAN mit N Knoten und Dimension d=7

Gleichung 5.3 gibt also die Erfolgswahrscheinlichkeit eines Schlüsselaustauschs mittels SPX an, da bereits ein einziger Insider-Angreifer auf dem Pfad durch das Overlay ausreichend wäre, um den Schlüsselaustausch zu infiltrieren, d. h. in den Besitz des Schlüssels zu kommen.

Abbildung 5.11 zeigt die Erfolgswahrscheinlichkeit p_{spx} eines SPX-Schlüsselaustauschs unter der Annahme eines Anteils m von bösartigen Knoten im Netz sowie einer SCAN-Dimension von $d = 3$ jeweils für eine Gesamtknotenanzahl von 100, 500, 1000 und 1500. In Abbildung 5.12 wird statt $d = 3$ angenommen, dass ein SCAN mit $d = 5$ Dimensionen vorliegt, in Abbildung 5.13 wird $d = 7$ verwendet. Aus dem Vergleich der Abbildungen 5.11 und 5.12 wird deutlich, dass die Dimensionalität d des SCAN-Raums einen Einfluss auf die Erfolgswahrscheinlichkeit eines Schlüsselaustauschs hat: steigt d, so steigt auch die Erfolgswahrscheinlichkeit des Schlüsselaustauschs. Dies hängt damit zusammen, dass bei höherer Dimension die durchschnittliche Länge eines Overlay-Routing-Pfads nach Formel 5.1 sinkt, somit befinden sich weniger Knoten auf diesem Pfad, d. h. weniger potentielle Angreifer sind vorhanden und damit steigt die Wahrscheinlichkeit für einen erfolgreichen Schlüsselaustausch mit SPX.

Mittels SPX lassen sich sowohl die Lookup- als auch die Insert-Operation von SCAN, bei gleichem Sicherheitsniveau, effizienter gestalten als beim bisher vorgestellten Entwurf der Lookup- und Insert-Operation. Dabei wird jeweils gleich vorgegangen: bevor die eigentliche Lookup- respektive Insert-Nachricht verschickt wird, erfolgt ein Schlüsselaustausch mittels SPX und außerdem werden die Vermittlungsschichtadressen der Kommunikationspartner ausgetauscht. Die eigentlichen Nachrichten der Lookup- bzw. Insert-Operation werden dann nicht mehr über das Overlay geschickt, sondern können direkt über die Vermittlungsschicht geschickt werden.

5.7.1.1 Lookup mit SPX

Bei der Lookup-Operation sind durch den Einsatz von SPX Einsparungen beim Kommunikationsaufwand zu erwarten, da vermieden wird, die Liste von Dienstbeschreibungen, die in der `LOOKUP_REP`-Nachricht enthalten ist, über das Overlay zu versenden. Stattdessen wird ein Schüssel ausgetauscht und die gesamte weitere Kommunikation erfolgt auf Vermittlungsschicht.

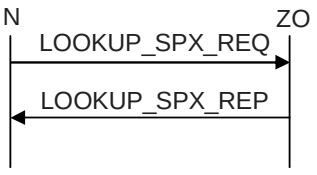


Abbildung 5.14: Ablauf der Lookup-Operation bei Verwendung von Single Path Key Exchange (SPX)

Abbildung 5.14 zeigt die Lookup-Operation unter Verwendung von SPX zwischen den Knoten N und ZO im Überblick.

Ein Dienstnutzer N ruft die Funktion `lookup_spx(Dienstname)` auf, um einen Dienst aufzufinden. Diese Funktion arbeitet analog zur Funktion `lookup(Dienstname)`, die bereits bei der Lookup-Operation beschrieben wurde:

Zuerst wird der Hash-Wert des Dienstnamens berechnet:

$$h_1 = hash(Dienstname)$$

Anschließend versendet die Funktion im Overlay eine Nachricht an h_1:

$$\boxed{1.)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h_1} : \textbf{LOOKUP\_SPX\_REQ}[\mathrm{key}, \mathrm{addr_N}]}$$

Diese Nachricht beinhaltet den Schlüssel *key*, der durch SPX für die Kommunikation zwischen den beiden Knoten eingerichtet wird, sowie die Vermittlungsschichtadresse $addr_N$ des Dienstnutzers. Die Nachricht wird mittels des Overlay-Routings versendet, wobei die Nachbarschlüssel jeweils zum Schutz von Integrität und Vertraulichkeit gegen Angriffe von Outsider-Angreifern benutzt werden. Ein Insider-Angreifer kann die Nachricht mitlesen oder verändern, wenn er sich direkt auf dem Overlay-Routing-Pfad befindet, den die `LOOKUP_SPX`-Nachricht vom Knoten N zu der Zone, in der h_1 liegt, nimmt. Der Verwalter dieser Zone, der Knoten ZO, sendet auf Vermittlungsschicht folgende Nachricht zurück:

$$\boxed{2.)\mathrm{ZO} \overset{\mathrm{NR}}{\rightarrow} \mathrm{N} : \textbf{LOOKUP\_SPX\_REP}[\mathrm{E_{key}}(\mathrm{DB} - \mathrm{Liste}), \mathrm{mac}]}$$

Der Zonenverwalter ZO kann die Nachricht nun über die Vermittlungsschicht verschicken, weil ihm durch die Nachricht `LOOKUP_SPX_REQ` die Vermittlungsschichtadresse des Dienstnutzers bekannt ist ($addr_N$). Die Nachricht beinhaltet die Liste der Dienstbeschreibungen ($DB - Liste$), die der Zonenverwalter ZO zu dem Hash-Wert h_1 gespeichert hat.

Die Vertraulichkeit der Nachricht ist durch Verschlüsselung mit dem zuvor ausgetauschten Schlüssel *key* geschützt. Die Integrität der Nachricht ist durch den Message Authentication Code *mac* geschützt, in dessen Erstellung ebenfalls der Schlüssel *key* einfließt.

Der Schlüssel *key* kann, falls auf dem Knoten ausreichend Speicher zur Verfügung steht, für weitere Anfragen gespeichert werden, so dass bei weiteren Anfragen derselbe Schlüssel verwendet werden kann.

Da eine Lookup-Operation mit SPX genau wie eine Lookup-Operation ohne SPX durch einen Insider-Angreifer auf dem Overlay-Routing-Pfad zwischen N und ZO angreifbar ist, geht mit der Verwendung der Lookup-Operation mit SPX nur ein leichter Anstieg des Sicherheitsniveaus einher, der daraus resultiert, dass der ausgetauschte Schlüssel für spätere Anfragen verwendet werden kann und somit ein

Knoten auf dem Pfad zwischen N und ZO, der erst zu einem Zeitpunkt nach dem Schlüsselaustausch zum Angreifer wird, keinen Einfluss mehr hat.

SPX geht allerdings mit einem Effizienzgewinn einher, da die Diensteliste nicht über das Overlay versendet werden muss, sondern direkt auf Vermittlungsschicht gesendet werden kann. Dies bedeutet wesentlich weniger Kommunikationsaufwand, da ein Hop im Overlay üblicherweise mehreren Hops auf Vermittlungsschicht entspricht.

Bei Lookup mit SPX wird der Timer `LOOKUP_SPX_REQ_TIMER` mit Versenden der `LOOKUP_SPX_REQ`-Nachricht gestartet. Der Timer läuft nach der Zeitdauer `LOOKUP_SPX_REQ_TIMEOUT` ab, falls nicht zuvor die `LOOKUP_SPX_REP`-Nachricht empfangen wurde. Der Dienstnehmer kann entsprechend die `LOOKUP_SPX_REQ`-Nachricht erneut verschicken.

5.7.1.2 Insert mit SPX

Auch bei der Insert-Operation kann der Kommunikationsaufwand durch SPX gesenkt werden, da in diesem Fall nicht die potentiell große Dienstbeschreibung sondern nur ein wenige Byte großer Schlüssel über das Overlay übertragen werden muss. Die Dienstbeschreibung kann dann anschließend über die Vermittlungsschicht versendet werden.

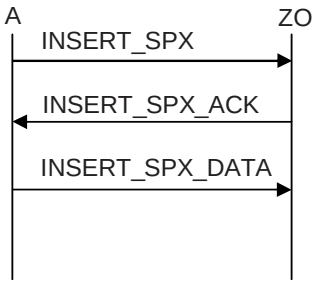


Abbildung 5.15: Ablauf der Insert-Operation bei Verwendung von Single Path Key Exchange (SPX)

Abbildung 5.15 zeigt den Ablauf der Insert-Operation mit SPX zwischen einem Knoten A und einem Knoten ZO.

Ein Dienstanbieter A ruf die Funktion `insert_spx(Dienstbeschreibung,Dienstname)` auf, die analog zur Funktion `insert(Dienstbeschreibung,Dienstname)` funktioniert. Die Funktion berechnet den Hash-Wert des Dienstnamens:

$$h_1 = hash(Dienstname)$$

Anschließend wird folgende Nachricht erzeugt und an h_1 verschickt:

$$\boxed{1.)\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT\_SPX}[\mathrm{key}, \mathrm{addr}_\mathrm{A}]}$$

Die `INSERT_SPX`-Nachricht beinhaltet einen vom Knoten A erzeugten Schlüssel (key), der zum Schutz der weiteren Kommunikation eingesetzt werden soll, sowie die Vermittlungsschichtadresse des Knoten A ($addr_A$).

Die `INSERT_SPX`-Nachricht wird über das Overlay-Routing übertragen, d. h. Integrität und Vertraulichkeit der Nachricht sind gegenüber Angriffen von Outsider-Angreifern gewährleistet.

Die Nachricht erreicht schließlich die Zone, in der h_1 liegt. Der Knoten, der diese Zone verwaltet (ZO) erhält den Schlüssel key und erzeugt eine Antwortnachricht, die er auf Vermittlungsschicht an $addr_A$ verschickt:

$$\boxed{2.)\mathrm{ZO} \overset{\mathrm{NR}}{\rightarrow} \mathrm{A} : \textbf{INSERT\_SPX\_ACK}[\mathrm{E}_{\mathrm{key}}(\mathrm{addr}_{\mathrm{ZO}}), \mathrm{mac}]}$$

Die Vertraulichkeit der `INSERT_SPX_ACK`-Nachricht ist durch die Verschlüsselung mit dem Schlüssel key geschützt. Die Integrität der Nachricht wird durch den Message Authentication Code mac geschützt, in dessen Erzeugung der Schlüssel key eingeflossen ist. Die Nachricht an sich enthält lediglich die Vermittlungsschichtadresse des Zonenverwalters ZO ($addr_{ZO}$). Gelingt dem Knoten A die Entschlüsselung der Nachricht, so versendet er schließlich die Dienstbeschreibung auf Vermittlungsschicht an ZO. Dazu erzeugt er folgende Nachricht an den Knoten mit der Adresse $addr_{ZO}$:

$$\boxed{3.)\mathrm{A} \overset{\mathrm{NR}}{\rightarrow} \mathrm{ZO} : \textbf{INSERT\_SPX\_DATA}[\mathrm{E}_{\mathrm{key}}(\mathrm{DB}), \mathrm{mac}]}$$

Diese Nachricht beinhaltet die Dienstbeschreibung DB. Die Nachricht ist zum Schutz der Vertraulichkeit mit dem Schlüssel key verschlüsselt. Der Message Authentication Code mac dient zum Schutz der Integrität der Nachricht. In seine Erstellung ist der Schlüssel key eingeflossen.

Für die Sicherheit der Insert-Operation mittels SPX gilt analog das bei der Lookup-Operation mittels SPX gesagte. Bei Lookup mit SPX wird der Timer `INSERT_SPX_TIMER` mit Versenden der `INSERT_SPX`-Nachricht gestartet. Der Timer läuft nach der Zeitdauer `INSERT_SPX_TIMEOUT` ab, falls nicht zuvor die `INSERT_SPX_ACK`-Nachricht empfangen wurde. Der Dienstnehmer kann entsprechend die `INSERT_SPX`-Nachricht erneut verschicken.

mx.d

5.7.2 Multi-Path-Key-Exchange (MPX)

Auch Multi-Path-Key-Exchange (MPX) benutzt das Overlay, um einen Schlüssel auszutauschen. Allerdings wird der Schlüssel bei MPX nicht nur über lediglich einen Overlay-Pfad ausgetauscht, sondern mehrere *Schlüsselteile* werden über jeweils verschiedene Overlay-Pfade versendet. Dadurch wird es einem Insider-Angreifer schwieriger, einen erfolgreichen Angriff auszuführen, da er sich jetzt nicht nur, wie beim

Single-Path-Key-Exchange, auf einem Overlay-Pfad befinden muss, um den Schlüsselaustausch zu infiltrieren, sondern er muss sich auf allen Pfaden befinden.

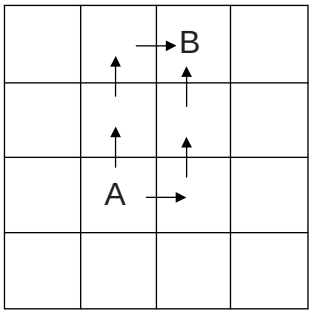


Abbildung 5.16: Schematische Darstellung eines MPX Schlüsselaustauschs

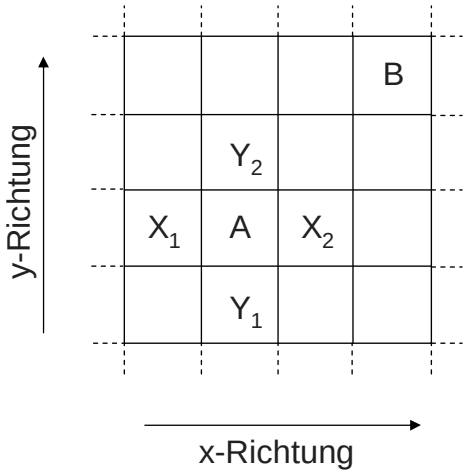


Abbildung 5.17: Nachbarn der Zone A in x- und in y-Richtung

Abbildung 5.16 illustriert einen Schlüsselaustausch zwischen Knoten A und einem Knoten B: A erzeugt einen Schlüssel key und teilt diesen in zwei Teile key_1 und key_2. Verschiedene Methoden zur Schlüsselteilung wurden bereits in Kapitel 2.2.5 beschrieben. Jeder dieser Schlüsselteile wird über einen gesonderten Overlay-Routing-Pfad versendet.

Knoten werden in SCAN so integriert, dass jede Zone in jeder Dimension mindestens zwei Nachbarn hat, denn nur so kann das Overlay-Routing überhaupt funktionieren. Abbildung 5.17 zeigt dies beispielhaft für $d = 2$: Die Zone A hat in der ersten Dimension, der x-Richtung, die Nachbarn X_1 und X_2 und in der zweiten Dimension, der y-Richtung, die Nachbarn Y_1 und Y_2. Die Zone A verfügt in jeder Dimension über einen Nachbarn, der, auf diese Dimension bezogen, am nächsten zum Ziel liegt. Im Beispiel sind dies von A aus gesehen zum Ziel B die Zonen X_2 und Y_2. Im Gesamten verfügt eine Zone also über d solcher Nachbarn, in jeder Dimension einen. Diese werden

später benutzt, um d zonendisjunkte Pfade zu konstruieren. Dabei sind zwei Overlay-Routing-Pfade genau dann *zonendisjunkt*, wenn die Pfade über keine gemeinsamen Zonen führen unter Ausnahme der Start- und Zielzone. Es könnten also mehr als die angesprochenen d zonendisjunkte Pfade gebildet werden, wenn genügend Nachbarn vorhanden sind. Allerdings würde dies dazu führen, dass längere Pfade entstehen, da nicht nur diejenigen Nachbarn gewählt werden, die in der entsprechenden Dimension am nächsten zum Ziel liegen. Außerdem könnte nicht sichergestellt werden, dass auch das Ziel (im Beispiel die Zone B) über genügend Nachbarn verfügt, so dass die Pfade zonendisjunkt sind. Deshalb wird in dieser Arbeit, wie oben besprochen, die Anzahl der verwendeten Pfade auf d eingeschränkt. Durch die Anzahl der zonendisjunkten Pfaden ist auch die Anzahl n an für MPX verwendbaren Geheimnisteilen festgelegt: $n \leq d$. Da durch das Overlay-Routing Integrität und Vertraulichkeit von Nachrichten zwischen benachbarten Overlay-Knoten geschützt werden, spielt es keine Rolle, ob mehrere zonendisjunkten Pfade über denselben Underlay-Knoten führen: da die Underlay-Knoten die Integrität und Vertraulichkeit einer Nachricht nicht verletzen können, werden in der Sicherheitsbetrachtung die Underlay-Knoten deshalb nicht berücksichtigt.

Die zonendisjunkten Pfade können wie folgt konstruiert werden: Der Startknoten A verfügt über d angrenzende Zonen, über die er die Nachricht in Richtung des Zielknotens B schicken kann. Dies sind die Knoten, die in der jeweiligen Dimension am nächsten zum Ziel liegen, z. B. die Knoten X_2 und Y_2 in Abbildung 5.17. Die d verschiedenen Pfade entstehen dadurch, dass der Knoten A die Nachrichten in jeder Dimension an denjenigen Nachbarn weitergibt, der in dieser Dimension am nächsten zum Ziel liegt. Alle Knoten führen den in Listing 5.1 angegebenen Routing-Algorithmus ausführen: Das SCAN-Routing leitet die Nachricht dann so lange in einer Dimension von Nachbar zu Nachbar weiter, bis in dieser Dimension das Ziel erreicht ist. Anschließend wird die Nachricht in einer anderen Dimension, in der das Ziel noch nicht erreicht ist, weitergeleitet. Die dadurch entstehenden Pfade überschneiden sich nicht und alle Nachrichten mit den Schlüsselteilen kommen erst in der Zone von Knoten B wieder zusammen. Abbildung 5.16 verdeutlicht die Konstruktion der zonendisjunkten Pfade: Im Beispiel wird die Nachricht in x- bzw. in y-Dimension in Richtung des Ziels versendet. Dadurch entstehen die zonendisjunkten Pfade.

```
1
2 routing(message msg,address fromAddr,dimension fromDim){
3
4 // Eingabe: msg = weiterzuleitende Nachricht
5 //      fromAddr = Vermittlungsschichtadresse Sender
6 //      fromDim = Dimension, in der Routing erfolgte
7
8 SCANaddress dest=msg.destination();
9
10 if(!dest.liesIn(this.zone,fromDim)){
11
12  /*falls Ziel der Nachricht in Dimension fromDim
```

```
13    nicht in eigener Zone liegt*/
14
15  neighbor n=neighborList.nearstInDimension(fromDim);
16  if(n.addr!=fromAddr){
17
18   /* Falls Nachricht nicht von diesem Nachbarn kam */
19
20   send(n.addr,msg)
21   return;
22  }
23 }
24
25 /* Wähle neue Dimension für Weiterleitung */
26
27 dimension nextDim=fromDim.next();
28
29 while(nextDim!=fromDim){
30  if(!dest.liesIn(this.zone,nextDim){
31
32   /* Ziel in dieser Dimension noch nicht erreicht */
33
34   neighbor n=neighborList.nearstInDimension(dest,nextDim);
35
36   if(n.addr==fromAddr){
37
38    /* Nachricht kam von Knoten n */
39
40    nextDim=nextDim.next();
41    continue;
42   }else{
43
44    /* Nachricht kann an n versendet werden */
45
46    send(n.addr,msg);
47    return;
48   }
49  }else{
50
51   /* Ziel in Dimension nextDim bereits erreicht */
52
53   nextDim=nextDim.next();
54  }
55 }
56
57 /* Ziel der Nachricht erreicht */
58
59 sendToUpperLayer(msg);
60 return;
```

```
61 }
```

Listing 5.1: Erzeugung der zonendisjunkten Pfade

Konkret läuft ein MPX-Schlüsselaustausch folgendermaßen ab: Der Knoten A versendet an den Knoten B folgende d Nachrichten mittels des Overlay-Routings:

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_1]}$$

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_2]}$$

...

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_\mathrm{d}]}$$

Integrität und Vertraulichkeit der Nachricht werden durch das Overlay-Routing gegen Angriffe von Outsider-Angreifern geschützt. Angreifer können nur Insider-Angreifer sein, die sich auf allen d Overlay-Routing-Pfad zwischen A und B befinden.

Der Knoten B empfängt die Schlüsselteile und konstruiert daraus den Schlüssel key. Damit verfügen A und B über den gemeinsamen Schlüssel key.

MPX bietet eine höheres Sicherheitsniveau gegenüber Insider-Angreifern als SPX: da nur Schlüsselteile versendet werden, muss sich ein Angreifer, um erfolgreich zu sein, auf allen verwendeten Overlay-Pfaden befinden. Nur dann kann er alle Schlüsselteile abhören und den Schlüssel damit rekonstruieren. Gegenüber SPX hat MPX jedoch noch einen weiteren Vorteil:

Da Knoten A den Schlüsselaustausch mit einer gewissen Koordinate im SCAN vornimmt, und nur der Knoten, der die entsprechende Zone im SCAN verwaltet, alle Schlüsselteile erhält, kann durch einen MPX-Schlüsselaustausch authentifiziert werden, ob ein Knoten eine bestimmte Koordinate im SCAN verwaltet. Dadurch wird es insbesondere möglich, während der Lookup- und Insert-Operation eine *Authentifizierung des speichernden Knotens* vorzunehmen, womit Eigenschaft E10 (Authentifizierung Zonenverwalter) von SCAN erreicht wird.

Damit ein Angreifer diese Authentifizierung angreifen könnte, müsste er über mindestens d korrumpierte Sensorknoten verfügen, die sich alle im Overlay auf jeweils einem der d disjunkten Pfade befinden. Mit Formel 5.3 wurde die Erfolgswahrscheinlichkeit von SPX (p_{spx}) angegeben, also die Wahrscheinlichkeit, dass sich kein Angreifer auf einem Overlay-Pfad von der Quelle zum Ziel befindet. Die Wahrscheinlichkeit, dass sich mindestens ein Angreifer auf einem Overlay-Pfad von der Quelle zum Ziel befindet ist damit $1 - p_{spx}$. Betrachtet man nun d zonendisjunkte Pfade, so ist die

Wahrscheinlichkeit, dass sich ein zusammenarbeitender Angreifer auf allen d Pfaden befindet damit:

$$p_{angriff} = (1 - p_{spx})^d = (1 - (1 - m)^{\frac{d}{4}N^{\frac{1}{d}}})^d \tag{5.4}$$

Wie in Formel 5.3 steht dabei m für den Anteil an Angreifern von den N gesamt im SCAN vorhandenen Knoten. Die Wahrscheinlichkeit $p_{angriff}$ gibt also an, wie wahrscheinlich ein erfolgreicher Angriff ist. Die Erfolgswahrscheinlichkeit der Authentifizierung mittels MPX ist die Umkehrwahrscheinlichkeit, also:

$$p_{auth} = 1 - p_{angriff} = 1 - (1 - (1 - m)^{\frac{d}{4}N^{\frac{1}{d}}})^d \tag{5.5}$$

In Abbildung 5.18 wird die Erfolgswahrscheinlichkeit p_{auth} der Authentifizierung für verschiedene Gesamtknotenanzahlen N und $d = 3$ gezeigt, also die Wahrscheinlichkeit, dass es einem zusammenarbeitenden Angreifer nicht gelingt, sich fälschlicherweise als Zonenverwalter einer Zone auszugeben, in der ein gewisser Punkt liegt. Insbesondere ist in der Abbildung zu erkennen, dass für weniger als 10% Angreifer die Wahrscheinlichkeit für eine erfolgreiche Authentifizierung bei nahezu 100% liegt. Abbildung 5.19 und Abbildung 5.20 zeigen p_{auth} für $d = 5$ bzw. $d = 7$.

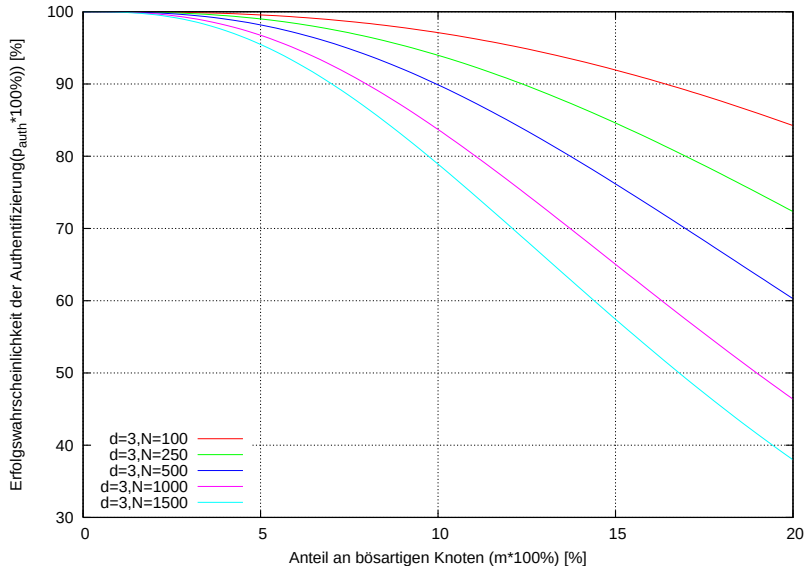


Abbildung 5.18: Erfolgswahrscheinlichkeit der Authentifizierung mittels MPX, d=3

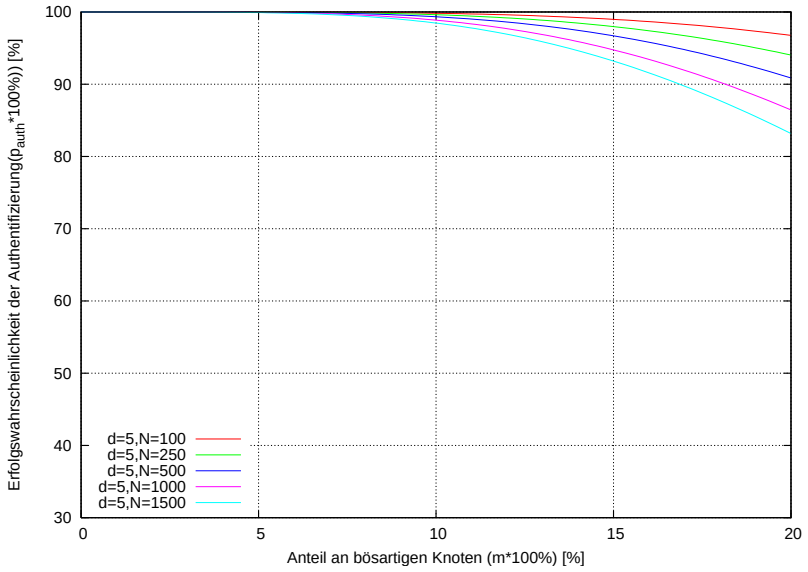


Abbildung 5.19: Erfolgswahrscheinlichkeit der Authentifizierung mittels MPX, d=5

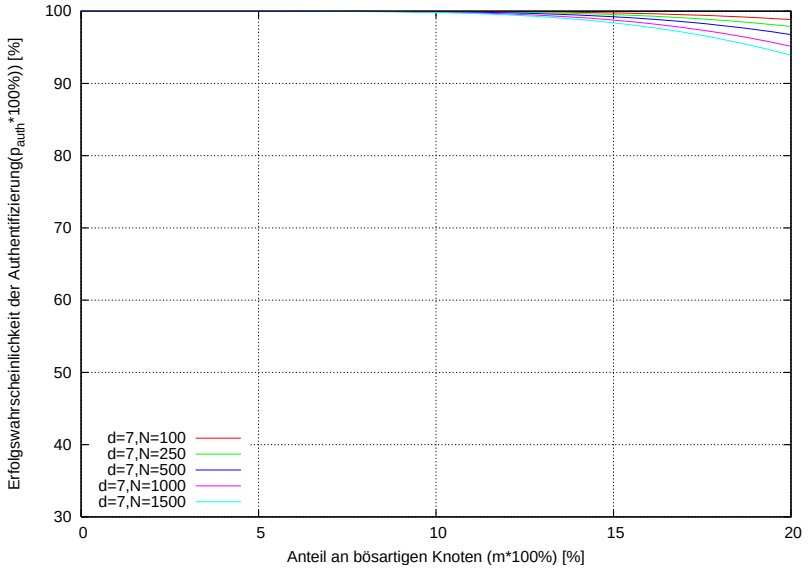


Abbildung 5.20: Erfolgswahrscheinlichkeit der Authentifizierung mittels MPX, d=7

Allerdings ist ein Schlüsselaustausch mittels MPX anfällig gegen Manipulation von Nachrichten: schon eine einzige manipuliere Nachricht reicht aus, dass der Empfänger den Schlüssel nicht mehr erfolgreich rekonstruieren kann. Somit kann ein Angreifer den Schlüsselaustausch verhindern. Um dieses Problem zu umgehen können spezielle Geheimnisteilungsverfahren, so genannte (k,n)-Schwellenwertverfahren eingesetzt werden (siehe Abschnitt 2.2.5). Bei einem (k,n)-Schwellenwertverfahren wird ein Schlüssel in n Teile geteilt. Um den Schlüssel aus den Schlüsselteilen zu rekonstruieren sind allerdings lediglich k Schlüsselteile notwendig, wobei gilt $k \leq n$.

Sei X eine Zufallsvariable, die angibt, auf wie vielen Overlay-Routing-Pfade zwischen zwei Knoten sich kein Angreifer befindet. Die Wahrscheinlichkeit p_{spx} aus Formel 5.3 gibt die Wahrscheinlichkeit an, dass sich auf einem einzigen Overlay-Routing-Pfad kein Angreifer befindet. Damit gilt:

$$\begin{aligned} P(X = c) &= \binom{d}{c} (p_{spx})^c * (1 - p_{spx})^{d-c} \\ &= \binom{d}{c} ((1-m)^h)^c (1 - (1-m)^h)^{d-c} \end{aligned}$$

Dabei sei m der Anteil an bösartigen Knoten im SCAN und h die mittlere Pfadlänge eines Overlay-Routing-Pfads. In SCAN beträgt die Wahrscheinlichkeit p_{mpx}, dass mindestens k der d möglichen Pfade nur aus gutartigen Knoten bestehen damit:

$$p_{mpx} = \sum_{i=k}^{d} P(X = i) \tag{5.6}$$

$$= \sum_{i=k}^{d} \binom{d}{i} ((1-m)^h)^i (1 - (1-m)^h)^{d-i} \tag{5.7}$$

$$\tag{5.8}$$

Die Wahrscheinlichkeit p_{mpx}, dass mit MPX erfolgreich ein Sitzungsschlüssel etabliert werden kann, beträgt in einem d-dimensionalen SCAN mit der Pfadlänge h aus Gleichung 5.1 somit:

$$p_{mpx} = \sum_{i=k}^{d} \binom{d}{i} ((1-m)^{\frac{d}{4} N^{\frac{1}{d}}})^i (1 - (1-m)^{\frac{d}{4} N^{\frac{1}{d}}})^{d-i} \tag{5.9}$$

Dabei ist N die Gesamtzahl an Knoten im SCAN. Abbildung 5.21 zeigt die Erfolgswahrscheinlichkeit p_{mpx} ($p_{mpx} \in [0, 1]$) eines MPX-Schlüsselaustauschs für ein SCAN der Dimension 4 unter Verwendung eines (3,4)-Schwellenwertverfahrens in einem SCAN mit 100, 250, 500, 1000 und 1500 Knoten.

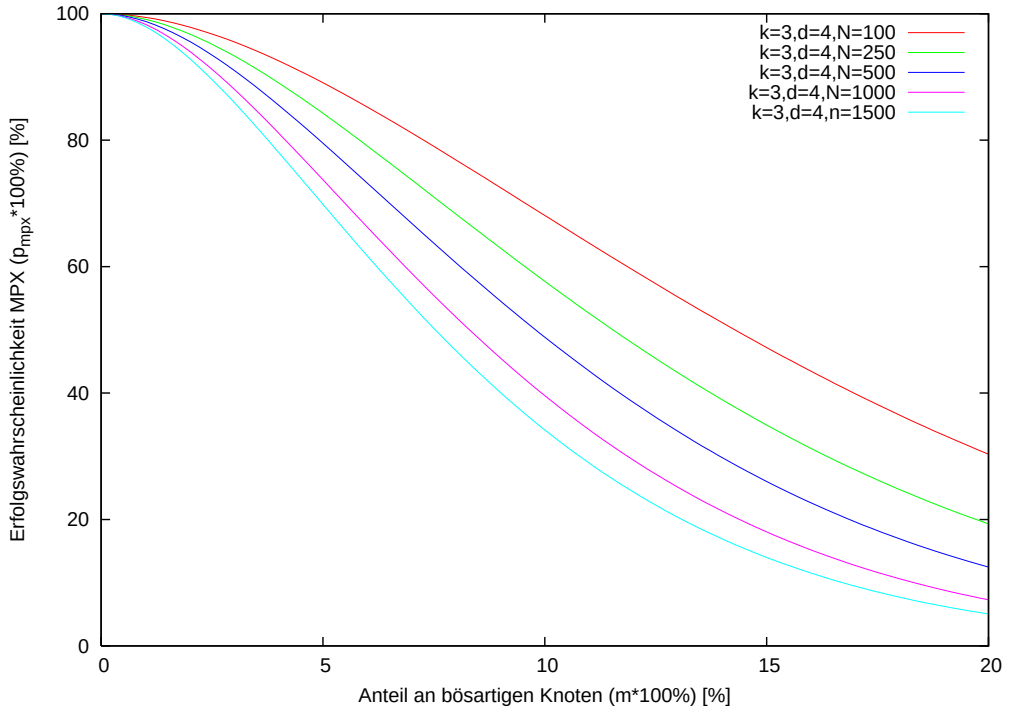


Abbildung 5.21: Erfolgswahrscheinlichkeit eines MPX-Schlüsselaustauschs für ein SCAN der Dimension 4 unter Verwendung eines (3,4)-Schwellenwertverfahrens für die Geheimnisteilung

Bei Einsatz eines (k,n)-Schwellenwertverfahrens besteht das Problem, dass ein Angreifer den Schlüsselaustausch sabotieren kann, indem er manipulierte Schlüsselteile übermittelt. Zwar ist schon für $k \leq n$ der Schlüsselteile eine Rekonstruktion möglich, allerdings ist für den Empfänger der Schlüsselteile nicht ersichtlich, welche Schlüsselteile manipuliert wurden und welche nicht. Um dieses Problem zu umgehen muss ein verifizierendes Verfahren eingesetzt werden, wie bereits in Abschnitt 2.2.5 beschrieben. In SCAN wird die Verifizierung von Schlüsselteilen dadurch erreicht, dass in jeder `MPX`-Nachricht nicht nur ein Schlüsselteil key_i übertragen wird, sondern auch ein so genanntes Committment c_j, das sich auf ein über einen anderen Pfad versendetes Schlüsselteil key_j bezieht. Das Committment berechnet sich folgendermaßen:

$$c_j = hash(key_j)$$

Die entsprechenden MPX-Nachrichten sehen folgendermaßen aus:

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_1, \mathrm{c}_2]}$$

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_2, \mathrm{c}_3]}$$

...

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_{\mathrm{d}-1}, \mathrm{c}_\mathrm{d}]}$$

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{B} : \mathbf{MPX}[\mathrm{key}_\mathrm{d}, \mathrm{c}_1]}$$

Der Knoten B, der die Schlüsselteile wieder zusammensetzt, kann dann für jedes Schlüsselteil überprüfen, ob Schlüsselteil und Committment zusammenpassen oder nicht. Falls dies nicht der Fall ist, werden die entsprechenden Schlüsselteile nicht dazu verwendet, den Schlüssel zu rekonstruieren.

Abbildung 5.22 vergleicht die Erfolgswahrscheinlichkeit eines Schlüsselaustauschs mit SPX und MPX. Deutlich sichtbar wird, dass ein Schlüsselaustausch mittels MPX einem Schlüsselaustausch mit SPX bis zu einem gewissen Anteil von Angreifern (hier: ca. 7%) überlegen ist, danach aber deutlich schlechter ist. Die Überlegenheit von MPX ist bei wenigen Angreifern damit zu erklären, dass bei einem (3,4)-Schwellenwertverfahren ein einziger Overlay-Routing-Pfad, auf dem sich ein Angreifer befindet, toleriert wird. Bei SPX führt dieser Pfad unweigerlich zum Mißerfolg. Ab einem gewissen Anteil von Angreifern ist es aber nicht mehr möglich, genügend Overlay-Routing-Pfade ohne Angreifer zu erwischen. Hier ist SPX überlegen, weil nur ein einziger Overlay-Routing-Pfad verwendet wird. Allerdings ist beim Vergleich von SPX und MPX zu beachten, dass MPX gegenüber SPX mehr leistet, da MPX auch die Authentifizierung eines Zonenverwalters ermöglicht.

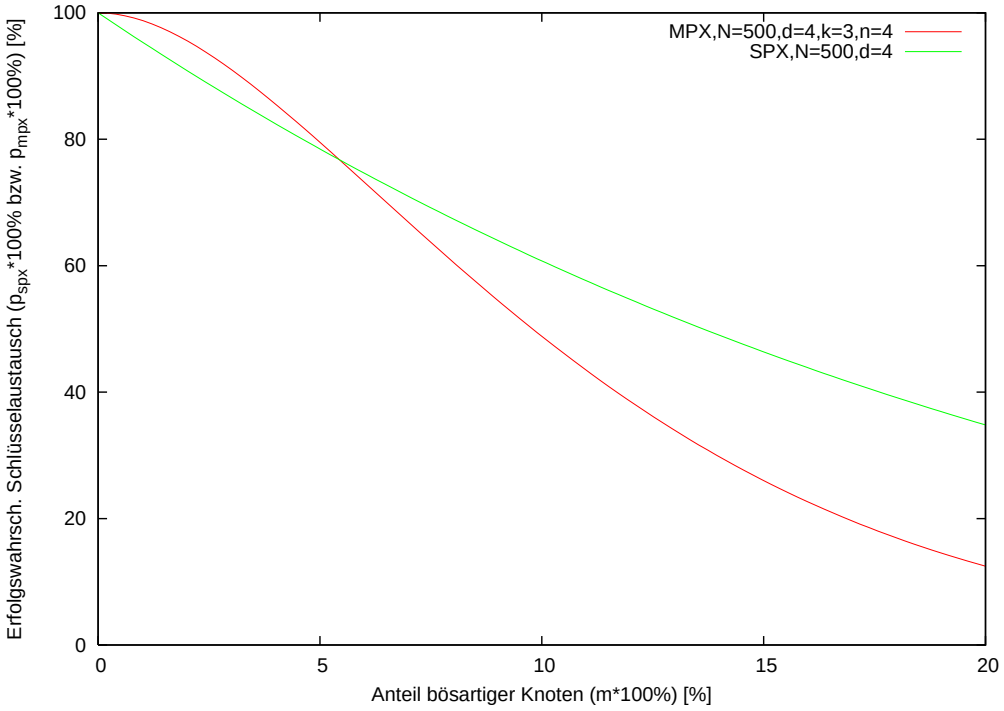


Abbildung 5.22: Vergleich der Erfolgswahrscheinlichkeiten von SPX und MPX in einem SCAN mit 500 Knoten und der Dimension 4. Für MPX kommt ein (3,4)-Schwellenwertverfahren zum Einsatz.

5.7.2.1 Sicherheitsbetrachtung

Zuallererst erlaubt MPX den Austausch eines Schlüssel zwischen beliebigen Knoten, wobei Angriffe von Outsider-Angreifern verhindert werden und Angriffe von Insider-Angreifern deutlich erschwert werden. Mit den ausgetauschten Schlüsseln können beliebige Knoten Integrität und Vertraulichkeit schützen, weswegen MPX für die Eigenschaft E5 (Integrität und Vertraulichkeit zwischen beliebigen Knoten) verantwortlich ist.

Weiterhin ermöglicht es MPX zu verifizieren, ob ein Knoten eine Zone verwaltet, in der ein gewisser Punkt liegt, da nur dieser Knoten alle Schlüsselteile erhält und den Schlüssel rekonstruieren kann. MPX zeichnet sich also auch für Eigenschaft E10 (Authentifizierung Zonenverwalter) verantwortlich.

Ebenso wie bei SPX kann mittels MPX die Lookup- und Insert-Operation effizienter gestaltet werden, da nur noch ein Schlüssel (bzw. Schlüsselteile) über das Overlay versendet werden muss, während die restliche Kommunikation auf Vermittlungsschicht geschieht. Im Folgenden wird beschrieben, wie die Lookup- und Insert-Operation mit MPX gestaltet werden.

5.7.2.2 Lookup mit MPX

Abbildung 5.23 zeigt die Lookup-Operation unter Verwendung von Multi-Path-Key-Exchange im Überblick.

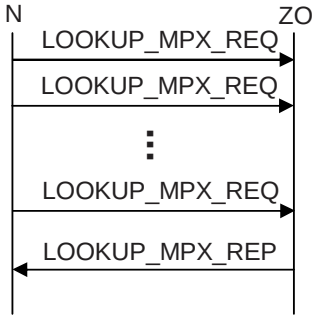


Abbildung 5.23: Ablauf der Lookup-Operation bei Verwendung von Multi-Path-Key-Exchange (SPX)

Ein Dienstnutzer N ruf die Funktion `lookup_mpx(Dienstname)` auf, um eine Liste mit Dienstbeschreibungen zu einem Dienstnamen zu erhalten. Bei Einsatz eines (k,n)-Schwellenwertverfahrens für MPX ($k \leq n \leq d$) werden n Overlay-Routing-Pfade verwendet. Wie üblich wird aus dem Dienstnamen durch Berechnung des Hash-Werts eine Koordinate im SCAN bestimmt:

$$h_1 = hash(Dienstname)$$

Außerdem wird zufällig ein Schlüssel key erzeugt und in n Schlüsselteile key_1, key_2, ..., key_n aufgeteilt. Anschließend werden folgende Nachrichten über das Overlay versendet. Um die disjunkten Pfade für die einzelnen Pfade zu erzeugen werden die einzelnen Nachrichten im ersten Hop des Overlay-Routings jeweils an verschiedene Nachbarn weitergegeben.

$$1.1)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_MPX\_REQ}[\mathrm{key}_1, \mathrm{c}_2, \mathrm{addr}_\mathrm{N}]$$

$$1.2)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_MPX\_REQ}[\mathrm{key}_2, \mathrm{c}_3, \mathrm{addr}_\mathrm{N}]$$

...

$$1.n-1)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_MPX\_REQ}[\mathrm{key}_{\mathrm{n}-1}, \mathrm{c}_\mathrm{n}, \mathrm{addr}_\mathrm{N}]$$

$$1.n)\mathrm{N} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{LOOKUP\_MPX\_REQ}[\mathrm{key}_\mathrm{n}, \mathrm{c}_1, \mathrm{addr}_\mathrm{N}]$$

Jede Nachricht enthält einen Teil des Schlüssels (key_i), ein Committment auf ein anderes Schlüsselteil (c_j) und die Vermittlungsschichtadresse des Knoten N ($addr_N$). Die `LOOKUP_MPX_REQ`-Nachricht wird mittels Overlay-Routing versendet, d. h. Integrität und Vertraulichkeit sind gegen Angriffe von Outsider-Angreifern geschützt. Insider-Angreifer können sowohl Schlüsselteile als auch die Committments manipulieren.

Die Nachricht erreicht schließlich denjenigen Zonenverwalter ZO, in dessen Zone h_1 liegt. Der Zonenverwalter erhält die Schlüsselteile key_1, key_2, ...,key_n sowie die Committments c_1, c_2,...,c_n. Der Zonenverwalter verifiziert die Schlüsselteile mittels der entsprechenden Committments und berechnet aus den Schlüsselteilen den Schlüssel key.

$$2.)\mathrm{ZO} \overset{\mathrm{NR}}{\rightarrow} \mathrm{N} : \textbf{LOOKUP\_MPX\_REP}[\mathrm{E}_{\mathrm{key}}(\mathrm{DB}-\mathrm{LISTE}), \mathrm{mac}]$$

Im zweiten Schritt sendet der Knoten ZO auf Vermittlungsschicht eine `LOOKUP_MPX_REP`-Nachricht an Knoten N. Die dazu benötigte Vermittlungsschichtadresse $addr_N$ ist aus den entsprechenden `LOOKUP_MPX_REQ`-Nachrichten bekannt.

Die Nachricht besteht aus der Liste von Dienstbeschreibungen ($DB - LISTE$), die der Zonenverwalter in seinem Teil der verteilten Hash-Tabelle unter dem Hash-Wert h_1 abgespeichert hat.

Die `LOOKUP_MPX_REP`-Nachricht ist mit dem zuvor ausgetauschten Schlüssel key verschlüsselt um die Vertraulichkeit der Nachricht zu schützen. Zum Schutz der Integrität der Nachricht kommt der Message Authentication Code mac zum Einsatz, zu dessen Erzeugung ebenfalls der Schlüssel key verwendet wird.

Bei Lookup mit MPX wird der Timer `LOOKUP_MPX_REQ_TIMER` mit Versenden der `LOOKUP_MPX_REQ`-Nachrichten gestartet. Der Timer läuft nach der Zeitdauer `LOOKUP_MPX_REQ_TIMEOUT` ab, falls nicht zuvor die `LOOKUP_MPX_REP`-Nachricht empfangen wurde. Der Dienstnehmer kann entsprechend die `LOOKUP_MPX_REQ`-Nachrichten erneut verschicken. Insbesondere kann er dabei auch andere Pfade wählen, falls zuvor nicht alle möglichen Pfade verwendet wurden. Damit steigt die Erfolgswahrscheinlichkeit, da unter Umständen zuvor Pfade mit zu vielen Angreifern gewählt wurden, die aber den Schlüsselaustausch nicht infiltrieren konnten.

Eine Realisierung der Lookup-Operation mittels MPX kann zu einer deutlichen Reduktion des Kommunikationsaufwands gegenüber dem bisher Entwurf der Lookup-Operation ohne Schlüsselaustausch führen, falls die Liste der Dienstbeschreibungen deutlich länger ist als die Summe der Länge der `LOOKUP_MPX_REQ`-Nachrichten, da in diesem Fall die Liste der Dienstbeschreibungen nicht aufwendig über das Overlay transportiert werden muss, sondern wesentlich effizienter auf Vermittlungsschicht transportiert werden kann. Darüber hinaus bietet die Lookup-Operation mittels MPX die Sicherheit, dass wirklich der Knoten antwortet, der auch für die Verwaltung der entsprechenden gesuchten Dienstbeschreibungen zuständig ist. Lookup mittels MPX erschwert Angriffe von Insider-Angreifern, die auf Lookup-Anfragen mit gefälschten Dienstbeschreibungen antworten, obwohl sie nicht die Zone verwalten, in welcher die gewünschten Dienstbeschreibungen liegen. Außerdem erlangen Insider-Angreifer keine Kenntnis über den Inhalt von abgefragten Dienstbeschreibungen, da diese verschlüsselt auf Vermittlungsschicht übertragen werden. Lookup mittels MPX ist also verantwortlich für die Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) von SCAN.

5.7.2.3 Insert mit MPX

Abbildung 5.24 zeigt den Ablauf der Insert-Operation mit MPX.

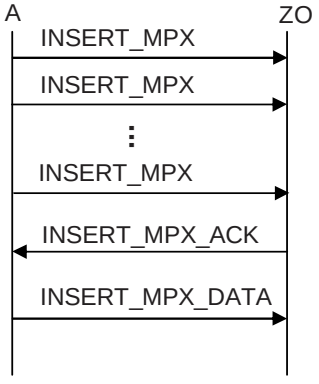


Abbildung 5.24: Ablauf der Insert-Operation bei Verwendung von Multi-Path-Key-Exchange (MPX)

Ein Dienstanbieter A ruft die Funktion `insert_mpx(Dienstname, Dienstbeschreibung)` auf, um eine Dienstbeschreibung unter einem Dienstnamen abzuspeichern. Bei

Einsatz eines (k,n)-Schwellenwertverfahrens für MPX ($k \leq n \leq d$) werden n Overlay-Routing-Pfade verwendet. Auch bei Insert mittels MPX wird aus dem Dienstnamen durch Berechnung eines Hash-Werts eine Koordinate im SCAN berechnet, an der die zugehörige Dienstbeschreibung gespeichert werden soll:

$$h_1 = hash(Dienstname)$$

Wie oben beschrieben wird ein Schlüssel key erzeugt und in n Schlüsselteile key_1, key_2, ...,key_n aufgeteilt. Diese Schlüsselteile werden mittels folgender Nachrichten über das Overlay versendet:

$$\boxed{1.1)\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT\_MPX}[\mathrm{key}_1, \mathrm{c}_2, \mathrm{addr}_\mathrm{A}]}$$

$$\boxed{1.2)\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT\_MPX}[\mathrm{key}_2, \mathrm{c}_3, \mathrm{addr}_\mathrm{A}]}$$

...

$$\boxed{1.2)\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT\_MPX}[\mathrm{key}_{\mathrm{n}-1}, \mathrm{c}_\mathrm{n}, \mathrm{addr}_\mathrm{A}]}$$

$$\boxed{1.n)\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{INSERT\_MPX}[\mathrm{key}_\mathrm{n}, \mathrm{c}_1, \mathrm{addr}_\mathrm{A}]}$$

Eine `INSERT_MPX`-Nachricht enthält ebenso wie die `LOOKUP_MPX_REQ`-Nachricht einen Schlüsselteil (key_i) des Schlüssels key, ein Committment c_i auf ein weiteres Schlüsselteil und die Vermittlungsschichtadresse des Knotens A ($addr_A$). Die `INSERT_MPX`-Nachrichten erreichen schließlich den Knoten ZO, der die Zone verwaltet, in der h_1 liegt.

Die `INSERT_MPX`-Nachrichten werden über das Overlay-Routing versendet. Dadurch sind Integrität und Vertraulichkeit der Nachricht gegen Angriffe von Outsidern geschützt.

$$\boxed{2.)\mathrm{ZO} \overset{\mathrm{NR}}{\rightarrow} \mathrm{A} : \textbf{INSERT\_MPX\_ACK}[\mathrm{E}_{\mathrm{key}}(\mathrm{addr}_{\mathrm{ZO}}), \mathrm{mac}]}$$

Der Zonenverwalter ZO rekonstruiert aus den Schlüsselteilen key_1, key_2, ..., key_n den Schlüssel key und antwortet auf Vermittlungsschicht mit einer `INSERT_MPX_ACK`-Nachricht, welche die Adresse des Knoten ZO beinhaltet ($addr_{ZO}$). Die Vertraulichkeit dieser Nachricht ist mittels Verschlüsselung unter dem Schlüssel key geschützt. Integritätsschutz der Nachricht wird durch den Message Authentication Code mac erreicht, in dessen Konstruktion ebenfalls der Schlüssel key einfließt.

3.)A $\overset{\text{NR}}{\rightarrow}$ ZO : **INSERT_MPX_DATA**[E_{key}(DB)]

Im dritten Schritt überträgt der Dienstanbieter A dann auf Vermittlungsschicht die Dienstbeschreibung DB an den Zonenverwalter ZO.

Bei Insert mit MPX wird der Timer `INSERT_MPX_TIMER` mit Versenden der `INSERT_MPX`-Nachrichten gestartet. Der Timer läuft nach der Zeitdauer `INSERT_MPX_TIMEOUT` ab, falls nicht zuvor die `INSERT_MPX_ACK`-Nachricht empfangen wurde. Der Dienstnehmer kann entsprechend die `INSERT_MPX`-Nachrichten erneut versenden. Insbesondere kann er dabei auch andere Pfade wählen, falls zuvor nicht alle möglichen Pfade verwendet werden. Damit steigt die Erfolgswahrscheinlichkeit an, da unter Umständen zuvor Pfade mit zu vielen Angreifern gewählt wurden, die aber den Schlüsselaustausch nicht infiltrieren konnten.

Für die Sicherheit des Verfahrens gilt das bei Lookup mittels MPX gesagt. Insbesondere ist Insert mittels MPX ebenfalls verantwortlich für die Eigenschaft E11 (Schutz von Dienstbeschreibungen während der Übertragung) von SCAN.

5.8 Behandlung ausgefallener Knoten und Dienste

In Sensornetzen ist damit zu rechnen, dass regelmäßig Sensorknoten ausfallen. Dies kann z. B. durch Energiemangel oder durch physikalische Zerstörung des Sensorknotens geschehen. Fällt ein Knoten aus, so stehen nicht nur die Dienste, welche dieser Knoten anbietet, nicht mehr zur Verfügung, sondern die Zone, die dieser Knoten verwaltet, ist nicht mehr erreichbar. Dieser Knotenausfall muss behandelt werden, um SCAN *robust* zu machen. Es ist notwendig, für ausgefallene Zonen einen neuen Zonenverwalter zu bestimmen, sowie Dienstbeschreibungen von Diensten, die der ausgefallene Knoten angeboten hat, aus dem SCAN zu entfernen. Die beiden folgenden Abschnitte beschäftigen sich mit dieser Problematik.

5.8.1 Umgang mit Ausfall einer Zone

In diesem Abschnitt wird beschrieben, wie mit dem Ausfall einer Zone umgegangen werden kann, der durch den Ausfall eines Zonenverwalters entsteht. Ein Zonenausfall hat folgende Auswirkungen:

- Die Zone ist für die Nachbarn nicht mehr erreichbar.
- Alle in der ausgefallenen Zone gespeicherten Dienstbeschreibungen gehen verloren.
- Der Schlüssel $key_{MD,ZO}$, den das Master Device und der Zonenverwalter gemeinsam haben, geht verloren.

- Die Nachbarschlüssel key_{ZO,N_i} zwischen dem Zonenverwalter und seinen Nachbarn gehen verloren und damit wird Eigenschaft E4 (Integrität und Vertraulichkeit zwischen Nachbarn) von SCAN verletzt

Um mit diesem Ausfall umzugehen muss ein Nachfolger für die ausgefallene Zone gefunden werden. Dies geschieht in SCAN mittels einem von zwei Reparatur-Verfahren. An ein Reparatur-Verfahren werden folgende Anforderungen gestellt:

- Die Nachbarschlüssel sollen wiederhergestellt werden, damit die Eigenschaft E4 (Integrität und Vertraulichkeit zwischen Nachbarn) wieder gilt.
- Ein Reparatur-Verfahren soll wenig Kommunikationsaufwand erzeugen.
- Es soll verhindert werden, dass das Verfahren von einem Angreifer missbraucht werden kann. Insbesondere soll das Verfahren verhindern, dass ein Angreifer gezielt gewisse Zonen übernehmen kann. Es gilt also, die Eigenschaften E8 (Keine Manipulation der eigenen Zone) und E9 (Zonenverifikation) von SCAN zu erhalten.
- Eine Join-Operation mit einem Join-Point in der ausgefallenen Zone soll weiterhin möglich sein.
- Ein Reparatur-Verfahren muss ohne Hilfe des Master Device funktionieren, da dieses nur während des Beitritts eines Knotens mit dem SCAN kommunizieren kann und sonst für die Knoten des Netzes nicht erreichbar ist.

Um einen Knotenausfall behandeln zu können, müssen die Nachbarn einer ausgefallenen Zone den Ausfall zuallererst erkennen. Dies geschieht dadurch, dass für jeden Nachbarn ein Timer geführt wird, der angibt, wann die letzte Nachricht von diesem Nachbarn erhalten wurde. Wird während der Zeitdauer T_{alive} keine Nachricht empfangen, so versendet der Knoten A eine `PING`-Nachricht an den Nachbarn B, von dem Nachrichten ausgeblieben sind:

$$\boxed{\mathrm{A} \overset{\mathrm{NR}}{\rightarrow} \mathrm{B} : \mathbf{PING}[]}$$

Der Knoten B antwortet auf diese Nachricht mit einer `PONG`-Nachricht um anzuzeigen, dass er nicht ausgefallen ist:

$$\boxed{\mathrm{B} \overset{\mathrm{NR}}{\rightarrow} \mathrm{A} : \mathbf{PONG}[]}$$

Der Knoten A startet den Timer `PING_TIMER` mit Versenden der `PING`-Nachricht. Trifft innerhalb der Zeitdauer `PING_TIMEOUT` keine `PONG`-Nachricht ein, so wiederholt

der Knoten A die `PING`-Nachricht. Läuft der Timer erneut ab, so geht der Knoten A vom Ausfall des Nachbarn B aus.

Werden in SCAN regelmäßig Nachrichten über das Overlay gesendet, so erzeugt das Verfahren keinen zusätzlichen Kommunikationsaufwand, da in diesem Fall auch keine `PING`-Nachrichtten versendet werden.

Im Rahmen dieser Arbeit wurden zwei Verfahren entwickelt, um mit dem Ausfall einer Zone umzugehen: Explizite Zonenübergabe und das RMPX-Verfahren. Beide Verfahren etablieren neue Nachbarschlüssel und sorgen dafür, dass die ausgefallene Zone wieder verwaltet wird. Explizite Zonenübergabe und das RMPX-Verfahren werden im Folgenden beschrieben.

5.8.1.1 Explizite Zonenübergabe

Mittels expliziter Zonenübergabe wird ein Zonenverwalter dazu befähigt, eine von ihm verwaltete Zone an den Zonenverwalter einer benachbarten Zone zu übergeben. Eine solche Zonenübergabe ist dann sinnvoll, wenn ein Zonenverwalter erkennen kann, dass er bald ausfallen wird, z. B. weil er ein vorausschauendes Energiemanagement verwendet, wie in [33] beschrieben. Im Folgenden wird der Zonenverwalter, der seine Zone abgeben möchte, als Releasing Node RN bezeichnet. Derjenige Zonenverwalter, der die Zone übernehmen soll, wird als Substituting Node SN bezeichnet.

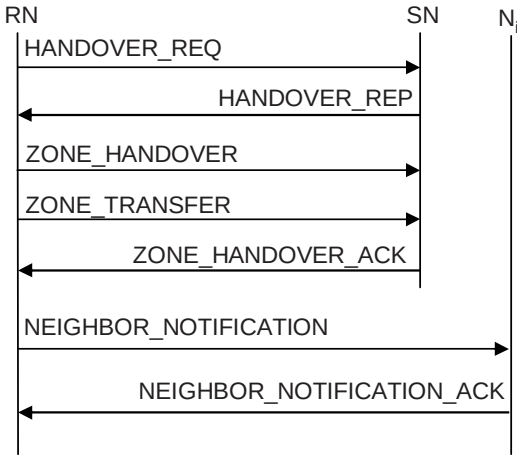


Abbildung 5.25: Ablauf der expliziten Zonenübergabe

Zuerst muss sich ein Zonenverwalter für einen SN entscheiden. Er wählt dazu denjenigen seiner Nachbarn, der die kleinste Zone verwaltet. Existieren mehrere kleinste Zonen mit gleicher Größe, so wählt der Zonenverwalter eine Zone zufällig. Die Zonengröße der Nachbarzonen ist dem RN aus seiner Nachbarliste bekannt. Die kleinste Zone wird gewählt, weil davon ausgegangen wird, dass die Last, die ein Zonenverwalter durch SCAN hat, proportional zur Zonengröße ist. Dem liegt die Überlegung zugrunde, dass eine Hash-Funktion eine Menge von Dienstnamen gleichverteilt auf das SCAN abbildet, und deshalb eine größere Zone im Mittel auch mit mehr Dienstbeschreibungen einhergeht. Abbildung 5.25 zeigt eine Übersicht über die explizite

Zonenübergabe.

1.)RN $\overset{\text{NR}}{\rightarrow}$ SN :**HANDOVER_REQ**$[Nonce, mac]$

2.)SN $\overset{\text{NR}}{\rightarrow}$ RN :**HANDOVER_REP**$[Flag, Nonce, mac]$

Mit der `HANDOVER_REQ` teilt der RN dem SN mit, dass er wünscht, seine Zonen zu übergeben. Der SN antwortet mit einer `HANDOVER_REP`-Nachricht, in der ein Flag enthalten ist, mit dem der SN Zustimmung zur Zonenübergabe (ACK) bzw. Ablehnung (NACK) signalisieren kann. Ein SN würde eine Zonenübergabe z. B. ablehnen, wenn er selbst mit einem baldigen Ausfall rechnet. In den beiden Nachrichten wird eine Nonce zur Vermeidung von Angriffen durch Wiedereinspielen verwendet. Der Message Authentication Code mac dient zum Schutz der Integrität der Nachricht.

3.)RN $\overset{\text{NR}}{\rightarrow}$ SN :**ZONE_HANDOVER**$[E_{key_{RN,SN}}(NKL, E_{key_{MD,RN}}(zone_{RN})), mac]$

4.)RN $\overset{\text{NR}}{\rightarrow}$ SN :**ZONE_TRANSFER**$[E_{key_{RN,SN}}(SDRs), mac]$

5.)SN $\overset{\text{NR}}{\rightarrow}$ RN :**ZONE_HANDOVER_ACK**$[Nonce, mac]$

Die `ZONE_HANDOVER`-Nachricht ist die zentrale Nachricht der expliziten Zonenübergabe: in der Nachricht ist die Neighbor Key List NKL enthalten, die für jeden Nachbarn einen Schlüssel, die Vermittlungsschichtadresse sowie die Zonenbeschreibung enthält. Die `ZONE_HANDOVER`-Nachricht befähigt den SN also zur Kommunikation mit den Nachbarn der übergebenen Zone. In der Nachricht ist weiterhin ein Nachweis der expliziten Übergabe enthalten ($E_{key_{MD,RN}}(zone_{RN})$). Dieser wird später benötigt, um dem Master Device gegenüber nachzuweisen, dass der SN auch wirklich der rechtmäßige SN der übernommenen Zone ist. Für diesen Nachweis speichert der SN daneben auch noch die Vermittlungsschichtadresse des RN sowie dessen Zonenbeschreibung, die dem SN aus seiner Nachbarliste bekannt sind. Das Master Device benötigt diese beiden Werte später, um den Schlüssel $key_{MD,RN}$ mittels dynamischer Schlüsselberechnung wiederherzustellen. Integrität und Vertraulichkeit der `ZONE_HANDOVER`-Nachricht sind durch Verschlüsselung und Message Authentication Code mac geschützt.

Die `ZONE_TRANSFER`-Nachricht ist optional. Sie dient zur Übertragung von Dienstbeschreibungen ($SDRs$), die in der übergebenen Zone gespeichert sind. Die Nachricht wurde bereits bei der Join-Operation beschrieben. Der SN bestätigt die erfolgreiche Zonenübergabe schließlich mit der `ZONE_HANDOVER_ACK`-Nachricht. Die Integrität dieser Nachricht ist durch den Message Authentication Code mac geschützt. Die

Nonce dient zum Schutz gegen Angriffe durch Wiedereinspielen.

6.)RN $\overset{\text{NR}}{\rightarrow}$ $\text{N}_{\text{i,RN}}$:
NEIGHBOR_NOTIFICATION$[E_{key_{RN,N_{i,RN}}}(addr_{SN}, Nonce), mac]$

7.)$\text{N}_{\text{i,RN}}$ $\overset{\text{NR}}{\rightarrow}$ RN :**NEIGHBOR_NOTIFICATION_ACK**$[Nonce, mac]$

Schließlich müssen noch die Nachbarn über die Zonenübergabe benachrichtigt werden. Dazu wird die `NEIGHBOR_NOTIFICATION`-Nachricht verwendet. Diese Nachricht beinhaltet die Vermittlungsschichtadresse $addr_{SN}$ des SN. Mit dieser Adresse ersetzten die Nachbarn in ihrer Nachbarliste die Vermittlungsschichtadresse des RN. Anschließend bestätigen die Nachbarn die Modifikation ihrer Nachbarliste mittels der `NEIGHBOR_NOTIFICATION_ACK`-Nachricht. Die `NEIGHBOR_NOTIFICATION`- und die `NEIGHBOR_NOTIFICATION_ACK`-Nachricht sind beide mittels eines Message Authentication Codes (*mac*) integritätsgeschützt. Verschlüsselung der `NEIGHBOR_NOTIFICATION`-Nachricht schützt die Vertraulichkeit dieser Nachricht. Die *Nonce* kommt zum Einsatz, um Schutz gegen Angriffe durch Wiedereinspielen zu erreichen.

Im Rahmen der Expliziten Zonenübergabe verwendet der Knoten RN die Timer: `HANDOVER_REQ_TIMER`, `ZONE_HANDOVER_TIMER` und mehrere `NEIGHBOR_NOTTIFICATION_TIMER`, die jeweils nach der Zeitdauer `HANDOVER_REQ_TIMEOUT`, `ZONE_HANDOVER_TIMEOUT` bzw. `NEIGHBOR_NOTIFICATION_TIMEOUT` ablaufen, falls nicht vorher die `HANDOVER_REP`-, `ZONE_HANDOVER_ACK`- oder `NEIGHBOR_NOTIFICATION_ACK`-Nachricht erhalten wurde. Falls einer dieser Timer abläuft, wird die entsprechende Nachricht erneut gesendet.

Die explizite Zonenübergabe verursacht wenig Kommunikationsaufwand: In einem SCAN mit d Dimensionen hat jede Zone im Durchschnitt $2d$ Nachbarn. Bei der explizite Zonenübergabe werden alle Nachrichten an benachbarte Zonen versendet, der Overlay-Routing-Pfad jeder Nachricht hat also die Länge 1. Damit kann der Kommunikationsaufwand *ohop*, gemessen in der Anzahl von notwendigen Overlay-Hops, berechnet werden. Bei der Berechnung bezeichnet $hops_i$ die Anzahl von Overlay-Hops im i-ten Schritt des Protokolls der expliziten Zonenübernahme:

$$ohop = \sum_{i=1}^{7} hops_i$$

Die Nachrichten `HANDOVER_REQ`, `HANDOVER_REP`, `ZONE_HANDOVER`, `ZONE_TRANSFER` und `ZONE_HANDOVER_ACK` benötigen jeweils einen Overlay-Hop, da sie direkt an einen einzigen Nachbarn versendet werden, also gilt $hops_1 = hops_2 = hops_3 = hops_4 = hops_5 = 1$. Die Nachricht `NEIGHBOR_NOTIFICATION` wird an alle Nachbarn verschickt. Die Nachbarn antworten alle mit einer `NEIGHBOR_NOTIFICATION_ACK`-Nachricht. Da eine Zone im Durchschnitt $2d$ Nachbarn hat gilt: $hops_6 = hops_7 = 2d$. Also berechnet sich der Kommunikationsaufwand zu:

$$ohop = 1 + 1 + 1 + 1 + 1 + 2d + 2d = 4d + 5$$

Da im Fall der expliziten Zonenübergabe die Nachbarschlüssel an den SN übergeben werden, gilt Eigenschaft E3 (Integrität und Vertraulichkeit zwischen Nachbarn) auch nach der expliziten Zonenübergabe. Da SN alle Daten erhält, die er für die Verwaltung der von RN aufgegebenen Zone benötigt, und diese Informationen insbesondere über den Nachweis vom Master Device überprüft werden können, gilt Eigenschaft E9 (Zonenverifikation). Da keine Änderungen an der Zone von SN oder an der übernommenen Zone vorgenommen werden gilt Eigenschaft E8 (Keine Manipulation der eigenen Zone).

Allerdings kann die explizite Zonenübergabe nur in den Fällen eingesetzt werden, in denen der Ausfall einer Zone vorhersagbar sind. Alle anderen Fälle werden durch das RMPX-Verfahren zur Reparatur von Zonenausfällen abgehandelt. Dieses wird im Folgenden beschrieben.

5.8.1.2 Reparatur eines Knotenausfalls mittels MPX (RMPX)

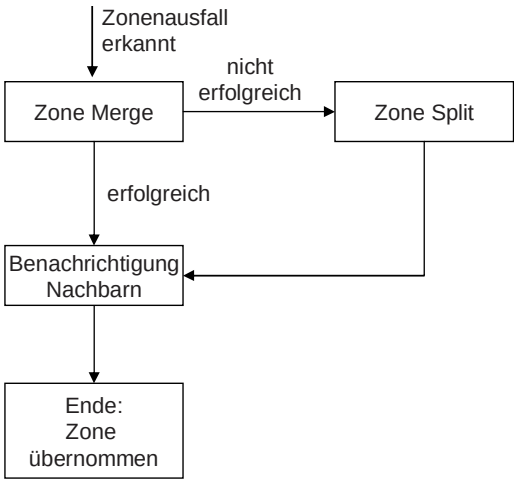
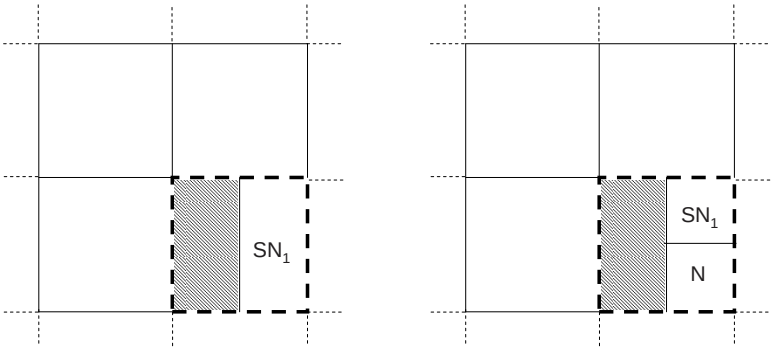


Abbildung 5.26: Ablauf der Zonenübernahme mit RMPX

Wird der Ausfall eines Nachbarn registriert, so geht der Knoten im SCAN-Protokollautomat (siehe Abbildung 5.1) in den Zustand „RMPX läuft“ über, was insbesondere bedeutet, dass er keine Join-Operationen mehr zulässt, welche einen Join Point in der eigenen Zone verwenden. Dadurch wird die Umgebung des ausgefallenen Knotens stabil gehalten, so dass das RMPX-Verfahren ohne Störung ablaufen kann. Das Master Device erkennt durch Timeout während der Join-Operation, dass ein Join nicht möglich ist und kann einen anderen Join Point wählen, der dann mit hoher Wahrscheinlichkeit in einer anderen Zone liegt. Nach Abschluss des RMPX-Verfahrens oder nach Ablauf einer gewissen Zeitspanne geht der Knoten im Protokollautomat von SCAN (siehe Abbildung 5.1) wieder zurück in den Zustand „Knoten ist Mitglied

im SCAN“. Abbildung 5.26 gibt einen Überblick über die Reparatur eines Knotenausfalls mittels RMPX. RMPX greift auf zwei Verfahren zurück: *Zone Merge* und *Zone Split*. Diese werden im Folgenden beschrieben.

- Das *Zone-Merge-Verfahren* greift auf eine Idee aus [77] zurück: Es wird versucht, diejenige Zone wiederherzustellen, aus der die ausgefallene Zone durch Teilung entstanden ist. Abbildung 5.27(a) zeigt ein Beispiel: die schraffierte Zone und die Zone des Zonenverwalters SN_1 sind aus der gestrichelt umrandeten Zone entstanden. Im Folgenden bezeichnen wir die beiden Zonen, die aus einer Zone $zone_{splitted}$ entstanden folgendermaßen: Die ausgefallene Zone wird als $zone_{failed}$ bezeichnet, die andere Hälfte der ursprünglichen Zone $zone_{splitted}$ wird als die zu $zone_{failed}$ *komplementäre Zone* $zone^c_{failed}$ bezeichnet. Nun fällt die schraffierte Zone ($zone_{failed}$) aus. Das Zone-Merge-Verfahren vereinigt die beiden Zonen wieder zu der gestrichelt umrandeten Zone ($zone_{splitted}$). Diese wird dann von SN_1 verwaltet. Das Zone-Merge-Verfahren ist effizient, falls die zweite Hälfte der Zone ($zone^c_{failed}$) noch existiert, im Beispiel die Zone von SN_1. Im Gegensatz zu [77] wird jedoch nicht versucht, weitere Join-Operationen in die Zone von SN_1 rückgängig zu machen, d. h. falls die Zone von SN_1 erneut geteilt wurde, wird das Zone-Merge-Verfahren nicht angewendet. Dies ist darin begründet, dass das in [77] beschriebene Verfahren in diesem Fall eine aufwendige Reorganisation der Overlay-Struktur vornimmt, die unter Umständen sehr viel Kommunikationsaufwand erfordert und die Wiederherstellung der Nachbarschlüssel erschwert. Abbildung 5.27(b) zeigt ein Beispiel, in dem das Zone-Merge-Verfahren nicht angewendet wird: diese Zonenaufteilung entsteht aus der Zonenaufteilung in Abbildung 5.27(a) durch eine Join-Operation des Knotens N, für den das Master Device einen Join Point in der Zone des Zonenverwalters SN_1 bestimmt hat. Die Zone von SN_1 wurde also ein weiteres mal geteilt. In diesem Fall wird das Zone-Split-Verfahren angewandt.

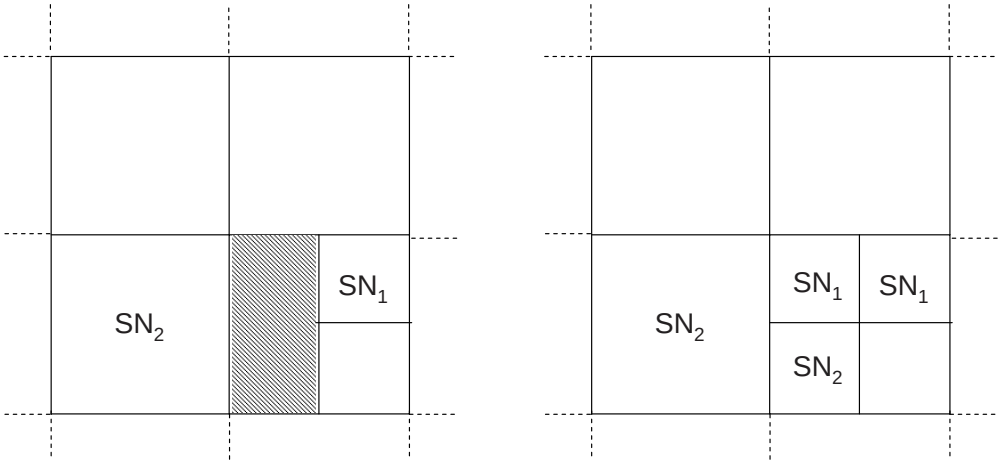


(a) Zonen Merge möglich (b) Kein Zone Merge

Abbildung 5.27: Zone Merge wird nur angewendet, wenn die benachbarte Zone, mit der die ausgefallene Zone vereinigt werden soll, nicht bereits wieder geteilt wurde

- Beim *Zone-Split-Verfahren* werden zwei Nachbarn bestimmt, die jeweils eine Hälfte der ausgefallenen Zone übernehmen. Abbildung 5.28(a) und 5.28(b)

zeigen dieses Verfahren: In Abbildung 5.28(a) ist der Zustand vor dem Zone-Split-Verfahren zu sehen: Die schraffierte Zone ist ausgefallen. Zu dieser Zone benachbart sind die Zonen der Zonenverwalter SN_1 und SN_2. Abbildung 5.28(b) zeigt die Situation nach dem Zone-Split-Verfahren: Die ausgefallene Zone ist in zwei Unterzonen aufgeteilt worden, die jeweils von SN_1 bzw. SN_2 verwaltet werden. Mittels dieser Aufteilung wird es einfach möglich, die Nachbarschlüssel neu einzurichten.



(a) Vor dem Zone Split (b) Nach dem Zone Split

Abbildung 5.28: Zone Split in einem zweidimensionalen SCAN

Der Ablauf des RMPX-Verfahrens, welches das Zone-Merge-Verfahren und das Zone-Split-Verfahren beinhaltet, ist wie folgt:

Im Rahmen des Zone-Merge-Verfahren bestimmen alle Nachbarn, die einen Ausfall erkannt haben, die zur ausgefallenen Zone $zone_{failed}$ komplementäre Zone $zone^c_{failed}$, d. h. $zone_{failed}$ und $zone^c_{failed}$ sind durch Teilung aus der Zone $zone_{splitted}$ entstanden. Die Bestimmung der Zone $zone_{splitted}$ ist rein rechnerisch möglich, da bei der Teilung der Zonen von einer Ordnung der Dimensionen ausgegangen wird und die Teilung immer in Reihenfolge dieser Ordnung geschieht. In einem SCAN der Dimension 2 mit den beiden Dimensionen x und y sowie der Ordnung $x \prec y$ würde die Teilung einer Zone zuerst in der Dimension x erfolgen. Die nächste Teilung in einer der Teilzonen würde dann in Dimension y erfolgen. In den daraus entstehenden Teilzonen würde die nächste Teilung in x-Dimension erfolgen und so weiter.

Weiterhin berechnet jeder Nachbar der ausgefallenen Zone einen Punkt P_1 im SCAN-Raum. Zur Berechnung in einem SCAN der Dimension d sei $N = (n_1, n_2, ..., n_d)$ die niederwertigste Ecke und $H = (h_1, h_2, ..., h_d)$ die höchstwertige Ecke der Zone $zone_{failed}$, d. h. $\forall i, 1 \leq i \leq d : n_i \leq h_i$. Damit berechnet sich der Punkt P folgendermaßen:

$$P_1 = (n_1 - 1, n_2 + 1, ..., n_d + 1)$$

Durch diese Berechnung wird ein Punkt in einer zur ausgefallenen Zone benachbarten Zone berechnet. Dieser Knoten wird im Folgenden als Koordinator K bezeichnet. Es ist die Aufgabe dieses Knotens zu überprüfen, ob die Zone $zone^c_{failed}$ bereits geteilt wurde, und somit das Zone Split Verfahren angestoßen werden muss. Falls die Zone nicht geteilt wurde, stellt der Koordinator K dem Zonenverwalter SN, der die ausgefallene Zone übernehmen wird, einen Nachweis aus, dass SN zur Übernahme der Zone berechtigt war.

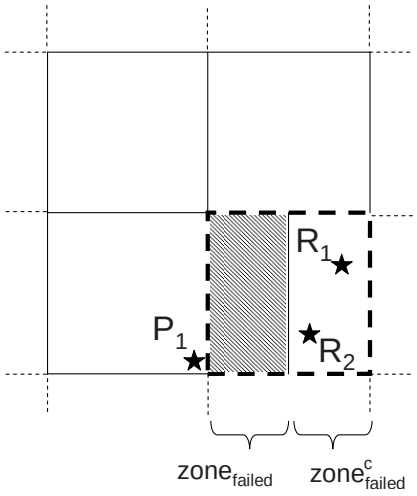


Abbildung 5.29: Wahl der Punkte P_1, R_1 und R_2 um zu überprüfen, ob die zur ausgefallenen Zone komplementäre Zone bereits wieder geteilt wurde.

Der Koordinator K geht dazu folgendermaßen vor: K bestimmt einen Punkt R_1 in der Zone $zone^c_{failed}$ zufällig und berechnet dazu einen Punkt R_2 so, dass R_1 und R_2 in verschiedenen Zonen liegen würden, falls die $zone^c_{failed}$ erneut geteilt wurde. Diese Berechnung erfolgt durch Punktspiegelung von R_1 am Mittelpunkt von $zone^c_{failed}$. Die Wahl der Punkte P_1, R_1 und R_2 wird an einem Beispiel in Abbildung 5.29 gezeigt.

K stößt einen MPX-Schlüsselaustausch mit den Punkten R_1 und R_2 an. Ziel ist zu ermitteln, ob sowohl R_1 als auch R_2 in einer einzigen Zone liegen, die also auch nur von einem Zonenverwalter verwaltet wird. Mittels des MPX-Schlüsselaustauschs werden die Schlüssel key_{K,R_1} bzw. key_{K,R_2} ausgetauscht. Aus den beiden Schlüssel berechnet der K den Schlüssel key_{K,SN_1} folgendermaßen:

$$key_{K,SN_1} = key_{K,R_1} \oplus key_{K,R_2}$$

Wurde die Zone $zone^c_{failed}$ nicht weiter geteilt, so erhält ein einzelner Zonenverwalter, eben SN_1, sowohl den Schlüssel key_{K,R_1} als auch den Schlüssel key_{K,R_2} und kann den Schlüssel key_{K,SN_1} berechnen.

Um zu überprüfen ob SN_1 den Schlüssel key_{K,SN_1} kennt, versendet K eine Challenge-Nachrichten, die eine Nonce beinhaltet:

$$1.)\mathrm{K} \overset{\mathrm{OR}}{\rightarrow} \mathrm{R}_1 : \mathbf{ZM\_CHALLENGE}[Nonce, addr_K]$$

Der Knoten SN_1 schickt die Nonce, verschlüsselt mit dem Schlüssel key_{K,SN_1} zurück:

$$2.)\mathrm{SN}_1 \overset{\mathrm{NR}}{\rightarrow} \mathrm{K} : \mathbf{ZM\_RESPONSE}[E_{key_{K,SN_1}}(Nonce), mac]$$

Nun kann sich K sicher sein, dass die Zone $zone^c_{failed}$ noch existiert und er stößt die Zonenvereinigung an:

$$3.)\mathrm{K} \overset{\mathrm{NR}}{\rightarrow} \mathrm{SN}_1 : \mathbf{ZM\_ZONE\_MERGE\_CERT}[E_{key_{K,SN_1}}(zone_K, E_{key_{MD,K}}(zone_{splitted})), mac]$$

Die `ZM_ZONE_MERGE_CERT`-Nachricht enthält einen Nachweis des Koordinators, dass SN_1 die Zone innerhalb des RMPX-Verfahrens übernommen hat. Der Nachweis $E_{key_{MD,K}}(zone_{splitted})$ ist mit dem Schlüssel $key_{MD,K}$ verschlüsselt, den das Master Device und der Koordinator K gemeinsam haben und enthält die Zonenbeschreibung der vereinigten Zone $zone_{splitted}$. Der Knoten SN_1 speichert den Nachweis, zusammen mit der Zonenbeschreibung $zone_K$ und der Vermittlungsschichtadresse von K. Die `ZM_ZONE_MERGE_CERT`-Nachricht ist zum Schutz der Vertraulichkeit mit dem Schlüssel key_{K,SN_1} verschlüsselt und beinhaltet zum Schutz der Integrität einen Message Authentication Code mac.

Um ein Scheitern der Schlüsselaustausch-Vorgänge zu erkennen verwendet der Koordinator zwei Timer `RMPX_MPX_R1_TIMER` und `RMPX_MPX_R2_TIMER`, die nach der Zeitdauer `RMPX_MPX_TIMEOUT` ablaufen, falls der Schlüsselaustausch bis dahin nicht erfolgreich war. Der Koordinator hat dann die Möglichkeit, den Schlüsselaustausch erneut vorzunehmen oder das Zone-Split-Verfahren anzustoßen. Weiterhin startet der Koordinator mit Versenden der `ZM_CHALLENGE`-Nachricht den `ZM_CHALLENGE_TIMER`, welcher nach der Zeitdauer `ZM_CHALLENGE_TIMEOUT` abläuft, falls bis dahin keine `ZM_RESPONSE`-Nachricht empfangen wurde. Auch hier kann der Koordinator die Nachricht wiederholen oder das Zone-Split-Verfahren anstoßen.

Beim Zone-Split-Verfahren wird die ausgefallene Zone aufgeteilt und von zwei Nachbarn verwaltet: dem Knoten SN_1 und dem Knoten SN_2. Dabei wird die Zone $zone_{failed}$ in die beiden Zonen $zone_{failed_1}$ und $zone_{failed_2}$ aufgeteilt. Die beiden Knoten SN_1 und SN_2 werden durch Berechnung von zwei Punkten P_1 und P_2 bestimmt. Zur Berechnung in einem SCAN der Dimension d sei $N = (n_1, n_2, ..., n_d)$ die niederwertigste Ecke und $H = (h_1, h_2, ..., h_d)$ die höchstwertige Ecke der Zone $zone_{failed}$, d. h. $\forall i, 1 \leq i \leq d : n_i \leq h_i$.

Dann berechnen sich P_1 und P_2 folgendermaßen:

$$P_1 = (n_1 - 1, n_2 + 1, ..., n_d + 1)$$

$$P_2 = (h_1 + 1, h_2 - 1, ..., h_d - 1)$$

Zur Übernahme der Zone $zone_{failed_1}$ startet der Knoten SN_1 einen MPX-Schlüsselaustausch mit dem Knoten SN_2, d. h. die `MPX`-Nachricht ist an P_2 adressiert. SN_2 geht analog vor. Dadurch prüfen SN_1 und SN_2 gegenseitig, ob sich der Punkt P_1 respektive P_2 in deren Zone befindet, d. h. SN_1 und SN_2 authentifizieren sich gegenseitig. Dadurch haben SN_1 und SN_2 zwei Schlüssel gemeinsam, die sie mittels XOR zum Schlüssel key_{SN_1,SN_2} vereinigen. Anschließend tauschen die beiden Knoten folgende Nachrichten aus:

1.)$\mathrm{SN}_1 \overset{\mathrm{OR}}{\rightarrow} \mathrm{SN}_2$:
ZS_ZONE_SPLIT_REQ$[E_{key_{SN_1,SN_2}}(zone_{SN_1}, addr_{SN_1}, E_{key_{MD,SN_1}}(zone_{failed_2}))]$

2.)$\mathrm{SN}_2 \overset{\mathrm{NR}}{\rightarrow} \mathrm{SN}_1$:
ZS_ZONE_SPLIT_REP$[E_{key_{SN_1,SN_2}}(zone_{SN_2}, E_{key_{MD,SN_2}}(zone_{failed_1})), mac]$

Die beiden Nachrichten beinhalten jeweils einen Nachweis über die Zone, die der entsprechende Knoten übernimmt. Die Nachweise sind genauso aufgebaut wie oben. Die Nachweise dienen dazu, später dem Master Device zu zeigen, dass die entsprechende Zone rechtmäßig übernommen wurde. Die `ZS_ZONE_SPLIT_REQ`- und die `ZS_ZONE_SPLIT_REP`-Nachricht sind durch Verschlüsselung und einen Message Authentication Code *mac* geschützt.

Auch die Knoten SN_1 und SN_2 verwenden Timer, um den Verlust von Nachrichten zu erkennen: Für den Schlüsselaustausch wird der Timer `RMPX_MPX_TIMER` verwendet, der nach `RMPX_MPX_TIMEOUT` abläuft, falls der Schlüsselaustausch nicht erfolgreich war. Für die `ZS_ZONE_SPLIT_REQ`-Nachricht wird analog der Timer `ZS_ZONE_SPLIT_REQ_TIMER` eingesetzt. Laufen die Timer ab, so werden die Nachrichten erneut versendet.

Egal ob das Zone-Merge-Verfahren erfolgreich war, oder ob das Zone-Split-Verfahren zur Anwendung kam: auf jeden Fall müssen nach Abschluss des jeweiligen Verfahrens die *Nachbarbeziehungen, insbesondere die Nachbarschlüssel, wiederhergestellt werden.*

SN_1 bzw. SN_2 ermitteln dazu auf Basis der Größe der übernommen Zone, mit welche Nachbarn Nachbarschlüssel ausgetauscht werden müssen. Dazu nehmen SN_1 und SN_2 mit den jeweiligen Nachbarn einen MPX-Schlüsselaustausch vor. Der jeweilige Nachbar vollzieht einen MPX-Schlüsselaustausch mit den (allgemein bekannten)

Punkten P_1 und P_2 um sicherzugehen, dass auch nur legitimierte Knoten in die Nachbarliste integriert werden. Der neue Nachbarschlüssel berechnet sich dann aus den beiden Schlüsseln durch Verknüpfung mit XOR. Insbesondere nehmen die Nachbarn nur Änderungen an ihren Nachbarlisten vor, falls sie selbst den Knotenausfall erkannt haben und den entsprechenden Übernahme-Vorgang nachvollziehen können. Eigenschaft E8 (Keine Manipulation der eigenen Zone) gilt also nach RMPX, da die Nachbarn die Übernommene Zone nur akzeptieren, falls die Zonenübernahme korrekt abgelaufen ist.

Da durch die MPX-Schlüsselaustausch-Vorgänge neue Nachbarschlüssel etabliert wurden, gilt nach RMPX wieder Eigenschaft E3 (Integrität und Vertraulichkeit mit dem Master Device). Eigenschaft E9 (Zonenverifikation) ist durch die Nachweise der Zonenübernahme erfüllt. Im Folgenden wird eine modifizierte Join-Operation vorgestellt, die diese Nachweise berücksichtigt.

5.8.1.3 Modifikation der Join-Operation

Während der Join-Operation verifiziert das Master Device anhand eines Schlüssels, in den die Zonenbeschreibung und die Vermittlungsschichtadresse eines Zonenverwalters eingeflossen sind, ob dieser Zonenverwalter auch wirklich die angegebene Zone verwaltet. Wurde ein Zonenausfall mittels RMPX repariert, so verfügen die Knoten SN_1 und SN_2 nicht über einen solchen Schlüssel. Im Fall der expliziten Zonenübergabe ist zwar der Schlüssel nach wie vor vorhanden, allerdings hat der neue Zonenverwalter eine andere Vermittlungsschichtadresse.

Um mit solchen Fällen umzugehen, wurde die `JOIN_REQ`-Nachricht so modifiziert, dass ein Knoten anzeigen kann, dass er die Zone zwar verwaltet, aber nicht über den notwendigen gemeinsamen Schlüssel mit dem Master Device verfügt:

$$\boxed{3.)\text{ZO} \overset{\text{NR}}{\rightarrow} \text{MD} : \textbf{JOIN\_REP}[''\text{NO\_KEY}'']}$$

Das Master Device stößt daraufhin einen MPX-Schlüsselaustausch mit dem Join Point an und tauscht den Schlüssel $key_{MD,ZO}$ mit dem Zonenverwalter ZO aus. Danach versendet das Master Device folgende Nachrichten (die Nummerierung bezieht sich auf die Protokollschritte aus Abschnitt 5.4):

$$\boxed{3a.)\text{MD} \overset{\text{NR}}{\rightarrow} \text{ZO} : \textbf{CERT\_REQ}[\text{Nonce}]}$$

$$\boxed{3b.)\text{ZO} \overset{\text{NR}}{\rightarrow} \text{MD} : \textbf{CERT\_REP}[\text{E}_{\text{key}_{\text{MD,ZO}}}(\text{Nonce}, \text{cert}), \text{mac}]}$$

Mit der `CERT_REQ`-Nachricht fordert das Master Device den Nachweis an, der während RMPX bzw. während der expliziten Zonenübergabe an den ZO gegeben wurde. Der Zonenverwalter sendet diesen Nachweis *cert* in der `CERT_REP`-Nachricht zurück,

welche durch Verschlüsselung und den Message Authentication Code *mac* geschützt ist. Eine Nonce dient in der `CERT_REQ`- und `CERT_REP`-Nachricht zum Schutz gegen Angriffe durch Wiedereinspielen.

Danach wird die Join-Operation weiter ausgeführt, allerdings wird im weiteren Verlauf der Join-Operation die gesamte Zone, und nicht nur eine Zonenhälfte, an den neu hinzukommenden Knoten übergeben. So wird eine Entlastung der Knoten erreicht, die eine Zone im Rahmen eines Reparatur-Verfahrens übernommen haben.

5.8.1.4 Wiederherstellung von Dienstbeschreibungen in deiner ausgefallenen Zone

Bei der expliziten Zonenübergabe können Dienstbeschreibungen, die in der übergebenen Zone gespeichert sind, an den neuen Zonenverwalter übergeben werden, so dass hier keine Dienstbeschreibungen verloren gehen. Bei Verwendung des RMPX-Verfahrens sind jedoch die in der ausgefallenen Zone gespeicherten Dienstinformationen verloren. In SCAN wird vor allem Soft State und Replikation von Dienstbeschreibungen eingesetzt, um Robustheit gegenüber dem Wegfall von gültigen Dienstbeschreibungen zu erreichen. Mit Soft State werden nach [52] Zustände bezeichnet, die, nach ihrer Einrichtung, nach einer gewissen Zeitdauer nicht mehr gültig sind und entfernt werden, es sei denn, sie werden, vor Ablauf der Zeitdauer, wieder aufgefrischt. Der Begriff Soft State wurde geprägt von [25].

Dienstbeschreibungen werden in SCAN als Soft-State gespeichert, d. h. wenn ein Dienstanbieter eine Dienstbeschreibung mittels Insert-Operation im SCAN hinterlegt, ist diese Dienstbeschreibung nur eine gewisse Zeitdauer gültig. Zonenverwalter notieren dazu, wie lange eine Dienstbeschreibung gültig ist und entfernen ungültige Dienstbeschreibungen regelmäßig aus ihrem Speicher. Dienstanbieter müssen deshalb Dienstbeschreibungen für von ihnen angebotene Dienste regelmäßig auffrischen. Dazu dient die `UPDATE`-Nachricht, die in Abschnitt 5.8.2 beschrieben wird.

Auch die Verwendung von Replikaten von Dienstbeschreibungen führt dazu, dass ein Zonenausfall nicht automatisch bedingt, dass die Dienste, deren Dienstbeschreibungen in der ausgefallenen Zone gespeichert sind, nicht mehr aufgefunden werden können. Da die Replikate der Dienstbeschreibung über das ganze SCAN verteilt sind, sind Dienstbeschreibungen aus der ausgefallenen Zone mit hoher Wahrscheinlichkeit auch noch in anderen Zonen vorhanden und können von dort bezogen werden.

5.8.2 Behandlung ausgefallener Dienste

Fällt ein Sensorknoten aus, der Dienste anbietet, so stehen diese Dienste nicht mehr zur Verfügung. In SCAN wird jedoch weiterhin die zugehörige Dienstbeschreibung gespeichert, da ein Zonenverwalter den Ausfall eines Dienstes nicht erkennt. Diese so genannten verwaisten Dienstbeschreibungen stellen ein Problem dar:

- Verwaiste Dienstbeschreibungen belegen Speicherplatz auf den entsprechenden speichernden Knoten.

- Verwaiste Dienstbeschreibungen erhöhen den Kommunikationsaufwand bei der Lookup-Operation
- Verwaiste Dienstbeschreibungen resultieren darüber hinaus in mehreren Anfragen, die ein Dienstnutzer benötigt, um einen aktiven Dienst aufzurufen.

Aus diesen Gründen ist es wünschenswert, dass Dienstbeschreibungen eines ausgefallenen Dienstes aus SCAN entfernt werden. Zwei Verfahren hierzu werden im Folgenden beschrieben:

5.8.2.1 Explizite Abmeldung

Falls die einzelnen Sensorknoten über ein Energie-Management wie z. B. in [33] verfügen, d. h. jederzeit darüber informiert sind, wie viel Energie ihnen noch zur Verfügung steht und wie viel Energie die Ausführung eines Dienstes benötigt, so können die Dienstanbieter einen von ihnen angebotenen Dienst rechtzeitig abmelden.

Um eine explizite Abmeldung zu ermöglichen, muss der Insert mittels SPX oder MPX vorgenommen worden sein, d. h. es muss ein gemeinsamer Schlüssel $key_{A,ZO}$ zwischen dem Dienstanbieter A und dem Zonenverwalter ZO bestehen. Weiterhin muss A die Vermittlungsschichtadresse des Zonenverwalters gespeichert haben. A versendet dann folgende Nachricht:

$$\mathrm{A} \overset{\mathrm{NR}}{\rightarrow} \mathrm{ZO} : \textbf{SIGNOFF}[\mathrm{E}_{\mathrm{key_{A,ZO}}}(\mathrm{h}_1, \mathrm{addr_A}), \mathrm{mac}]$$

Die Nachricht beinhaltet den Hash-Wert h_1 des Dienstnamens, der es dem ZO zusammen mit der Vermittlungsschichtadresse von A ($addr_A$) ermöglicht, die richtige Dienstbeschreibung zu löschen. Integrität der Nachricht wird über den Message Authentication Code *mac* erreicht. Vertraulichkeit wird durch Verschlüsselung realisiert. Soll Explizite Abmeldung verwendet werden, so ist zu beachten, dass ein Zonenverwalter nach einem Insert mittels SPX oder MPX die jeweiligen ausgetauschten Schlüssel speichern muss und diese auch bei der Teilung einer Zone entsprechend weitergibt.

5.8.2.2 Soft-State

Um auch mit unvorhergesehenen Knotenausfällen umgehen zu können werden die Dienstbeschreibungen wie oben beschrieben als Soft-State gespeichert. Ein Dienstanbieter kann die Lebenszeit einer Dienstbeschreibung durch eine `UPDATE`-Nachricht verlängern, oder er kann eine Dienstbeschreibung nach Ablauf der Gültigkeit durch die Insert-Operation erneut einfügen.

Für das Versenden der `UPDATE`-Nachricht gibt es zwei Möglichkeiten:

- Versenden der Nachricht über das Overlay: Beim Versenden über das Overlay berechnet der Dienstanbieter A wie oben den Hash-Wert h_1 des Dienstnamens

und sendet eine Nachricht an diesen Hash-Wert:

$$\boxed{\mathrm{A} \overset{\mathrm{OR}}{\rightarrow} \mathrm{h}_1 : \textbf{UPDATE}[\mathrm{addr}_\mathrm{A}]}$$

Die Vermittlungsschichtadresse $addr_A$ des Dienstanbieters A ermöglicht es dem entsprechenden Zonenverwalter ZO zusammen mit dem Hash-Wert des Dienstnamens, die richtige Dienstbeschreibung aufzufrischen. Integrität und Vertraulichkeit werden durch das Overlay-Routing geschützt. Ein Outsider-Angreifer kann also Integrität und Vertraulichkeit der Nachricht nicht verletzen, allerdings ist es jedem Insider-Angreifer möglich, die Update-Nachricht zu fälschen.

- Versenden über die Vermittlungsschicht: Weniger Kommunikationsaufwand erfordert das Versenden der Nachricht über die Vermittlungsschicht. Dieses ist allerdings nur möglich, wenn die entsprechende Dienstbeschreibung per Insert mit SPX oder MPX eingespeichert wurde, da nur in diesem Fall dem Dienstanbieter die Netzwerk-Adresse des entsprechenden Zonenverwalters ($addr_{ZO}$) sowie ein gemeinsamer Schlüssel key bekannt sind. Es werden folgende Nachricht versendet:

$$\boxed{\mathrm{A} \overset{\mathrm{NR}}{\rightarrow} \mathrm{addr_{ZO}} : \textbf{UPDATE}[\mathrm{E_{key}}(\mathrm{addr_A}, \mathrm{h_1}), \mathrm{mac}]}$$

Die Nachricht beinhaltet die Adresse des Dienstanbieters ($addr_A$) sowie den Hash-Wert des Dienstnamens h_1. Die beiden Werte zusammen ermöglichen es dem Zonenverwalter, die richtige Dienstbeschreibung aufzufrischen. Die Vertraulichkeit der Nachricht wird durch Verschlüsselung mit dem Schlüssel key erreicht. Die Integrität der Nachricht wird durch den Message Authentication Code mac geschützt.

5.9 Sichere Dienstbeschreibungen

Die bisher beschriebenen Protokolle schützten Integrität und Vertraulichkeit der Kommunikation zwischen dem Dienstanbieter und dem Zonenverwalter bzw. zwischen dem Zonenverwalter und dem Dienstnutzer. Implizit werden damit auch die übertragenen Dienstbeschreibungen geschützt. Allerdings gibt es keinen Schutz der (abgesetzten) Kommunikation zwischen einem Dienstanbieter und einem Dienstnutzer, da diese Kommunikation zeitversetzt über die Speicherung beim Zonenverwalter erfolgt. Dies bedeutet zum Beispiel, dass ein Zonenverwalter von ihm gespeicherte Dienstbeschreibungen beliebig manipulieren kann, ohne dass dies einem Dienstnutzer auffällt. Im Folgenden wird ein Verfahren beschrieben, wie die Integrität und Vertraulichkeit von Dienstbeschreibungen auch zwischen Dienstanbieter und Dienstnutzer, über die Speicherung im SCAN hinweg, geschützt werden können.

Die Integrität von Dienstbeschreibungen wird bereits zu einem gewissen Grad durch die Replikation von Dienstbeschreibungen und den Mehrheitsentscheid geschützt,

allerdings ist das Verfahren aufwendig und ermöglicht keinen Schutz der Vertraulichkeit.

Um Integrität und Vertraulichkeit von Dienstbeschreibungen während der Speicherung zu erreichen wird deshalb im Folgenden ein gemeinsamer Schlüssel zwischen einem Dienstanbieter, der die Dienstbeschreibung im Diensteverzeichnis hinterlegt, sowie einem Dienstnutzer, der die Dienstbeschreibung abfragt, eingerichtet. Mittels diesem gemeinsamen Schlüssel kann mittels eines Message Authentication Codes in der Dienstbeschreibung die Integrität sowie mittels Verschlüsselung der Dienstbeschreibung die Vertraulichkeit der Dienstbeschreibung sichergestellt werden. Mittels dieses Verfahrens wird Eigenschaft E12 (Schutz von Dienstbeschreibungen während der Speicherung) von SCAN erreicht. Zur Konstruktion des Schlüssels zwischen dem Dienstanbieter und dem Dienstnutzer werden Informationen benötigt, die dem Zonenverwalter, der die Dienstbeschreibung speichert, nicht bekannt sind. Dies können z. B. sein:

- Der Dienstname, oder
- Der Kontext, in dem sich ein Dienst befindet

Im Folgenden wird beschrieben, wie jede diese beiden Informationen dazu verwendet werden kann, um den gemeinsamen Schlüssel zwischen einem Dienstanbieter und einem Dienstnutzer herzuleiten.

5.9.1 Dienstname als gemeinsames Geheimnis

Der Dienstname eines Dienstes wird im SCAN nie im Klartext versendet, sondern es wird lediglich der Hash-Wert des Dienstnamens verwendet. Da eine Hash-Funktion nur sehr schwer umkehrbar ist, und der Name eines Dienstes dem Dienstanbieter und den Dienstnutzern bekannt ist, kann aus dem Dienstnamen ein Schlüssel hergeleitet werden, mit dem Integrität und Vertraulichkeit der Dienstbeschreibung während der Speicherung geschützt werden können. Dazu wird an einen beliebigen, aber allgemein bekannten, String s der Dienstname angehängt (im Folgenden dargestellt durch $|$) und der Schlüssel key mittels einer Hash-Funktion hergeleitet:

$$key = hash(s|Dienstname)$$

Wiederum wegen der Eigenschaften von kryptographischen Hash-Funktionen kann vom Hash-Wert $hash(Dienstname)$, der dem die Dienstbeschreibung speichernden Knoten bekannt ist, nicht auf $hash(s|Dienstname)$ geschlossen werden.

Der Schlüssel key wird dazu verwendet, die Dienstbeschreibung zu verschlüsseln und einen Message Authentication Code über die Dienstbeschreibung zu bilden. Damit wird ein Schutz von Integrität und Vertraulichkeit der Dienstbeschreibungen, und somit Eigenschaft E12 (Schutz von Dienstbeschreibungen während der Speicherung), erreicht.

5.9.2 Kontext als gemeinsames Geheimnis

Eine weitere Möglichkeit um ein gemeinsames Geheimnis einzurichten besteht darin, Kontextinformationen zur Herleitung des gemeinsamen Geheimnisses zu verwenden. Vielfältige Möglichkeiten stehen hierzu zur Verfügung:

- Ein Kontextgeber könnte über einen Location Limited Channel Kontext-IDs verteilten. Beispielsweise könnte in einem Raum ein Infrarotsender vorhanden sein, der regelmäßig eine lange Zahl aussendet, die als Kontext-ID verwendet wird. In [93] wurde ein solches System implementiert und die generelle Einsetzbarkeit gezeigt.
- Ein Display könnte eine Zahl als Barcode ausgeben, die von einem Sensor, der sich in diesem Kontext befindet, visuell erfasst und als Kontext verwendet wird.

Die Kontext-ID ist nur innerhalb eines gewissen Bereichs gültig, in dem sich der Dienstanbieter und Dienstnutzer, nicht aber der Zonenverwalter befinden sollten. Das hier beschriebene Verfahren eignet sich also vor allem für Sensornetze, in denen Dienste nur in einer räumlichen Umgebung aufgerufen werden.

Dieser gemeinsame Kontext wird als gemeinsames Geheimnis verwendet und Integrität und Vertraulichkeit der Dienstbeschreibungen werden damit gesichert wie oben beschrieben. Weiterhin ist es möglich, eine Dienstbeschreibung nicht unter $hash(Dienstname)$ abzuspeichern sondern unter $hash(Dienstname|Kontext)$. Damit sind die entsprechenden Dienstbeschreibungen nur noch für Dienstnutzer im gleichen Kontext auffindbar.

5.10 Zusammenfassung

In diesem Kapitel wurde der Entwurf von Secure Content Addressable Networks (SCAN) vorgestellt. SCAN ermöglicht es einem Dienstanbieter, Dienstbeschreibungen zu einem Dienst mittels der Insert-Operation unter dem Namen des Dienstes zu veröffentlichen. Die Dienstbeschreibungen werden in einer verteilten Hash-Tabelle dezentral gespeichert. Dienstnutzer können über die Lookup-Operation eine Liste von Dienstbeschreibungen erhalten, die in SCAN unter einem gewissen Dienstnamen hinterlegt wurden. Sowohl während der Lookup- als auch während der Insert-Operation werden die Integrität und Vertraulichkeiten der Dienstbeschreibungen gesichert. Auch während der Speicherung sind Integrität und Vertraulichkeit von Dienstbeschreibungen sichergestellt.

SCAN besteht aus einer Menge von Knoten, die während dem Beitritt zu SCAN mittels der Join-Operation authentifiziert und zum Beitritt autorisiert werden. Während der Join-Operation werden außerdem Nachbarschlüssel zwischen im SCAN-Raum benachbarten Knoten eingerichtet. Basierend auf diesen Nachbarschlüsseln wurden zwei Schlüsselaustauschprotokolle, Single-Path-Key-Exchange (SPX) und

Multi-Path-Key-Exchange (MPX), entwickelt, die einen Schlüssel zwischen zwei beliebigen Knoten des SCAN einrichten, mit dem Integrität und Vertraulichkeit beliebiger Nachrichten geschützt werden können. MPX bietet, neben der Möglichkeit zum Schlüsselaustausch, auch noch die Möglichkeit, zu verifizieren, welche Zone ein Knoten verwaltet. Von dieser Eigenschaft wird insbesondere während der Insert- und der Lookup-Operation Gebrauch gemacht.

Um SCAN robust gegen den Ausfall von Sensorknoten zu machen, wurden die Reparaturverfahren explizite Zonenübergabe und RMPX entwickelt, um Nachfolger für ausgefallene Zonenverwalter zu bestimmen. Darüber hinaus werden Dienstbeschreibungen in SCAN als Soft-State gespeichert, so dass Dienstbeschreibungen von ausgefallenen Diensten nach einer gewissen Zeitspanne aus SCAN verschwinden und Dienstbeschreibungen, die aufgrund eines Zonenausfalls verloren gegangen sind, wieder in SCAN erscheinen.

SCAN erfüllt damit alle Anforderungen aus Abschnitt 5.1. Insbesondere verhindert SCAN die in Abschnitt 3.2 beschriebenen Angriffe: den Sybil-Angriff, den Churn-Angriff, Manipulation der SCAN-Routing-Tabellen, den Eclipse-Angriff, den Man-in-the-Middle Angriff (bzw. feindliches Overlay) sowie Angriffe auf Dienstbeschreibungen während der Übertragung bzw. Speicherung.

6. Implementierung und Leistungsanalyse

Um das sichere Diensteverzeichnis für Sensornetze, Secure Content Addressable Networks (SCAN), zu evaluieren, wurde es zur Leistungsanalyse im Netzwerksimulator GloMoSim [9, 116] sowie als Prototyp auf MICAz Sensorknoten [104] implementiert. Der Prototyp diente als Demonstrator und wird in [47] beschrieben. Im Folgenden wird zuerst die Implementierung im Simulator GloMoSim besprochen und anschließend der Prototyp vorgestellt.

6.1 Implementierung und Leistungsanalyse in GloMoSim

GloMoSim ist ein diskreter, ereignisbasierter Simulator, der speziell für drahtlose Netze entwickelt wurde. In einem ereignisbasierten Simulator erfolgen Zustandsänderungen nur auf Basis von Ereignissen, wie beispielsweise dem Empfang einer Nachricht. GloMoSim verwendet die Simulationssprache Parsec [8], welche an die Sprache C angelehnt ist und sowohl sequentielle als auch parallele Simulationen unterstützt. Abbildung 6.1 zeigt den modularen Aufbau des Simulators GloMoSim, der an das ISO/OSI-Basisreferenzmodell [120] anlehnt, aber auch Module enthält, die so nicht im ISO/OSI-Basisreferenzmodell vorgesehen sind (z. B. Cluster-Bildung). Durch den modularen Aufbau ermöglicht GloMoSim eine einfache Erweiterung des Simulators und eine Austauschbarkeit der Protokolle.

6.1.1 Simulationsmodell

In diesem Abschnitt wird das verwendete Simulationsmodell beschriebene, welches maßgeblich durch die während der Demonstrator-Implementierung gewonnenen Erkenntnisse über SCAN beeinflusst ist.

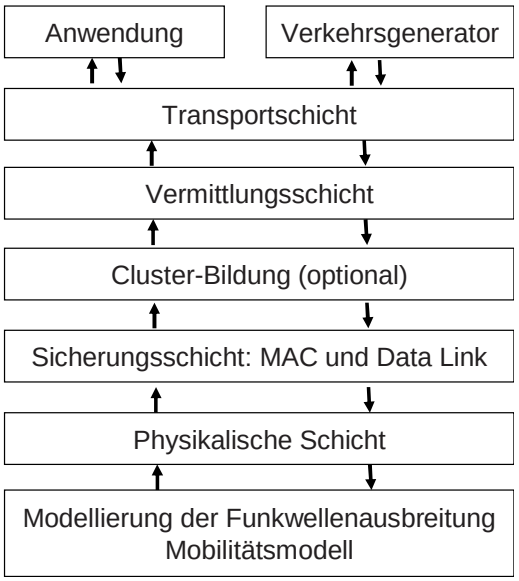


Abbildung 6.1: Schichten-Aufbau des Simulators GloMoSim

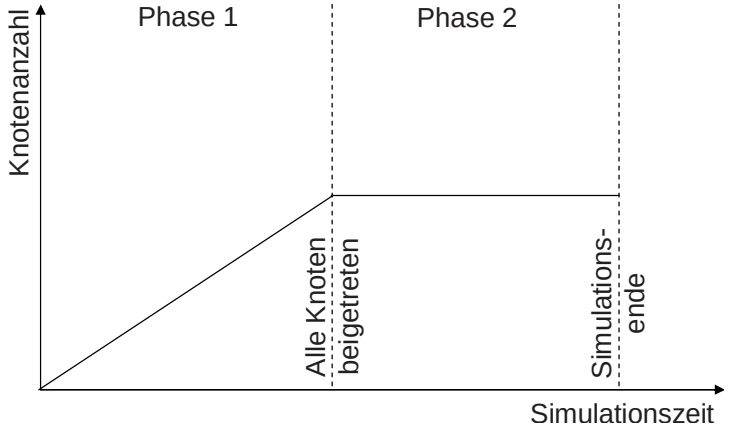


Abbildung 6.2: Überblick über die Vorgehensweise der Simulation

6.1.1.1 Methodisches Vorgehen

Eine wesentliche Vorüberlegung zur Simulation war, Simulationsläufe mit verschiedener Gesamtknotenanzahl N vergleichbar zu machen, um die Leistungsfähigkeit von SCAN bei steigender Gesamtknotenanzahl untersuchen zu können. Weiterhin sollten nicht nur SCANs untersucht werden, in die bereits alle Knoten integriert sind, sondern es sollte auch der Aufbau der SCANs mit berücksichtigt werden. Dazu wurde die Simulation in zwei Phasen gegliedert wie in Abbildung 6.2 zu sehen: In der ersten Phase werden sukzessive alle Knoten über die Join-Operation zum SCAN hinzugefügt bis alle Knoten Mitglied im SCAN sind, und damit eine Zone verwalten. In dieser Phase wird das SCAN also nach und nach aufgebaut. In Phase 2 werden dann keine Knoten mehr zum SCAN hinzugefügt. Diese Phase läuft bis zum Simu-

lationsende. Beide Phasen sollen in die Ergebnisse mit gleichem Gewicht eingehen, weswegen die Zeitdauer beider Phasen gleich lang gewählt wurde. Eine Vergleichbarkeit der Ergebnisse für verschiedenen Knotenanzahlen wird nun dadurch erreicht, dass die Gesamtsimulationszeit an die Gesamtknotenanzahl angepasst wird. Da im Simulationsmodell jede 30 Sekunden ein Knoten hinzugefügt wird, und die zweite Phase so lange wie die erste Phase dauern soll, wird als Gesamtsimulationszeit t_{Sim} folgender Wert gewählt:

$$t_{Sim} = 2 * 30s * N$$

Dieses Vorgehen bei der Simulation entspricht der Realität, in der Sensornetze oft sukzessive aufgebaut werden. Während die theoretischen Berechnungen der vorherigen Kapitel sich immer auf ein bereits vollständig aufgebautes SCAN bezogen, wird in der Simulation auch der Aufbau des SCANs an sich betrachtet.

Um ein möglichst realistisches Simulationsmodell zu erhalten, werden sowohl in der ersten Phase als auch in der zweiten Phase von den Knoten Lookup- und Insert-Operationen ausgeführt und dadurch Verkehr erzeugt. Jeder Knoten führt dabei, nach seiner eigenen Join-Operation, seine Lookup- und Insert-Operationen durch und wiederholt diese nach einer gewissen Zeitspanne (`LOOKUP_AGAIN_INTERVAL` respektive `INSERT_REFRESH_INTERVAL`). Lookup- und Insert-Operationen finden also sowohl in Phase 1 als auch in Phase 2 statt, in Phase 1 parallel zu den Join-Operationen anderer Knoten. Die Join-Operation wird allerdings nur in Phase 1 ausgeführt. In Abbildung 6.3 wird gezeigt, in welcher Reihenfolge die Knoten die Join-, Lookup- und Insert-Operation ausführen und wann die Lookup-Operationen bzw. Insert-Operationen wiederholt werden.

Neben der Reihenfolge der Operationsausführung musste auch festgelegt werden, wie oft die Lookup- bzw. Join-Operation durchgeführt wird, was maßgeblich dadurch bestimmt wird, wie viele Dienste ein Knoten anbietet und wie viele Dienste er nutzt. In der Simulation bieten Sensoren zufällig gleichverteilt zwischen einem und 7 Diensten an und nutzen zufällig und gleichverteilt zwischen einem und 10 Diensten. Diese Wahl ist durch eine in [47] beschriebene verteilte dienstorientierte Sensornetzanwendung motiviert, die zur Dienstesuche SCAN einsetzt. Die Wahl der angebotenen und benutzten Dienste bedeutet insbesondere, dass Knoten sowohl als Dienstanbieter als auch als Dienstnutzer auftreten. Weiterhin wurde festgelegt, dass 100 verschiedene Dienste im Netz existieren, d. h. für jeden Knoten werden die entsprechenden Dienste aus dieser Menge von 100 Diensten gewählt. Auch dies ist motiviert aus den praktischen Erfahren mit der in [47] beschriebenen Sensornetzanwendung.

Schließlich wurde für die Simulation ein Simulationsgebiet von 500 m auf 500 m gewählt, in dem die Knoten zufällig und gleichverteilt platziert werden.

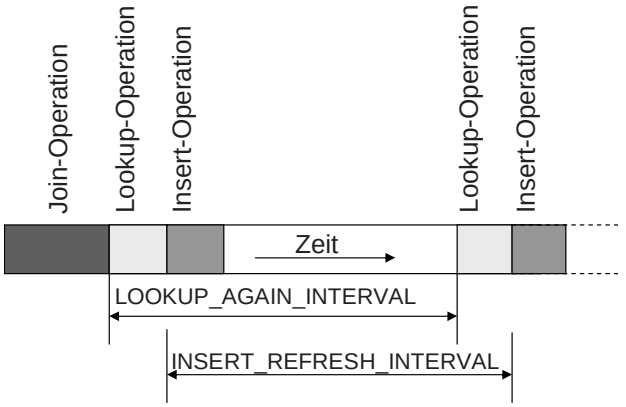


Abbildung 6.3: Reihenfolge der Ausführung der Join-, Lookup- und Insert-Operation auf einem Sensorknoten

6.1.1.2 Angreifermodellierung

Das in Abschnitt 3.1 beschrieben Angreifermodell wird in der Simulation folgendermaßen umgesetzt:

Zu Beginn der Simulation wird eine Menge von Knoten zufällig ausgewählt, deren Knoten im Laufe der Simulation zu Angreifern werden. Die Menge beinhaltet dabei den Anteil m aller Knoten. Hierdurch wird eine Vergleichbarkeit der Simulationen mit verschiedenen Gesamtknotenanzahl erreicht, da keine absolute Anzahl an Angreifern sondern ein Prozentsatz an Angreifern von der Gesamtknotenanzahl festgelegt wird. Für jeden dieser Knoten wird zufällig gleichverteilt innerhalb des Intervalls [Beitrittszeitpunkt zu SCAN,Simulationsende] der Zeitpunkt bestimmt, zu dem dieser Knoten zum Angreifer wird. Insbesondere ist es damit auch möglich, dass ein Knoten noch während Phase 1 der Simulation zum Angreifer wird und somit einen größeren Einfluss auf das SCAN hat, als in den theoretischen Berechnungen der letzten Kapitel, da der Knoten über eine größere Zone im SCAN verfügt, falls er früher korrumpiert wird.

In den Simulationen wird ein zusammenarbeitender Insider-Angreifer simuliert, da dies für die vorliegende Arbeit der schlimmste mögliche Angreifer ist. Ein Angreifer legt byzantinisches Verhalten an den Tag, d. h. er verhält sich beliebig und kann sich insbesondere auch protokollkonform verhalten. Alle Insider-Angreifer der Simulation markieren Overlay-Nachrichten, die sie im Overlay weiterleiten. Bei der Auswertung werden die Nachrichten dann so behandelt, dass sie den maximalen Schaden (Worst Case) anrichten würden. Wie markierte Nachrichten im einzelnen Fall für die Auswertung behandelt werden, wird beim entsprechenden Experiment angegeben. Auch hier soll darauf hingewiesen werden, dass mit diesem Vorgehen eine Abweichung zu den theoretischen Werte aus Kapitel 5 zu erwarten ist: während bei der theoretischen Berechnung davon ausgegangen wird, dass alle Angreifer zu einem gewissen Zeitpunkt bösartig werden, geschieht dies in der Simulation sukzessive. Damit ist

es unter anderem möglich, dass ein Angreifer einen großen Einfluss auf SCAN hat, wenn er zu einem sehr frühen Zeitpunkt dem SCAN beitritt.

6.1.1.3 Knotenausfall

Um die Reparaturverfahren zur Behandlung von Knotenausfällen zu testen, werden Knotenausfälle simuliert, indem zu Beginn eine Menge von Knoten festgelegt wird, deren Knoten im Laufe der Simulation ausfallen. Die Menge beinhaltet einen Anteil f aller Knoten, wobei auch Angreifer ausfallen können. Der Ausfallzeitpunkt wird für diese Knoten zufällig gleichverteilt im Intervall [Beitrittszeitpunkt zu SCAN,Simulationsende] bestimmt. Fällt ein Knoten aus, so beantwortet er keine SCAN-Nachrichten mehr. Die Simulation von Knotenausfällen kommt allerdings nur bei der Untersuchung der Protokolle zum Umgang mit Knotenausfällen zum Einsatz. In allen anderen Fällen werden lediglich Angreifer simuliert.

6.1.1.4 Parameterisierung der Simulation

GloMoSim wurde für die Simulation mit den Eigenschaften der MICAz Sensorknoten parametrisiert, die dem Datenblatt [104] entnommen wurden. Diese Sensorknoten wurden auch für die Prototyp-Implementierung von SCAN verwendet. Nach Datenblatt beträgt die Datenrate 250 kBit/s und die Übertragung erfolgt im Frequenzband um 2,4 GHz. Sende- und Empfangsleistung aus dem Datenblatt führen in GloMoSim zu einer maximalen Übertragungsreichweite von 150m. Als Vermittlungsschicht-Protokoll kommt PRECALC [12] zum Einsatz. PRECALC berechnet zu Beginn einer Simulation, basierend auf der Position der einzelnen Knoten im Simulationsgebiet sowie der maximalen Sendereichweite, kürzeste Pfade zwischen Knotenpaaren. Diese werden in eine statische Routing-Tabellen eingetragen, die im Weiteren für Routing-Entscheidungen verwendet wird. Für den Transport der Daten werden IP-Pakete eingesetzt. Die Wahl von PRECALC für die Simulation beruht darauf, dass sich bisher noch kein Routing-Protokoll für Sensornetze etabliert hat. Zudem hat [12] gezeigt, dass sich herkömmliche Routing-Protokolle wie AODV [72] oder OLSR [51] nicht für drahtlose Sensornetze mit 1000 Knoten und mehr eignen. Um die Ergebnisse der Simulation nicht zu verfälschen, wird das Protokoll PRECALC verwendet. Zum Medienzugriff wird die in GloMoSim vorhandene Implementierung von IEEE 802.11 CSMA/CA verwendet. IEEE 802.11 CSMA/CA ist zu dem von MICAz Sensorknoten eingesetzten Medienzugriffsverfahren ähnlich. Als Funkwellenausbreitungsmodell wird das Two-Ray-Modell verwendet.

6.1.1.5 Parameter kryptographischer Verfahren

Der Entwurf von SCAN legt keine konkret zu verwendeten kryptographischen Algorithmen fest, sondern setzt ganz allgemein eine symmetrische Chiffre, eine Hash-Funktion und einen Algorithmus zur Berechnung von Message Authentication Codes voraus. Für die Simulation sind jedoch die konkreten Parameter solcher Algorithmen, z. B. die Schlüssellänge, von Interesse, da sie unter anderem die Länge der zu übertragenden Nachrichten beeinflussen. In der Simulation werden deshalb zwei Parametermengen betrachtet:

- *Full Security (FS):* In dieser Parametermenge kommt der Verschlüsselungsalgorithmus AES [30] oder ein vergleichbarer Algorithmus zum Einsatz. Die Schlüssellänge beträgt dabei $128Bit$. Weiterhin kommt die Hash-Funktion SHA-1 [71] oder ein vergleichbarer Algorithmus zum Einsatz, die Hash-Werte von $128Bit$ Länge erzeugen. Als Message Authentication Code wird HMAC [56] verwendet, was in einem $128Bit$ langen Message Authentication Code resultiert.

- *Tiny Security (TS):* Diese Parametermenge ist an TinySec [54] angelehnt. TinySec ist ein Security-Framework für Sensornetze, das unter anderem Verschlüsselung und Integritätsschutz bietet. In TinySec wird zur Verschlüsselung der Algorithmus Skipjack [99] eingesetzt, der einen Schlüssel der Länge $80Bit$ verwendet. Für den Message Authentication Code kommt eine Konstruktion ähnlich zu CBC-MAC zum Einsatz, die $32Bit$ lang ist. In TinySec ersetzt diese jedoch den im Nachrichtenformat von TinyOS benötigten CRC-Wert, so dass hier eigentlich kein zusätzlicher Overhead entsteht. Für die Simulation in GloMoSim wird allerdings trotzdem ein MAC von $32Bit$ angenommen, da das TinyOS-Nachrichtenformat nicht simuliert wird.

Tabelle 6.1 fasst diese beiden Parametermengen noch einmal zusammen:

Name	Schlüssellänge	MAC Länge
Full Security	128 Bit	128 Bit
Tiny Security	80 Bit	32 Bit

Tabelle 6.1: Parametermengen Full Security (FS) und Tiny Security (TS)

6.1.1.6 Integration von SCAN in GloMoSim

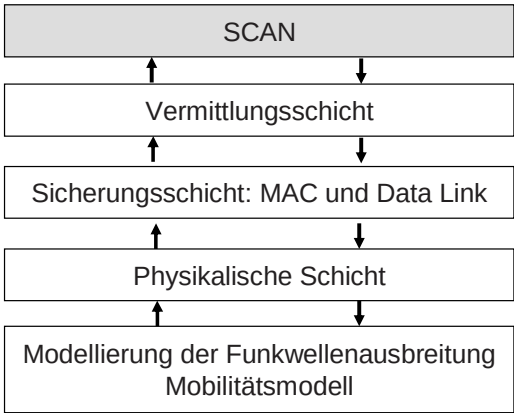


Abbildung 6.4: Integration von Secure Content Addressable Networks in GloMoSim

Abbildung 6.4 zeigt die Integration von SCAN in den GloMoSim-Protokollstapel. SCAN setzt auf der Vermittlungsschicht auf und kann so auf bereits in GloMoSim

vorhandene Vermittlungsschichtprotokolle zurückgreifen. Weitere Protokolle, z. B. Protokolle der Anwendungsschicht, kommen in der Simulation nicht zum Einsatz. Der Verkehr wird nicht durch einen gesonderten Verkehrsgenerator erzeugt, sondern durch SCAN selbst, wie oben beschrieben. Die Implementierung der Simulation verzichtet auf die in GloMoSim optionale Schicht zur Cluster-Bildung.

6.1.1.7 Verwendete Metriken

Um den *Erfolg einer Lookup-Operation* zu quantifizieren, und damit die Qualität von SCAN zu beurteilen, wird der Erfolg pro Dienstbeschreibung betrachtet, d. h. es wird ermittelt, welcher Anteil an Dienstbeschreibungen erfolgreich zurückgeliefert wird. Wir bezeichnen diesen Anteil als *Lookup-Wahrscheinlichkeit* p_{lookup}, da er die Wahrscheinlichkeit angibt, dass eine bestimmte Dienstbeschreibung während einer Lookup-Operation erfolgreich zurückgeliefert wird. Die Lookup-Wahrscheinlichkeit p_{lookup} berechnet sich aus der Gesamtanzahl $gesamt$ von eingefügten Dienstbeschreibungen sowie aus der Anzahl der aufgefundenen Dienstbeschreibungen $gefunden$ während einer Lookup-Operation folgendermaßen:

$$p_{lookup} = 100\% * \begin{cases} 1 & , falls \quad gesamt = 0 \\ \frac{gefunden}{gesamt} & , falls \quad gesamt \neq 0 \end{cases}$$

Eine Lookup-Operation ist auch dann erfolgreich, wenn keine Dienstbeschreibungen zurückgeliefert wurden, weil für den angefragten Dienst keine Dienstbeschreibungen vorhanden sind. Angreifer finden ihre Berücksichtigung in der Anzahl an gefundenen Dienstbeschreibungen $gefunden$. Die Gesamtanzahl an Dienstbeschreibungen wird in der Simulation ermittelt, indem zu Beginn der Insert-Operation die Anzahl der verfügbaren Dienstbeschreibungen des jeweiligen Dienstes um eins erhöht wird. Da die Gesamtanzahl der im Netz verfügbaren Dienstbeschreibungen in die Erfolgswahrscheinlichkeit einer Lookup-Operation einfließt, wird so erreicht, dass auch fehlgeschlagene Insert-Operationen berücksichtigt werden. Durch das beschriebene Vorgehen erfolgt allerdings eine leichte Verschlechterung der Simulationsergebnisse, falls Lookup- und Insert-Operationen in kurzer zeitlicher Abfolge ausgeführt werden: falls die entsprechenden Insert-Nachrichten den zuständigen Zonenverwalter noch nicht erreicht haben, bevor die Nachrichten der Lookup-Operation dort eintreffen, ist zwar die Gesamtzahl der Dienstbeschreibungen schon um eins erhöht, aber die entsprechende Dienstbeschreibung wird in der Lookup-Operation nicht berücksichtigt.

Da sich für Sensornetze noch keine einheitlichen Protokolle auf Schicht 2 und Schicht 3 etabliert haben, wird als Metrik für den Kommunikationsaufwand einer Operation die *Anzahl der im Overlay gesendeten Bytes* betrachtet. Dies bedeutet, dass die Längen der Overlay-Nachrichten, die im Rahmen des Overlay-Routings versendet werden, aufsummiert werden. Wird also eine Nachricht von einem Overlay-Knoten A an einen Overlay-Knoten B versendet, und befinden sich auf dem Overlay-Routing-Pfad zwischen A und B im Overlay die Zwischenknoten K_1, K_2 und K_3, so wird die die Nachrichtenlänge viermal gezählt, da vier Knoten (A, K_1, K_2 und K_3) die

Nachricht im Overlay senden. D.h. es gilt:

Anzahl im Overlay gesendete Bytes=Overlay-Hops*Nachrichtenlänge

Diese Metrik bezieht insbesondere keine Kontrollinformationen der unterliegenden Schichten mit ein, da diese stark von den eingesetzten Protokollen auf diesen Schichten abhängen. Insbesondere wird auch keine maximale und feste Nachrichtenlänge angenommen, sondern die von SCAN verschickten Nachrichten können unterschiedliche Längen haben. Zwar hat TinyOS eine voreingestellte Nachrichtenlänge von 59 Byte (29 Byte davon Nutzdaten), allerdings kann diese auch verändert werden.

Im Rahmen dieser Arbeit wird die Anzahl der im Overlay gesendeten Bytes oft auf die Gesamtanzahl der Knoten umgelegt, um zur Metrik *Anzahl im Overlay gesendeter Bytes pro Knoten* zu kommen: diese Sichtweise eignet sich gut zum Vergleich von Netzen mit unterschiedlicher Knotenanzahl: insbesondere kann bei verteilten Algorithmen mitberücksichtigt werden, dass die Aufgaben des verteilten Algorithmus in großen Netzen auf mehr Knoten verteilt ist als in kleinen Netzen.

6.1.1.8 Zusammenfassung der Parameterisierung von GloMoSim

Tabelle 6.2 gibt einen Überblick über die verwendeten Simulationsparameter.

Parameter	Wert
Anzahl Knoten	100, 250, 500, 1000, 1500
Simulationszeit	2 * 30 s * Anzahl Knoten
Größe des Simulationsgebiets	500m auf 500m
Verteilung der Knoten im Simulationsgebiet	zufällig
Mobilität	keine
Routing-Protokoll	PRECALC
Medienzugriff	802.11
Datenrate	250 kBit/s
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$

Tabelle 6.2: Parameter der Simulation

6.1.2 Untersuchung der Topologie des Overlay-Netzes

In diesem Abschnitt werden einige Voruntersuchungen zu SCAN vorgenommen, betreffend der Topologie des Overlay-Netzes.

In SCAN stellt die Dimension d des SCAN-Raums einen wichtigen Parameter dar. Bei der Wahl von d müssen folgende Aspekte berücksichtigt werden:

- Ein Sensorknoten speichert über jeden seiner Nachbarn Informationen (z. B. Schicht-3-Adresse, Nachbarschlüssel). Die Anzahl der Nachbarn bestimmt also

maßgeblich den Speicheraufwand, den SCAN auf einem Sensorknoten verursacht. Ein Sensorknoten muss mindestens Informationen über $2d$ Nachbarn speichern, da er für das Overlay-Routing in jeder Dimension mindestens zwei Nachbarn benötigt.

- Die Dimension d bestimmt den Grad der Verbundenheit der Sensorknoten im Overlay. Wird d erhöht, so steigt auch der Grad der Verbundenheit. In SCAN bedeutet dies konkret, dass ein Knoten im Overlay mehr Nachbarn hat. Damit werden auch die Overlay-Routing-Pfade kürzer, was in weniger Kommunikationsaufwand resultiert.

- Die Dimension d legt eine untere Grenze für die Anzahl verfügbaren zonendisjunkten Pfade fest: Bei einem SCAN der Dimension $2d$ hat ein Knoten mindestens $2d$ Nachbarn. Von diesen befinden sich wenigstens d Nachbarn in Richtung des Ziels und können für die Konstruktion der disjunkten Pfade verwendet werden.

In SCAN kann es jedoch vorkommen, dass ein Sensorknoten den gleichen Sensorknoten mehrmals als Nachbarn hat. Dies geschieht z. B. wenn nicht genügend Sensorknoten im SCAN vorhanden sind. Der Extremfall ist ein SCAN, das nur aus einem Knoten besteht: dieser Knoten ist in allen Dimensionen zu sich selbst benachbart. Die Anzahl verschiedener Nachbarn spielt eine maßgebliche Rolle für die Konstruktion der disjunkten Pfade: es können maximal so viele verschiedene zonendisjunkte Pfade erzeugt werden, wie verschiedene Nachbarn vorhanden sind. Um diesen Aspekt zu untersuchen, werden im folgenden Experiment SCAN-Netze mit verschiedener Gesamtknotenanzahl und verschiedener Dimension aufgebaut. Am Ende der Simulation wird ermittelt, wie viele verschiedene Nachbarn ein Knoten hat. Dabei gelten zwei Nachbarn als gleich, wenn sie dieselbe Vermittlungsschichtadresse haben. Tabelle 6.3 zeigt die Parametrisierung der Simulation für das Experiment.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 750, 1000, 1500$
Dimension d	$2, 3, 4, 5, 6, 7, 8, 9, 10$
Anzahl abzufragender Dienste pro Knoten	1
Anzahl angebotener Dienste pro Knoten	1
Anteil Angreifer (m)	0%
Simulationsläufe	pro Dimension und Knotenanzahl 20

Tabelle 6.3: Parameter zur Untersuchung der Anzahl verschiedener Nachbarn

Da primär die Struktur des Overlays untersucht wurde und die Belastung des Netzes durch den durch Lookup- und Insert-Operationen verursachten Kommunikationsaufwand in diesem Experiment nicht interessant ist, wurde zu Gunsten einer schnellen Simulation lediglich ein Dienst pro Knoten angeboten und ein Dienst pro Knoten abgefragt. Die durchschnittliche Anzahl der verschiedenen Nachbarn ist in Abbildung 6.5 zu sehen, zusammen mit der theoretischen Anzahl von verschiedenen Nachbarn

(2d). Für jede Knotenanzahl ist zu beobachten, dass die Anzahl verschiedener Nachbarn ab einer gewissen Dimension nicht mehr zunimmt. Dies ist dadurch bedingt, dass nicht genügend Knoten zur Verfügung stehen, um in allen Dimensionen verschiedene Nachbarn zu haben. Um die Aussagekraft des Durchschnitts zu unterstreichen werden in den Tabellen 6.4(a), 6.4(b), 6.4(c) und 6.4(d) die minimale und maximale Anzahl verschiedener Nachbarn für SCANs mit 100, 500, 1000 und 1500 Knoten dargestellt. Darüber hinaus werden auch noch das 10-Perzentil, der Median und das 90-Perzentil aufgeführt. Abbildung 6.6 zeigt diese Werte grafisch beispielhaft für 500 Knoten: die vertikale Linie verläuft von der minimalen Anzahl verschiedener Nachbarn zur maximalen Anzahl verschiedener Nachbarn. Die Box beinhaltet den Median (schwarzer horizontaler Strich) und wird durch das 10-Perzentil nach unten und durch das 90-Perzentil nach oben begrenzt. 80% aller Knoten haben also eine Anzahl verschiedener Nachbarn, die innerhalb der Box liegt. Median, 10- und 90-Perzentil bleiben ab einer festgelegten Dimensionszahl nahezu konstant, d. h. die Aussage gilt für den größten Teil der Knoten. Weiterhin ist zu beobachten, dass starke Ausreißer vorkommen können, insbesondere bei den Maxima. Hierbei handelt es sich um sehr große Zonen, die sehr viele Nachbarn haben. Allerdings nimmt nur ein sehr kleiner Teil der Knoten diese Extremwerte an.

Aus dem Experiment folgert also, dass die Anzahl an verschiedenen Nachbarn bei fester Gesamtknotenanzahl ab einer gewissen Anzahl von Dimensionen auch durch Erhöhung von d für den Großteil der Knoten nicht mehr gesteigert werden kann.

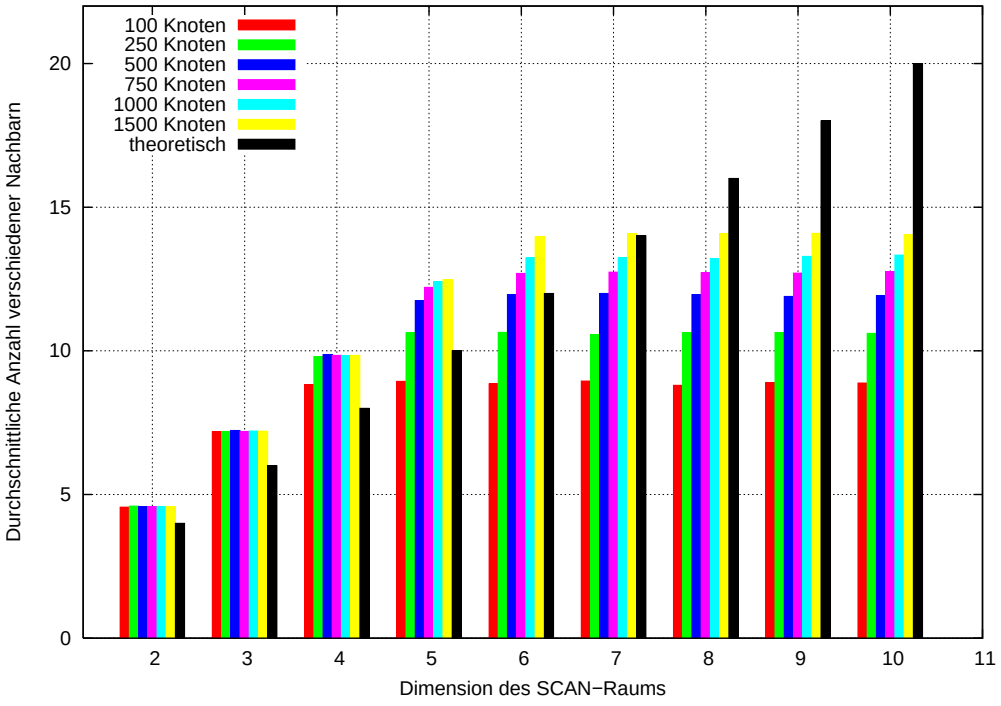


Abbildung 6.5: Durchschnittliche Anzahl verschiedener Nachbarn pro Zone bei verschiedener Gesamtknotenanzahl

Tabelle 6.4: Minimum (min), Maximum (max), 10-Perzentil (10p), Median (med) und 90-Perzentil (90p) für die Anzahl verschiedener Nachbarn in SCANs mit verschiedener Gesamtknotenanzahl

(a) 100 Knoten

d	MIN	MAX	10P	MED	90P
2	4	10	4	4	6
3	6	23	6	6	9
4	6	27	7	8	11
5	6	33	7	8	12
6	6	23	7	8	12
7	6	26	7	8	11
8	6	30	7	8	11
9	6	31	7	8	11
10	6	35	7	8	11

(b) 500 Knoten

d	MIN	MAX	10P	MED	90P
2	4	14	4	4	6
3	6	24	6	7	10
4	8	41	8	9	13
5	8	43	10	10	16
6	8	39	10	11	16
7	9	52	10	11	16
8	8	49	10	11	16
9	8	53	10	11	16
10	9	53	10	11	16

(c) 1000 Knoten

d	MIN	MAX	10P	MED	90P
2	4	11	4	4	6
3	6	29	6	7	10
4	8	43	8	9	13
5	9	48	10	11	17
6	9	62	11	12	18
7	9	65	11	12	18
8	9	52	11	12	18
9	9	48	11	12	18
10	9	63	11	12	18

(d) 1500 Knoten

d	MIN	MAX	10P	MED	90P
2	4	13	4	4	6
3	6	27	6	7	9
4	8	41	8	9	13
5	10	49	10	11	17
6	10	82	11	12	18
7	10	65	11	13	18
8	10	63	11	13	18
9	10	67	11	13	18
10	10	61	11	13	18

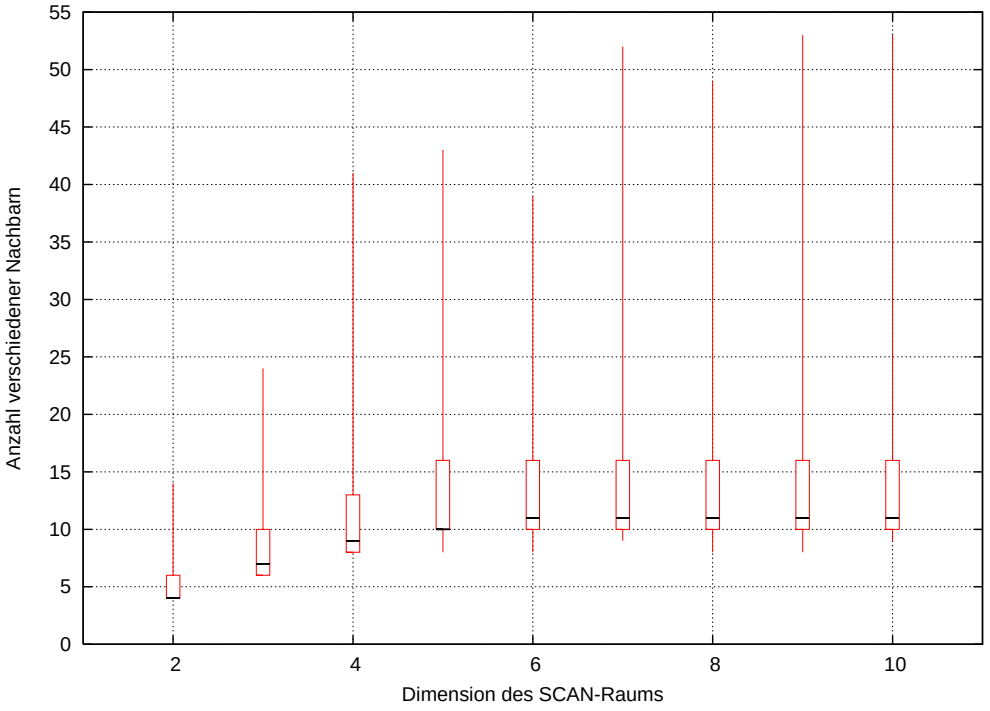


Abbildung 6.6: Verteilung der Anzahl von Nachbarn einer Zone, dargestellt durch minimale Anzahl, maximale Anzahl, Median, 10-Perzentil und 90-Perzentil.

Aber nicht nur die Anzahl an verschiedenen Nachbarn ist für MPX von Interesse, sondern auch die Länge der Overlay-Routing-Pfade. Diese wird wie gesagt durch d beeinflusst. Um die durchschnittliche Pfadlänge zu ermitteln wurde im obigen Experiment für jede versendete Overlay-Nachricht ermittelt, wie viele Overlay-Hops sie bis zum Ziel zurückgelegt hat. Da im Experiment jeder Knoten lediglich eine Dienst anbietet und abfragt ist sichergestellt, dass alle Knoten gleich gewichtet in die Berechnung eingehen. Wie nach Formel 2.1 zu erwarten, sinkt die mittlere Pfadlänge bei steigender Dimension d stark ab. Ab einem gewissen Wert für d (hier: d=5) führt eine weitere Erhöhung von d zu nur noch unwesentlich kürzeren Pfaden.

In diesem Abschnitt wurden zwei wichtige Erkenntnisse zur Wahl der Anzahl der Dimension d eines SCANs gewonnen: Erstens steigt ab einem gewissen Wert von d die Anzahl von verschiedenen Nachbarn nicht mehr, weswegen auch ab diesem Wert eine weitere Erhöhung von d keinen nennenswerten Anstieg mehr bei der Anzahl zonendisjunkter Pfade bewirkt. Zweitens sinkt auch die durchschnittliche Overlay-Routing-Pfadlänge ab einer gewissen Dimension d nicht mehr nennenswert.

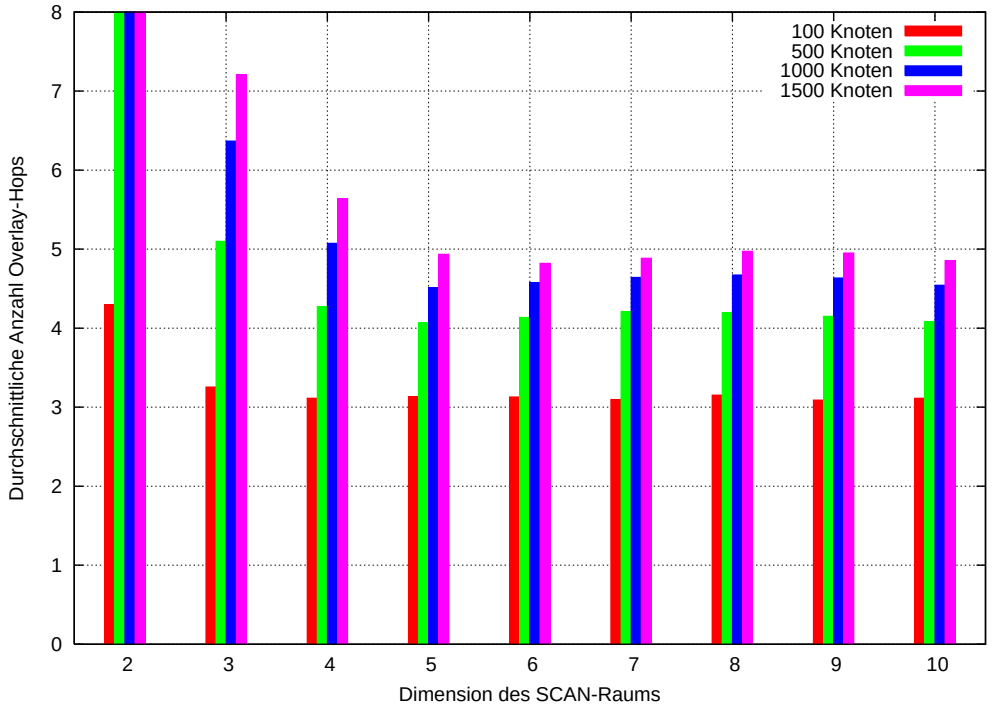


Abbildung 6.7: Durchschnittliche Pfadlänge in Overlay-Hops

6.1.3 Parameterisierung des MPX-Verfahrens

Neben der Anzahl von zonendisjunkten Pfaden, die maßgeblich von der Dimension d des SCAN-Raums und der Anzahl an verschiedenen Nachbarn abhängt, spielen die Parameter von MPX eine wichtige Rolle für die Lookup-Wahrscheinlichkeit, weswegen sie in diesem Abschnitt Gegenstand einer Voruntersuchung sind. Da MPX ein (k,n)-Schwellenwertverfahren einsetzt, bei dem mindestens k von n Schlüsselteilen benötigt werden, um den Schlüssel zu rekonstruieren, muss untersucht werden, wie die Parameter k und n für MPX günstig gewählt werden können. Dazu wurden Simulationen mit 500 Knoten durchgeführt in einem SCAN der Dimension $d = 7$. Für größere Dimensionen ist keine Verbesserung des MPX-Verfahrens mehr zu erwarten, da die Anzahl an verschiedenen Nachbarn selbst für die größte betrachtete Knotenanzahl von 1500 Knoten in diesem Fall nicht mehr zunimmt (siehe Abbildung 6.5). Es wurden jeweils 10 Simulationsläufe durchgeführt. Die Ergebnisse erwiesen sich nach 10 Simulationsläufen als aussagekräftig genug. Tabelle 6.5 fasst die Parameter für diesen Versuch noch einmal zusammen.

Parameter	Einstellung
Gesamtknotenanzahl	500
Dimension d	7
Gesamtanzahl Schlüsselteile (n)	$1, 2, 3, 4, 5, 6, 7$
Anzahl zur Rekonstruktion benötigte Schlüsselteile (k)	$1, 2, 3, 4, 5, 6, 7$, wobei $k \leq n$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	5%
Parametermenge für Sicherheitsverfahren	TinySec (TS)
Simulationsläufe	pro (n, k)-Paar 10

Tabelle 6.5: Parameter der Simulation zur Untersuchung der Parameterisierung des MPX-Verfahrens

Bedingung	Ereignis
$k \leq i$	Schlüsselaustausch fehlgeschlagen, Fehlschlag nicht erkannt, Schlüssel infiltriert
$n - k < i < k$	Schlüsselaustausch fehlgeschlagen, Fehlschlag erkannt
$i \leq n - k$	Schlüsselaustausch erfolgreich

Tabelle 6.6: Simulation eines zusammenarbeitenden Worst-Case-Angreifers bei MPX: Bei i von Angreifern korrumpierten Schlüsselteilen werden die Bedingungen von oben nach unten geprüft bis eine Bedingung erfüllt ist.

Für die Simulationen wurde ein zusammenarbeitender Angreifer folgendermaßen simuliert: Angreifer markieren Nachrichten als korrumpiert, verwerfen diese jedoch nicht. Bei der Rekonstruktion des Schlüssels beim Empfänger wird von einem Worst-Case-Angreifer ausgegangen, wie in Tabelle 6.6 zusammengefasst dargestellt:

- Ist es einem zusammenarbeitenden Angreifer gelungen k oder mehr Nachrichten mit Schlüsselteilen zu korrumpieren, so kann er den Schlüssel rekonstruieren, da er über genügend Schlüsselteile verfügt. In diesem Fall wird also davon ausgegangen, dass der Angreifer die Nachrichten abgehört aber nicht verändert hat. Der Schlüssel gilt in diesem Fall als infiltriert, was vom Empfänger nicht bemerkt wird. Alle Nachrichten, für die im weiteren Verlauf dieser Schlüssel verwendet wird, werden als korrumpiert betrachtet.

- Gelingt es einem Angreifer nicht, mehr als k Schlüsselteile abzuhören, so kann es ihm aber immer noch gelingen, den Schlüsselaustausch zu verhindern: hat ein Angreifer mehr als $n - k$ Nachrichten mit Schlüsselteilen aber weniger als k Nachrichten mit Schlüsselteile korrumpiert, so ist es dem Angreifer möglich, den Schlüsselaustausch zu unterbinden, indem er die Schlüsselteile verändert. Die Rekonstruktion des Schlüssels scheitert in diesem Fall, da dem Empfänger zu wenig nicht korrumpierte Schlüsselteile zur Verfügung stehen. Ein Knoten

kann dies jedoch erkennen, da ein verifizierendes Schwellenwertverfahren eingesetzt wird.

- Nur wenn die Anzahl der korrumpierten Schlüsselteile $n - k$ oder weniger und kleiner als k ist, wurde der Schlüssel erfolgreich übertragen und gilt als intakt. Der Empfänger kann die zur Rekonstruktion des Schlüssel benötigten k korrekt übertragenen Schlüsselteile erkennen, da ein verifizierendes Geheimnisteilungsverfahren eingesetzt wird.

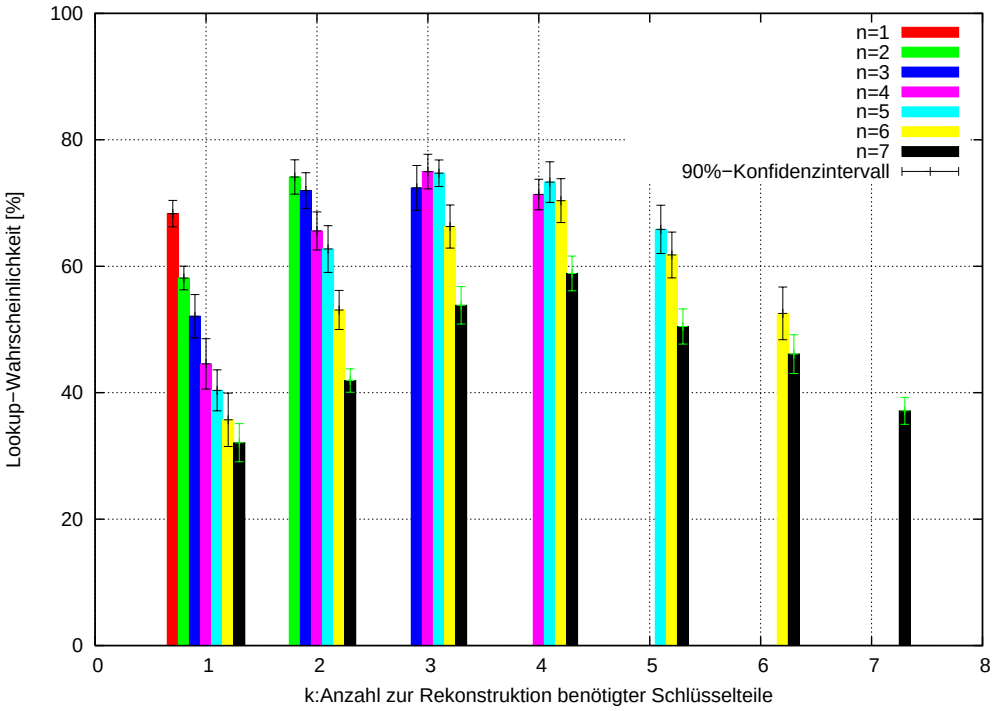


Abbildung 6.8: Lookup-Wahrscheinlichkeit bei Verwendung von MPX und verschiedenen Werten für k (Anzahl der zur Rekonstruktion des Schlüssels mindestens benötigten Schlüsselteile) und n (Gesamtanzahl an Schlüsselteilen). Dabei gilt $k \leq n$.

Abbildung 6.8 zeigt die im Experiment ermittelte durchschnittliche Lookup-Wahrscheinlichkeit für verschiedene Werte von n und k. Dabei wurde $n \in \{1, ...7\}, n \in \mathbb{N}$ gewählt und $k \in \{1, ...7\}, k \in \mathbb{N}$ mit der Bedingung $k \leq n$. Dabei ist zu beachten, dass MPX mit den Parametern $k = 1, n = 1$ einem Single Path Key Exchange (SPX) entspricht. Aus Abbildung 6.8 wird deutlich, dass die durchschnittliche Lookup-Wahrscheinlichkeit für $k = 1$ mit steigendem n sinkt. Dies ist dadurch bedingt, dass mit steigendem n mehr disjunkte Pfade verwendet werden. Jeder neue Overlay-Routing-Pfad bringt jedoch eine Menge von Overlay-Knoten mit, die potentielle Angreifer sind. Einem zusammenarbeitenden Angreifer reicht bereits ein korrumpierter Knoten auf einem der Overlay-Routing-Pfade, um den ausgetauschten Schlüssel zu infiltrieren.

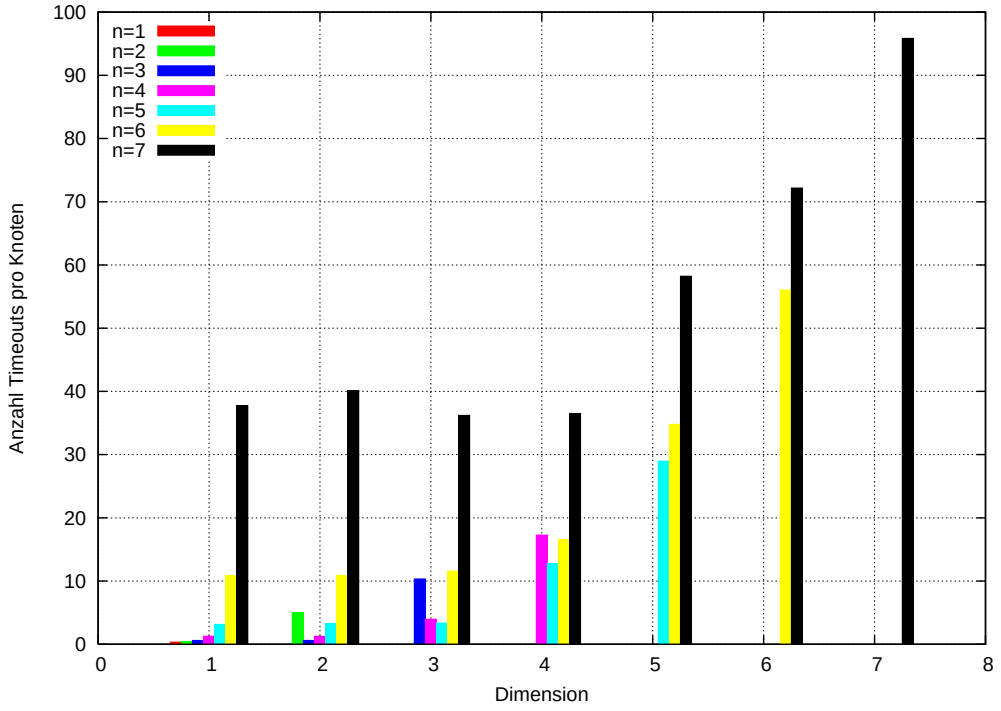


Abbildung 6.9: Anzahl Timeouts der Lookup- und Insert-Operationen pro Knoten für verschiedene Anzahl von Dimensionen d

Weiterhin zeigt sich, dass die Erfolgswahrscheinlichkeit einer Lookup-Operation bei konstantem n und variablem k zuerst ansteigt und dann wieder abfällt. Der Maximalwert für die Lookup-Wahrscheinlichkeit liegt ungefähr bei $k = n/2$. Der Anstieg im Bereich $1 \leq k \leq n/2$ ist dadurch bedingt, dass bei höherem k die Wahrscheinlichkeit eines Angreifers auf entsprechend vielen Pfaden sinkt. Im Bereich $n/2 < k \leq n$ ist der Abfall damit zu erklären, dass die entsprechende Anzahl von Angreifern nun den Schlüsselaustausch verhindern kann. Um diese Aussage zu untermauern wird in Abbildung 6.9 die Anzahl von Timeouts bei der Lookup- und Insert-Operation im Experiment gezeigt. Ein Timeout tritt immer dann auf, wenn der den Schlüsselaustausch initiierende Knoten den Schlüsselaustausch nicht innerhalb eines vorgegebenen Intervalls abschließt. Dies kann z. B. passieren, wenn Nachrichten verloren gehen. Ein Grund hierfür kann sein, dass Knoten Nachrichten korrumpieren, um den Schlüsselaustausch zu verhindern. Die Anzahl der Timeouts der Lookup- und Insert-Operation nimmt im Experiment für $k > n/2$ deutlich zu. Dadurch kommt es beim Empfänger zum Timeout und der Schlüsselaustausch wird erneut angestoßen. Da bei jedem Schlüsselaustausch die zu verwendenden Pfade zufällig bestimmt werden, ist es möglich, dass der erneute Schlüsselaustausch funktioniert, weil andere Pfade gewählt wurden, auf denen sich weniger Angreifer befinden. Dies funktioniert natürlich nur, wenn noch genügend Pfade zur Verfügung stehen, die im vorigen Versuch nicht genutzt wurden. Dies ist bei hohen Werten von n nicht der Fall, weswegen auch die

durchschnittliche Lookup-Wahrscheinlichkeit von $n = 6$ und $n = 7$ im Experiment meist niedriger ist, als für $1 \leq n \leq 5$.

In Tabelle 6.7 sind die in Abbildung 6.8 visualisierten Werte aufgeführt. Dabei sind diejenigen Tabellenzellen hinterlegt, für die gilt: $k = \lfloor n/2 \rfloor + 1$. Fett ist das Maximum der Lookup-Wahrscheinlichkeit einer Zeile hinterlegt. Bis auf $n = 3$ liefert bei der Wahl von $k = \lfloor n/2 \rfloor + 1$ MPX immer die höchste Lookup-Wahrscheinlichkeit. Im Fall von $n = 3$ ist die Abweichung zum Maximum mit 0.41 sehr gering und liegt innerhalb des 90%-Konvergenzintervalls, so dass $k = \lfloor n/2 \rfloor + 1$ als gute Wahl für den Parameter k angesehen werden kann.

-	K=1	K=2	K=3	K=4	K=5	K=6	K=7
n=1	**68.33%**						
n=2	58.14%	**74.12%**					
n=3	52.09%	71.98%	**72.39%**				
n=4	44.55%	65.58%	**74.98%**	71.35%			
n=5	40.37%	62.73%	**74.72%**	73.31%	65.83%		
n=6	35.71%	53.09%	66.28%	**70.38%**	61.79%	52.54%	
n=7	32.08%	41.93%	53.83%	**58.86%**	50.47%	46.11%	37.12%

Tabelle 6.7: Lookup-Wahrscheinlichkeit für verschiedene Werte von k und n. Das Maximum einer Zeile ist fett gedruckt, die Zelle, für die gilt $k = \lfloor n/2 \rfloor + 1$ ist farbig hinterlegt.

Neben der durchschnittlichen Lookup-Wahrscheinlichkeit ist bei der Wahl der Parameter k und n auf den erzeugten Kommunikationsaufwand zu achten. In Abbildung 6.10 ist die Anzahl von im Overlay gesendeten Bytes pro Knoten aufgetragen, die während der gesamten Simulationszeit von 15000 Sekunden im Rahmen des obigen Experiments für alle Lookup-Operationen versendet wurden. Da für jede Lookup-Anfrage n Schlüsselteile versendet werden, steigt der Kommunikationsaufwand auch mit steigendem n. Sehr deutlich ist der Unterschied im Kommunikationsaufwand zwischen $n = 1$ bis $n = 5$ einerseits und $n = 6$ und $n = 7$ andererseits. Abbildung 6.11 zeigt die Delivery Ratio für das Experiment. Die Delivery Ratio berechnet sich dabei als Anzahl im Overlay gesendeter Bytes geteilt durch Anzahl im Overlay empfangener Bytes. Deutlich sichtbar wird, dass die Delivery Ratio für $n = 6$ und $n = 7$ für alle Werte von k deutlich niedriger ist als bei allen anderen Werten für n. Der deutliche Anstieg des Kommunikationsaufwands in Abbildung 6.10 resultiert also aus Übertragungswiederholungen, die von den Knoten nach einem Timeout angestoßen werden. Idealerweise sollte also ein möglichst kleiner n-Wert gewählt werden.

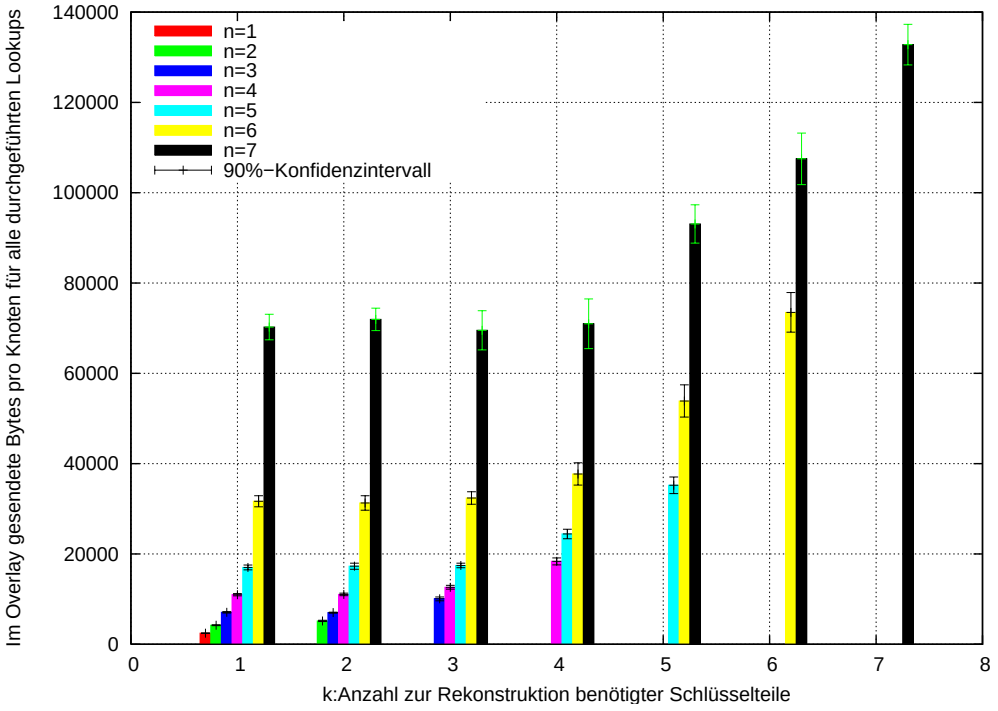


Abbildung 6.10: Untersuchung des Kommunikationsaufwand (gemessen in im Overlay versendeten Bytes pro Knoten während der gesamten Simulationszeit) verschiedener (k,n)-Multi-Path-Key-Exchange-Verfahren beim Lookup

Wie oben bereits festgestellt wächst die Wahrscheinlichkeit, dass ein Schlüsselaustausch wegen Angreifern auf den Pfaden scheitert, der Schlüssel aber nicht kompromittiert wird, ab $k = n/2$ stark an. Da der Empfänger die korrumpierten Schlüsselteile erkennt und deshalb den Protokollablauf unterbricht, tritt beim Sender ein Timeout während der Lookup-Operation auf. Der Sender stößt als Reaktion darauf den Schlüsselaustausch erneut an. Dadurch wird weiterer Kommunikationsaufwand erzeugt, weswegen auch der k-Wert nicht zu groß gewählt werden sollte. Der erneute Schlüsselaustausch zeigt sich für alle Werte von n in Abbildung 6.10 durch einen steigenden Kommunikationsaufwand ab $k = \lfloor n/2 \rfloor + 1$

Basierend auf den Untersuchungen aus diesem Kapitel werden für die weiteren Simulationen, falls nicht anders angegeben, die Parameter folgendermaßen gewählt: Für k wird $k = \lfloor n/2 \rfloor + 1$ gewählt. Für n werden Werte zwischen 2 und $\lfloor d/2 \rfloor + 1$ in die Überlegungen einbezogen, da bei diesen Werten nur ein Teil der zonendisjunkten Pfade für den MPX-Schlüsselaustausch genutzt werden. Scheitert der Schlüsselaustausch, und wird dies erkannt, so stehen noch genügend ungenützte Pfade zur Verfügung.

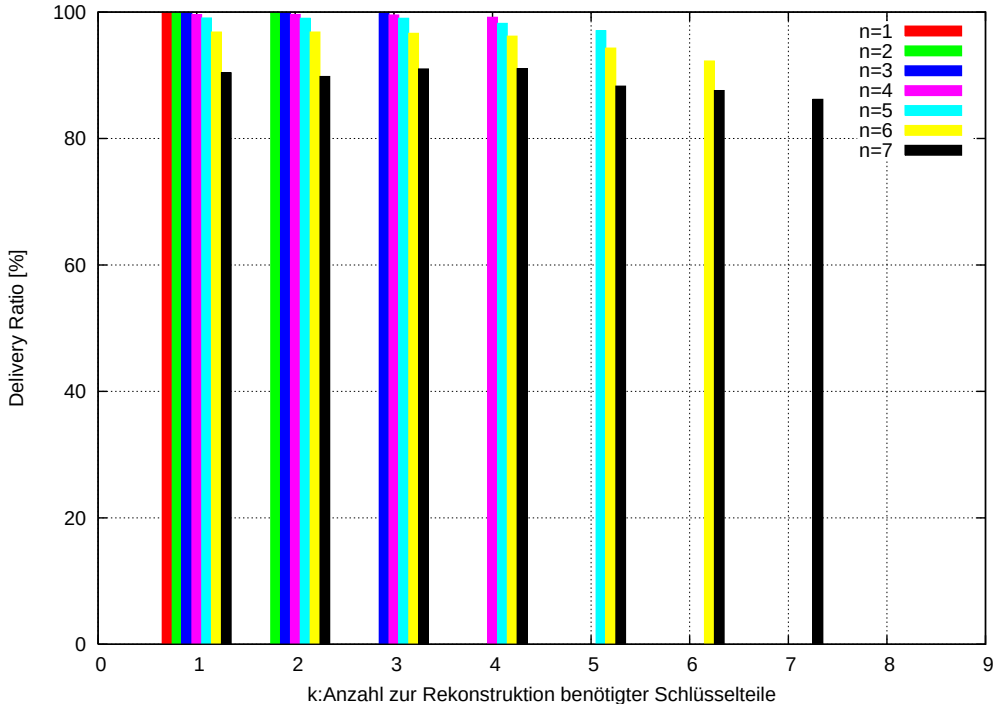


Abbildung 6.11: Delivery Ratio für verschiedene Werte von n und k.

6.1.4 Leistungsanalyse Dienste-Suche

Im Rahmen dieses Abschnitts wird die Leistung der Dienste-Suche untersucht. Als Metrik der Leistungsfähigkeit kommt die *Lookup-Wahrscheinlichkeit* zum Einsatz: Je höher diese Wahrscheinlichkeit ist, desto besser arbeitet die Dienste-Suche. Dabei macht nur eine Betrachtung der *durchschnittlichen* Lookup-Wahrscheinlichkeit Sinn, denn die individuelle Lookup-Wahrscheinlichkeit für jeden Knoten schwankt, da sie stark von der Lage der Angreifer in SCAN abhängt. So hat ein Knoten, der nur Angreifer als Nachbarn hat, die individuelle Lookup-Wahrscheinlichkeit 0 %, da jede Nachricht von einem Angreifer manipuliert wird. Um zu untersuchen, inwieweit sich die individuellen Lookup-Wahrscheinlichkeiten voneinander unterscheiden, wurde ein Experiment mit 100, 250, 500, 1000 und 1500 Knoten unter Verwendung von SPX für die Lookup- und Insert-Operation durchgeführt. Die Parameter der Simulation sind in Tabelle 6.8 aufgeführt.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 1000, 1500$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1
Simulationsläufe	30 je Knotenanzahl und je Anteil Angreifer

Tabelle 6.8: Parameter der Simulationen zur Untersuchung der individuellen Lookup-Wahrscheinlichkeiten

In Abbildung 6.12 ist die relative Standardabweichung von der durchschnittlichen Lookup-Wahrscheinlichkeit bei steigendem Anteil an Angreifern für dieses Experiment zu sehen. Durch die steigende relative Standardabweichung wird der Einfluss von Angreifern auf die Unterschiede zwischen den individuellen Lookup-Wahrscheinlichkeiten der einzelnen Knoten deutlich sichtbar. Im Weiteren wird deshalb die durchschnittliche Lookup-Wahrscheinlichkeit verwendet, um die Leistungsfähigkeit der Dienste-Suche zu bewerten.

Als Metrik für den durch die Dienste-Suche verursachten Kommunikationsaufwand wird die durchschnittliche *Anzahl im Overlay gesendeter Bytes pro Knoten* betrachtet.

Falls nicht anders angegeben werden alle folgenden Untersuchungen in einem SCAN der Dimension $d = 7$ durchgeführt. Dieser Wert von d ist auch für 1500 Knoten geeignet, wie die vorherigen Experimente gezeigt haben. Für MPX werden die Werte $n = \lfloor d/2 \rfloor + 1 = 4$ und $k = \lfloor n/2 \rfloor + 1 = 3$ verwendet, da diese für $d = 7$ laut den Voruntersuchungen zu einer hohen Lookup-Wahrscheinlichkeit führt. Tabelle 6.9 fasst die Standardparameter für alle folgenden Experimente noch einmal zusammen.

Standardparameter	Einstellung
Dimension d	7
Anzahl Schlüsselteile, die für MPX verwendet werden (n)	4
Anzahl Schlüsselteile, die zur Rekonstruktion des Schlüssels notwendig sind (k)	3
Simulationsläufe	30

Tabelle 6.9: Falls nicht anders angegeben, werden für die folgenden Versuche die Parameter aus dieser Tabelle verwendet

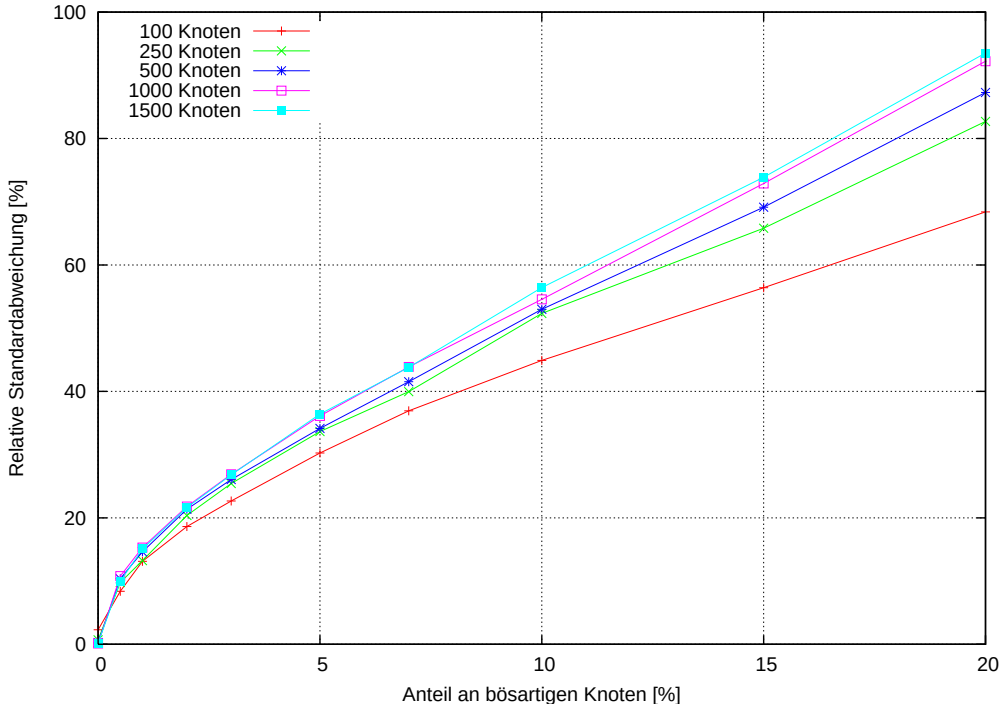


Abbildung 6.12: Relative Standardabweichung bei Berechnung der durchschnittlichen Lookup-Wahrscheinlichkeit

6.1.4.1 Single-Path-Key-Exchange

Um die Leistungsfähigkeit von SPX zu untersuchen wurde die durchschnittliche Lookup-Wahrscheinlichkeit für SCANs mit einer Gesamtknotenanzahl von 100, 250, 500, 1000 und 1500 Knoten untersucht. Tabelle 6.10 fasst die Parameter der Simulationen noch einmal zusammen.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 1000, 1500$
Anzahl abzufragende Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotene Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS), Full Security (FS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1
Simulationsläufe	Jeweils 30

Tabelle 6.10: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit von SPX

Abbildung 6.13 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für dieses Experiment bei bei Verwendung der Parametermenge Tiny Security. Die durchschnittliche Lookup-Wahrscheinlichkeit unter Verwendung der Parametermenge Full Security ist in Abbildung 6.14 zu sehen. Die 90%-Konfidenzintervalle für 500 und 1000 Knoten und Verwendung der Parametermenge Tiny Security sind der besseren Übersichtlichkeit wegen getrennt in Abbildung 6.15 dargestellt. Wie auf Grund der theoretischen Vorüberlegungen zu erwarten, nimmt die durchschnittliche Lookup-Wahrscheinlichkeit mit zunehmendem Anteil von Angreifern ab. In den Grafiken zeigt sich, dass ein größerer Anteil an Angreifern einen um so stärkeren Einfluss auf die Lookup-Wahrscheinlichkeit hat, je mehr Knoten im Netz vorhanden sind. Dies resultiert daraus, dass bei mehr Knoten im Netz auch die durchschnittliche Pfadlänge im Overlay länger ist, weswegen auch die Wahrscheinlichkeit höher ist, dass sich ein Angreifer auf einem Pfad befindet. Die durchschnittliche Länge der Overlay-Pfade wird für verschiedene Gesamtknotenanzahlen in Tabelle 6.11 angegeben.

Gesamtknotenanzahl	Durchschnittliche Pfadlänge
100	3.1834
250	3.7282
500	4.1917
1000	4.6562

Tabelle 6.11: Gemessene durchschnittliche Pfadlänge bei verschiedener Gesamtknotenanzahl

Abbildung 6.16 zeigt dagegen die Lookup-Wahrscheinlichkeit nicht unter einem Anteil von Angreifern sondern unter einer Anzahl von Angreifern, d. h. in Abbildung 6.16 sind absolute Werte bezüglich der Angreifer angegeben, während in Abbildung 6.13 relative Werte angegeben sind. Da in den Simulationen maximal 20 % der Knoten des SCAN Angreifer waren, enden die Kurven bei den entsprechenden Meßwerten

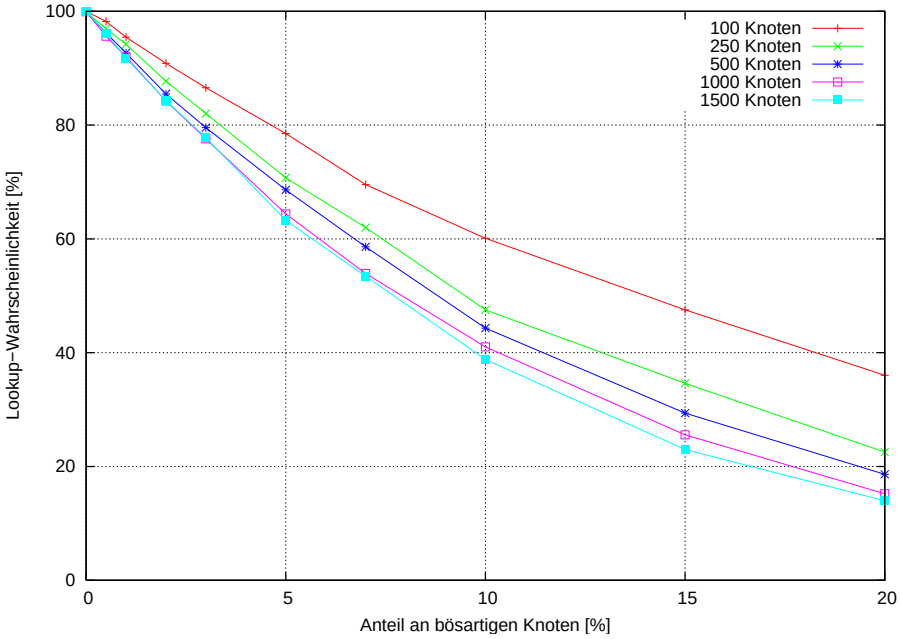


Abbildung 6.13: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX mit der Parametermenge Tiny Security und steigendem Anteil an Angreifern

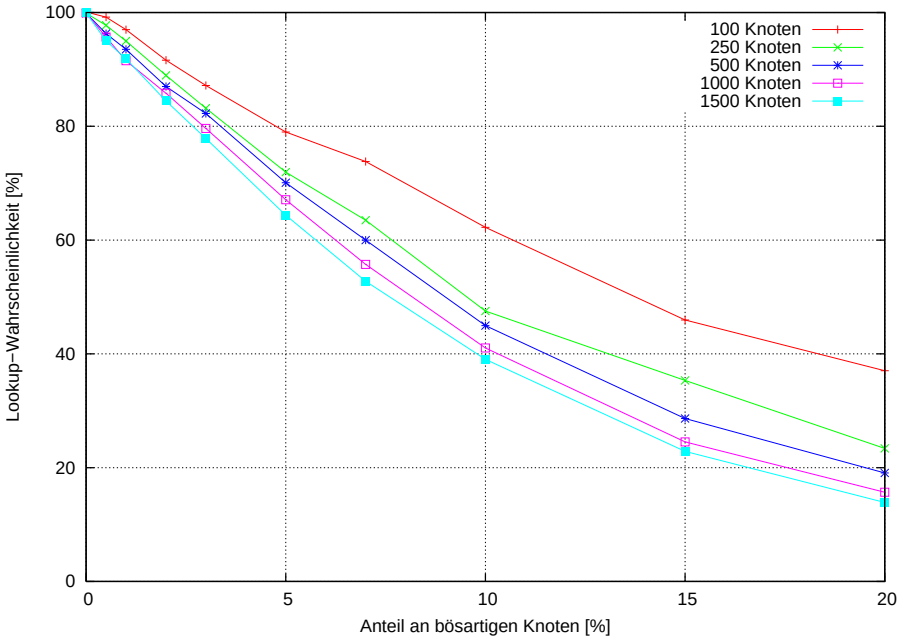


Abbildung 6.14: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX mit der Parametermenge Full Security und steigendem Anteil an Angreifern

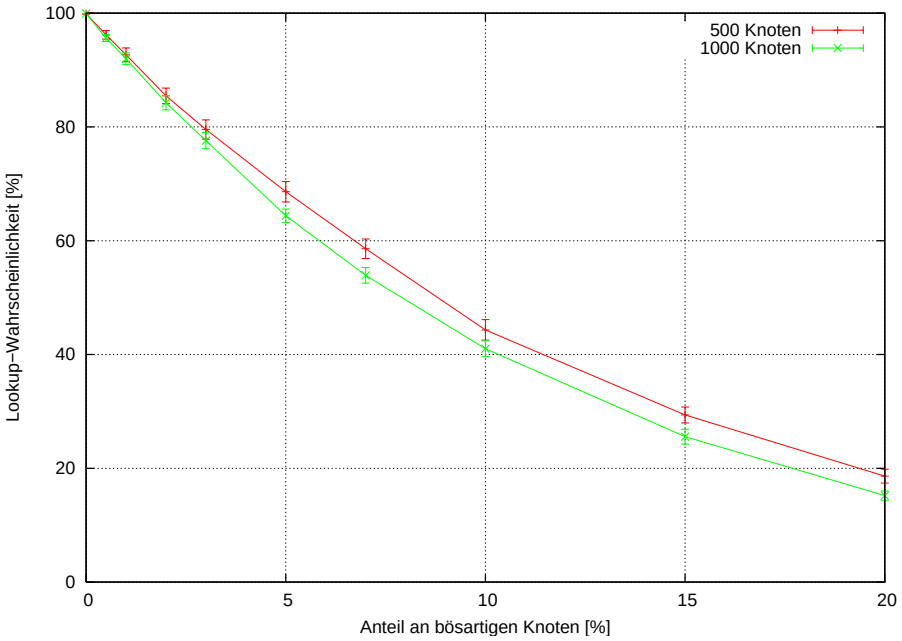


Abbildung 6.15: Durchschnittliche Lookup-Wahrscheinlichkeit mit 90%-Konfidenzintervallen bei Verwendung von SPX und der Parametermenge Tiny Security (TS)

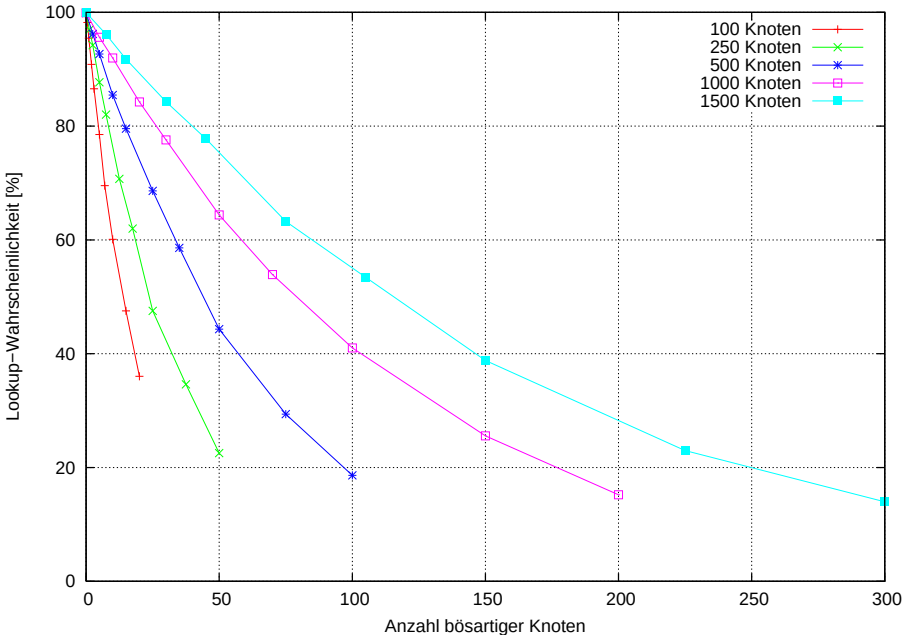


Abbildung 6.16: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX und steigender absoluten Anzahl an Angreifern unter Verwendung der Parametermenge Tiny Security (TS)

(also bei 100 Knoten bei 20 Angreifern, bei 250 Knoten bei 50 Angreifern etc). Aus Abbildung 6.16 wird sichtbar, dass mehr Knoten auch ein höheres Maß an Sicherheit gewährleisten: Während die Lookup-Wahrscheinlichkeit für 250 Knoten und 50 Angreifern lediglich knapp über 22% liegt, ist die Lookup-Wahrscheinlichkeit für ein SCAN mit 1000 Knoten bei der gleichen absoluten Anzahl von Angreifern immer noch bei 65 %. Dieses Ergebnis war zu erwarten: wird in einem SCAN-Raum, der in 250 Zonen eingeteilt ist, eine Zone übernommen, so hat diese im Durchschnitt eine größere Fläche, also damit auch mehr Einfluss, als eine Zone in einem SCAN mit 1000 Knoten. Der Einfluß einer übernommenen Zone (und damit eines korrumpierten Knotens) sinkt also mit der Knotenanzahl.

Da ein Angreifer in der Realität einen gewissen Aufwand hat, um einen Knoten zu korrumpieren, er also mit einem begrenzten Arbeitsaufwand auch nur eine begrenzte Anzahl an Knoten unabhängig von der Netzgröße infiltrieren kann, folgt aus dem letzten Ergebnis, dass die Sicherheit bei Einsatz von SCAN mit steigender Knotenanzahl steigt.

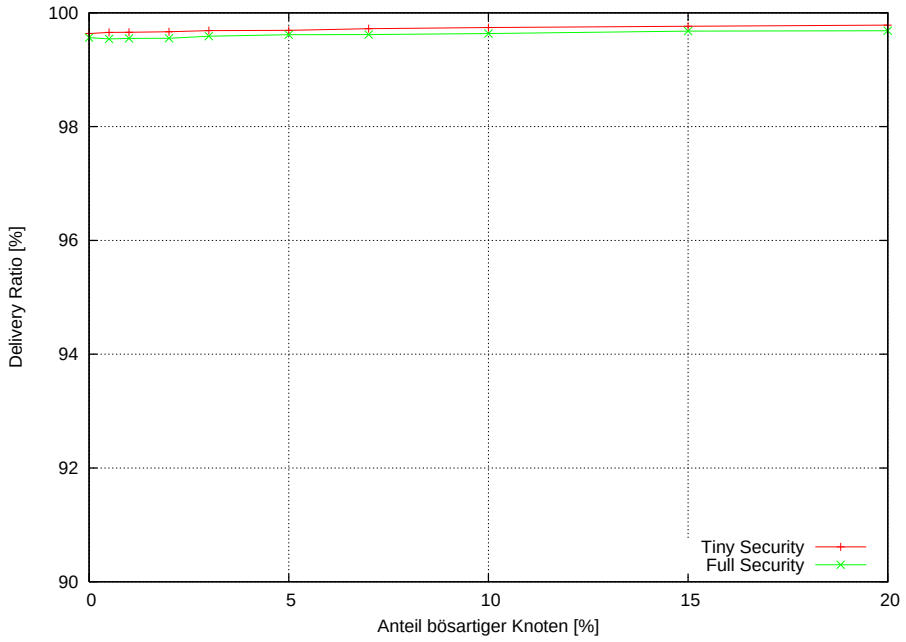


Abbildung 6.17: Delivery Ratio bei Verwendung der Parametermenge Tiny Security und Full Security und 1000 Knoten

Vergleicht man die durchschnittliche Lookup-Wahrscheinlichkeit von SPX unter Verwendung von Tiny Security mit der durchschnittlichen Lookup-Wahrscheinlichkeit von SPX unter Verwendung von Full Security, so sind nur minimale Unterschiede festzustellen, die innerhalb der Konfidenzintervalle liegen. Die Ähnlichkeit der Ergebnisse rührt daher, dass sich Tiny Security und Full Security lediglich in der Länge der übertragenen Nachrichten unterscheiden. Ein Unterschied in der durchschnittlichen Lookup-Wahrscheinlichkeit ist bei Verwendung von Tiny Security im Vergleich zur

Verwendung von Full Security nur dann zu erwarten, wenn die größere Nachrichtenlänge von Full Security zu Kollisionen bei der drahtlosen Übertragung führt und somit Nachrichtenwiederholungen verursacht. Abbildung 6.17 zeigt die Delivery Ratio für die Simulationen mit 1000 Knoten aus dem Experiment. Selbst bei dieser großen Anzahl von Knoten liegt die Delivery Ratio immer noch bei über 99%. Beim Einsatz von SPX führt die Parametermenge Full Security, aus Sicht des Secure Content Addressable Networks, also weder zu einer erhöhten noch zu einem verminderten Sicherheitsniveau (gemessen durch die durchschnittliche Lookup-Wahrscheinlichkeit), allerdings bietet Full Security aus kryptographischer Sicht mehr Schutz gegen Angreifer, z. B. auf Grund von größeren Schlüssellänge, die es einem Angreifer erschweren, über Brute-Force-Angriffe den Schlüssel zu erraten. Diese Aspekte wurden in der Simulation allerdings nicht berücksichtigt.

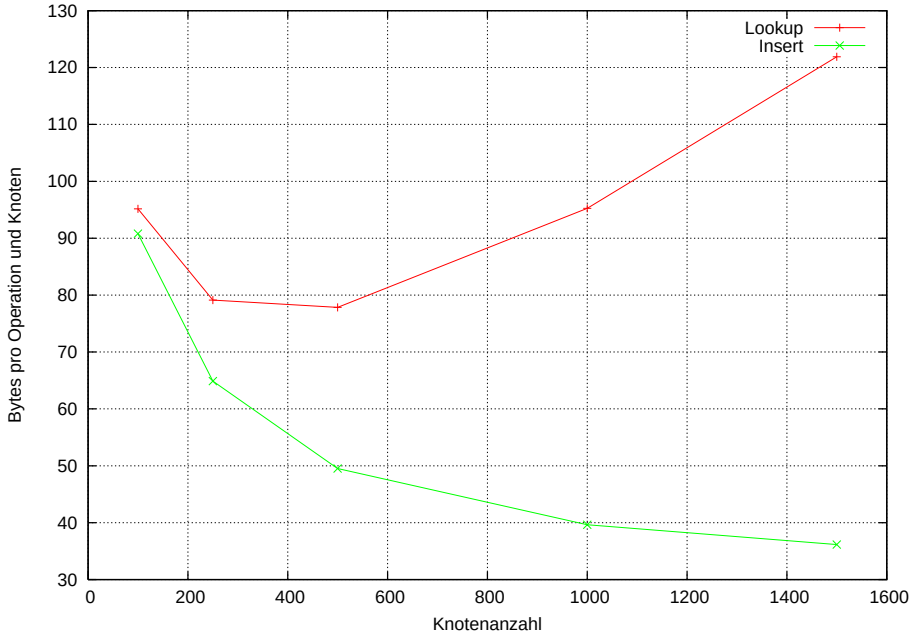


Abbildung 6.18: Durchschnittlicher Kommunikationsaufwand der Lookup- und Insert-Operation mit SPX bei Verwendung der Parametermenge Tiny Security: relative Werte

Abbildung 6.18 zeigt, wie sich der Kommunikationsaufwand von SPX unter Verwendung der Parametermenge Tiny Security pro Lookup- bzw. Insert-Operation und Knoten bei steigender Knotenanzahl verhält. Die Größe des 90%-Konfidenzintervall beträgt in allen Fällen weniger als 1.7 Bytes, weswegen das Konfidenzintervall aus Gründen der Übersichtlichkeit nicht dargestellt wird. Da der Kommunikationsaufwand jeweils auf alle Knoten im Netz umgerechnet wird, sinkt der Kommunikationsaufwand der Insert-Operation, denn es sind nicht alle Knoten an einer konkreten Insert-Operation beteiligt. Bei der Lookup-Operation beobachtet man das gleiche Verhalten bis 500 Knoten, danach steigt der Aufwand wieder an. Dies ist der Tatsache geschuldet, dass mit steigender Knotenanzahl auch die Anzahl der Dienst-

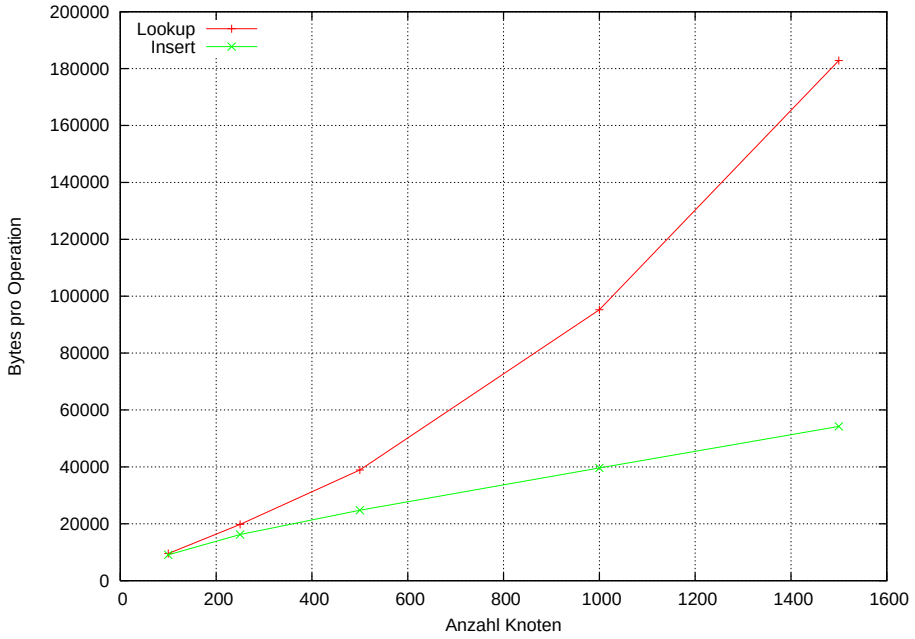


Abbildung 6.19: Durchschnittlicher Kommunikationsaufwand der Lookup- und Insert-Operation mit SPX bei Verwendung der Parametermenge Tiny Security: absolute Werte

beschreibungen zu einem Dienst zunimmt, weil mehrere Knoten diesen Dienst anbieten, und damit auch die entsprechende Liste von Dienstbeschreibungen, die von der Lookup-Operation zurückgeliefert wird, größer wird. Abbildung 6.19 zeigt die absoluten Zahlen zum Kommunikationsaufwand der Lookup- und Insert-Operation: Es wird also die Anzahl gesendeter Bytes im Overlay pro Lookup- bzw. Insert-Operation betrachtet. Es erfolgt also keine Betrachtung pro Knoten mehr. Auch hier ist das Auseinanderdriften des Kommunikationsaufwands der Lookup- und der Insert-Operation durch die längeren Dienstelisten zu erklären.

In Abbildung 6.20 wird der durchschnittliche Kommunikationsaufwand für die Lookup- und Insert-Operation bei Verwendung der Parametermengen Full Security (FS) bzw. Tiny Security (TS) dargestellt. Gezeigt wird wieder die durchschnittliche Anzahl von im Overlay gesendeten Bytes pro Lookup- bzw. Insert-Operation pro Knoten. Während bei Tiny Security die 90%-Konfidenzintervalle sehr klein sind, sind diese bei Full Security, insbesondere für 100 Knoten, relativ groß. Wie zu erwarten geht die Parametermenge Full Security mit einem deutlich höheren Kommunikationsaufwand einher als die Parametermenge Tiny Security.

Um den oben angesprochenen Einfluss der im Netz vorhandenen Dienstbeschreibungen auf die Experimente zu untersuchen wurde eine Experiment vorgenommen, in der die Diensteanzahl über alle Netzgrößen hinweg konstant gehalten wurde: Es sollten insgesamt 1500 Dienste (aus 100 verschiedenen) im Netz verteilt werden. Für 100 Knoten bot deshalb jeder Knoten gleichverteilt zwischen einem und 29 Diensten

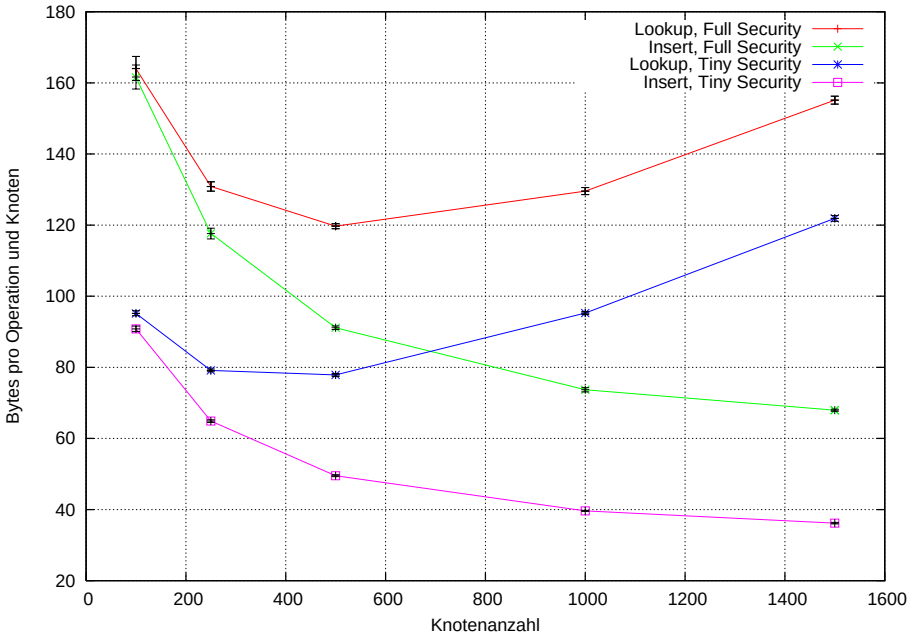


Abbildung 6.20: Durchschnittlicher Kommunikationsaufwand für die Lookup- und Insert-Operation für die beiden Parametermengen Full Security (FS) und Tiny Security (TS) im Vergleich

an, während bei 1500 Knoten jeder Knoten nur einen einzigen Dienst anbot. Analog wurden in jedem SCAN 3000 Lookup-Operationen durchgeführt. Die Parameter des Experiments sind noch einmal in Tabelle 6.12 aufgeführt.

In Abbildung 6.21 ist der Aufwand der Lookup-Operation bei konstanter Diensteanzahl in Vergleich gesetzt zum Aufwand der Lookup-Operation mit steigender Diensteanzahl, wie er in Abbildung 6.18 zu sehen war. Deutlich ist zu sehen, dass der Kommunikationsaufwand bei konstanter Diensteanzahl und steigender Knotenanzahl sinkt, d. h. der oben beschriebene Einfluss der Diensteanzahl auf die Simulationsergebnisse ist belegt.

Zusammenfassend kann man also sagen, dass der Kommunikationsaufwand pro Knoten für die Insert-Operation mit steigender Knotenanzahl sinkt, was die Skalierbarkeit bei steigender Knotenanzahl belegt. Der Kommunikationsaufwand pro Knoten für die Lookup-Operation hängt, wegen der Länge der zurückgelieferten Diensteliste, stark davon ab, wie viele Dienste im Netz angeboten werden. Bleibt die Anzahl der Dienste konstant, so sinkt der Kommunikationsaufwand mit steigender Knotenanzahl, was belegt, dass auch die Lookup-Operation mit steigender Knotenanzahl skaliert. Steigt die Anzahl von Knoten bei gleich bleibender Anzahl Angreifer, so steigt auch die Lookup-Wahrscheinlichkeit. Es hat sich also herausgestellt, dass SPX mit steigender Knotenanzahl gut skaliert und sich deshalb auch für große Sensornetze eignet.

Zu den Standardparametern aus Tabelle 6.9 zusätzliche Parameter	Einstellung
Gesamtknotenanzahl N	$100, 250, 500, 1000, 1500$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 6000/N - 1\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 3000/N - 1\}, \in \mathbb{N}$
Anteil Angreifer (m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1
Simulationsläufe	Jeweils 30

Tabelle 6.12: Parameter der Simulationen zur Untersuchung des Einflusses der Anzahl von angebotenen und abgefragten Diensten

Als eine Maßnahme, um das Ergebnis einer Lookup-Operation zu verbessern, wurde Replikation angesprochen. Um dies zu untersuchen wurde ein Experiment durchgeführt, bei dem eine Dienstbeschreibung einmal, dreimal, fünfmal oder siebenmal im SCAN hinterlegt wurde. Es wurden nur ungerade Werte gewählt, da diese für den Mehrheitsentscheid besonders günstig sind: Bei r Replikaten sind mindestens $\lfloor r/2 \rfloor + 1$ erfolgreich übertragende Dienstbeschreibungen für einen Erfolg des Mehrheitsentscheids notwendig. Ist r gerade, dann gilt: $\lfloor r/2 \rfloor + 1 = \lfloor (r+1)/2 \rfloor + 1$, d. h. für r und $r+1$ Replikate wird dieselbe Anzahl an erfolgreich übertragenen Dienstbeschreibungen benötigt. Da bei $r+1$ eine Dienstbeschreibung mehr abgefragt wird, ist ein erfolgreicher Mehrheitsentscheid wahrscheinlicher. Tabelle 6.13 gibt einen Überblick über die für die Simulation gewählten Parameter:

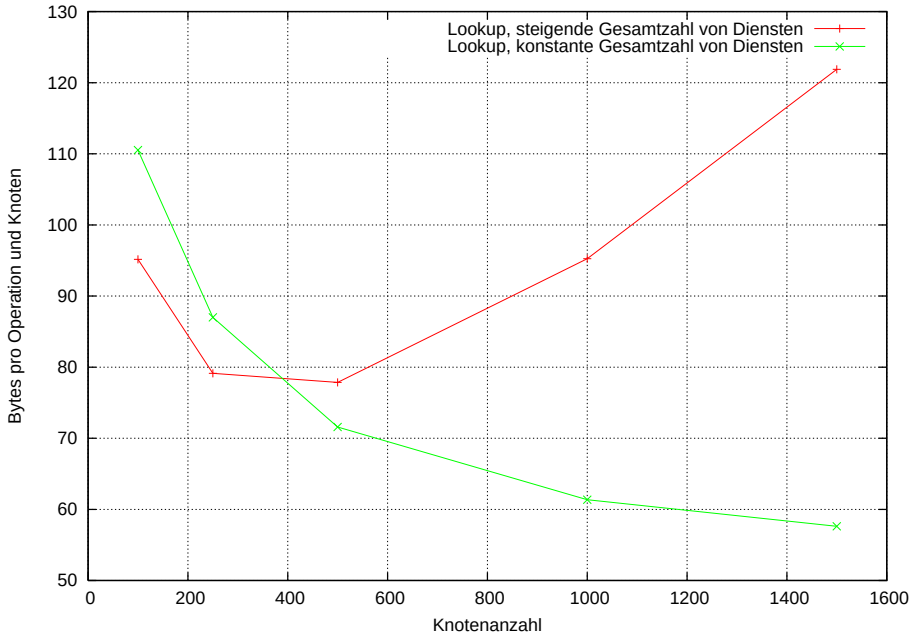


Abbildung 6.21: Durchschnittlicher Kommunikationsaufwand der Lookup-Operation bei konstanter und bei steigender Diensteanzahl.

Parameter	Einstellung
Gesamtknotenanzahl	500
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS), Full Security (FS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1,3,5,7
Simulationsläufe	Jeweils 30

Tabelle 6.13: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit von SPX mit Replikation

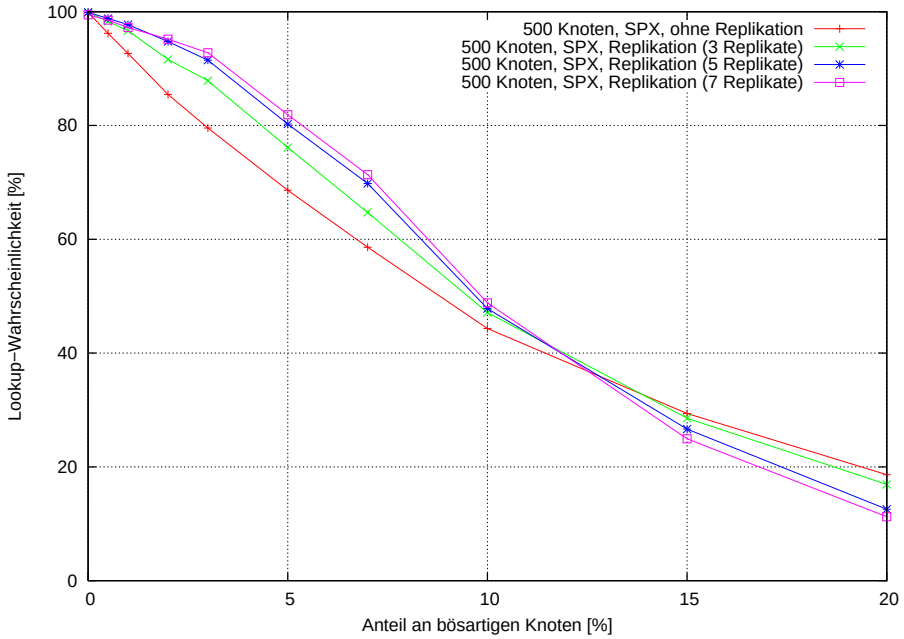


Abbildung 6.22: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX mit und ohne Replikation unter Verwendung der Parametermenge Tiny Security (TS)

Abbildung 6.22 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit des Experiments für die Parametermenge Tiny Security (TS), Abbildung 6.23 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für die Parametermenge Full Security (FS). Für beide Verfahren ist schon bei Verwendung von 3 Replikaten ein deutlichen Zugewinn bei der durchschnittlichen Lookup-Wahrscheinlichkeit für einen moderaten Anteil von Angreifern zu erkennen. Bei 5% Angreifern ergibt die Verwendung von drei Replikation z. B. einen Zugewinn von annähernd 10% bei der Lookup-Wahrscheinlichkeit. Der Zugewinn war auf Basis der theoretischen Überlegungen auch erwartet. Die Verwendung von fünf und sieben Replikaten ergibt einen weiteren Zuwachs der Lookup-Wahrscheinlichkeit, allerdings ist dieser Zuwachs nicht mehr so deutlich, wie beim Wechsel von einem auf drei Replikaten. Dies ist damit zu begründen, dass durch den zusätzlichen Kommunikationsaufwand, der durch die Verwendung von Replikaten entsteht, die Delivery Ratio abnimmt. Abbildung 6.24 gibt den Kommunikationsaufwand der Lookup- und Insert-Operation bei Verwendung von einem, drei, fünf und sieben Replikaten an. Wie erwartet erhöht sich der Kommunikationsaufwand pro verwendetes Replikat. Die Delivery Ratio bei Verwendung von einem, drei, fünf oder sieben Replikaten ist in Abbildung 6.25 für die Parametermenge Tiny Security zu sehen. Ab einem gewissen Anteil von Angreifern, in der Abbildung bei 15%, verringert sich die Lookup-Wahrscheinlichkeit bei Verwendung von Replikation im Vergleich zur Lookup-Operation ohne Replikation. Dies ist damit begründet, dass in diesem Fall zu viele Dienstbeschreibungen von

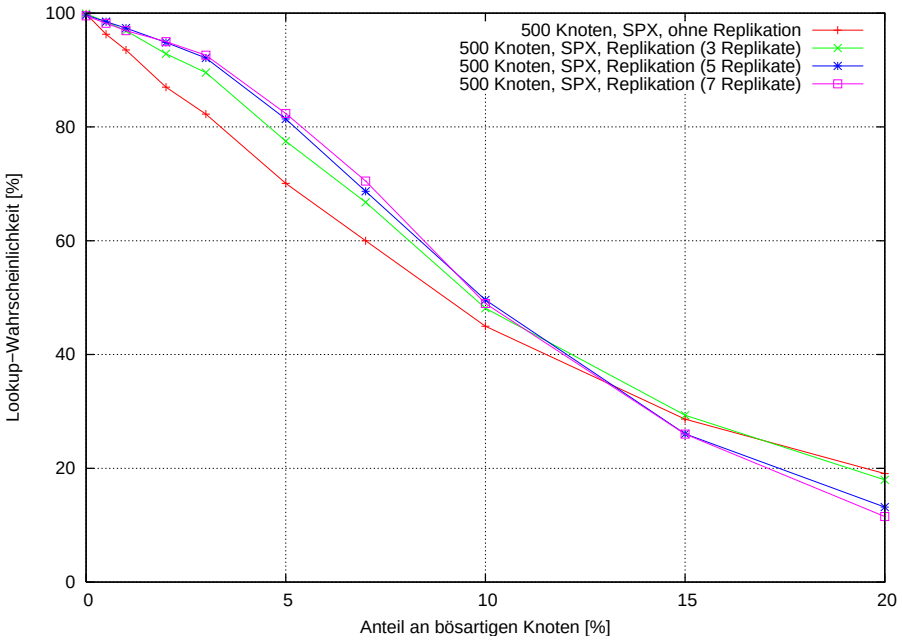


Abbildung 6.23: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX mit und ohne Replikation unter Verwendung der Parametermenge Full Security (FS)

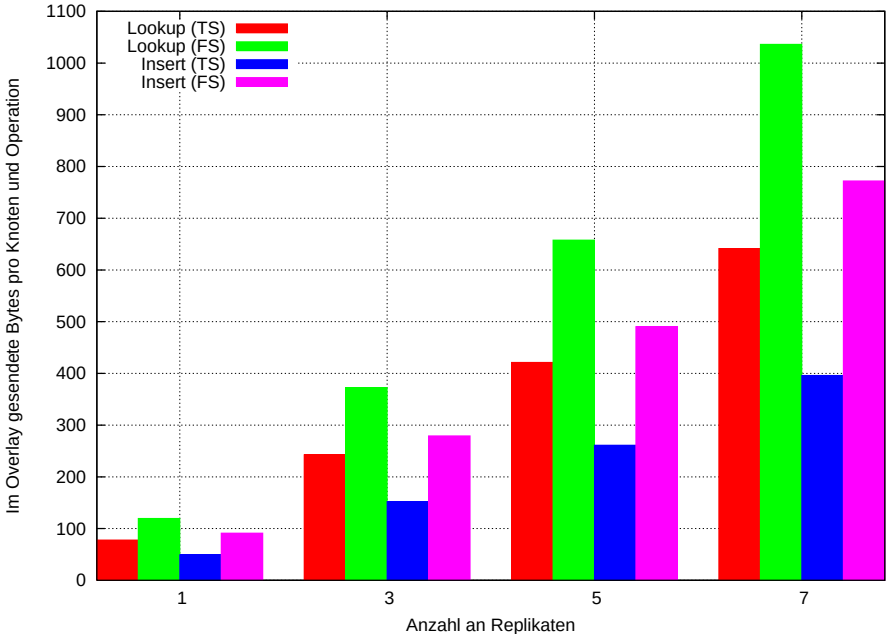


Abbildung 6.24: Durchschnittlicher Kommunikationsaufwand für die Lookup- und Insert-Operation bei Verwendung von 1, 3, 5 und 7 Replikaten für die Parametermengen Tiny Security (TS) und Full Security (FS)

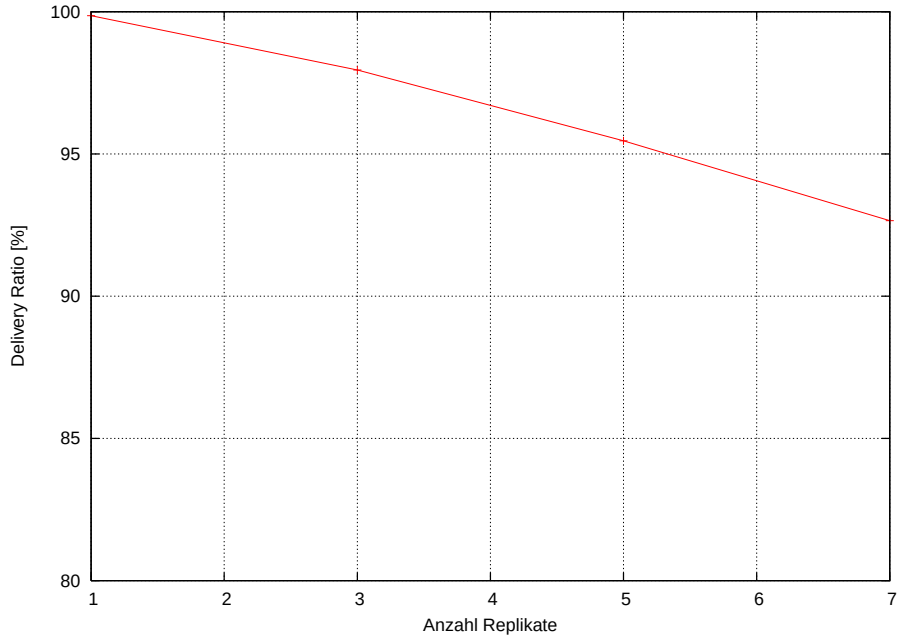


Abbildung 6.25: Delivery Ratio bei 500 Knoten und 1, 3, 5 bzw. 7 Replikaten (Parametermenge Tiny Security)

Angreifern manipuliert wurden, so dass der Mehrheitsentscheid zu Gunsten der vom zusammenarbeitenden Angreifer manipulierten Dienstbeschreibungen ausgeht.

In einem weiteren Experiment, dessen Parameter in Tabelle 6.14 dargestellt sind, wurde untersucht, wie sich die durchschnittliche Lookup-Wahrscheinlichkeit für steigende Knotenanzahl bei Verwendung von SPX mit Replikation verhält.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 1000$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	5%
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1,3,5,7
Simulationsläufe	jeweils 30

Tabelle 6.14: Parameter der Simulationen zur Untersuchung der Leistung von SPX mit Replikation bei steigender Knotenanzahl

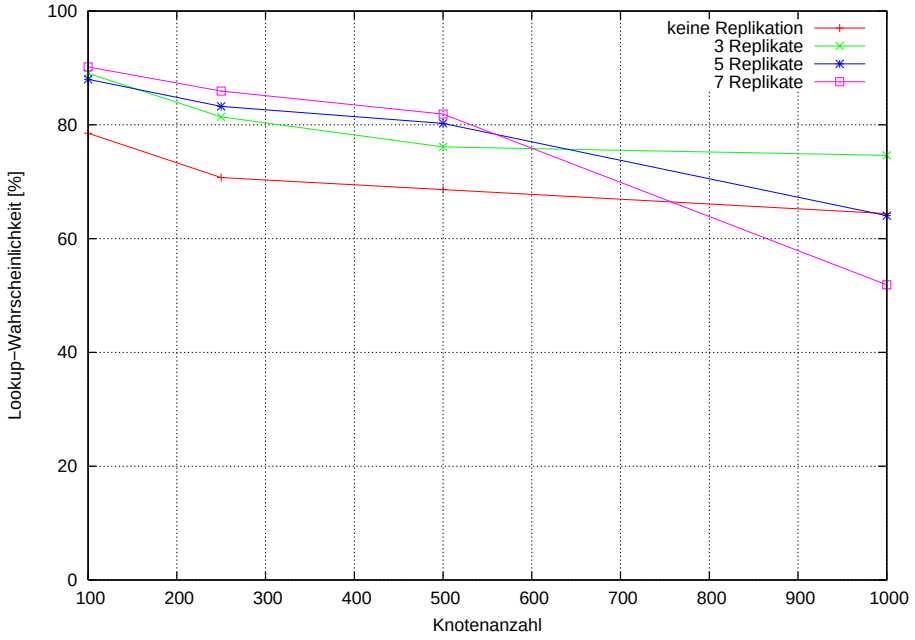


Abbildung 6.26: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von Replikaten und steigender Knotenanzahl

Ein deutliches Sinken der durchschnittlichen Lookup-Erfolgswahrscheinlichkeit zwischen 500 und 1000 Knoten ist bei der Verwendung von 5 oder 7 Replikaten zu erkennen. Dies ist bedingt durch eine Überlastung des Netzes, da die Delivery Ratio auf 78.6% (bei 500 Knoten) bzw. 67% (bei 1000 Knoten) fällt.

Zusammenfassend kann man sagen, dass SPX bei steigender Knotenanzahl gut skaliert und bei gleich bleibender Anzahl an Angreifern und steigender Knotenanzahl

die Lookup-Wahrscheinlichkeit erhöht. SPX eignet sich also für den Einsatz in Sensornetzen. Eine weitere Steigerung der Lookup-Wahrscheinlichkeit kann durch Einsatz von Replikation erreicht werden, allerdings geht die Verbesserung der Lookup-Wahrscheinlichkeit mit einem erhöhten Kommunikationsaufwand einher. SPX mit Replikation eignet sich deshalb vor allem für Szenarien, in denen wenige Lookup- und Insert-Operationen gleichzeitig ablaufen (z.B. weil wenige Dienste im Netz vorhanden sind oder weil Sensorknoten zeitlich stark versetzt Dienste anbieten oder abfragen). Replikation bietet die Möglichkeit eines Tradeoffs zwischen Sicherheit und Kommunikationsaufwand.

6.1.4.2 Multi-Path-Key-Exchange

Um die Leistungsfähigkeit von MPX zu beurteilen wurde ein Experiment mit den in Tabelle 6.15 angegebenen Parametern durchgeführt.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 1000, 1500$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer(m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS), Full Security (FS)
Schlüsselaustauschverfahren für Insert und Lookup	MPX
Gesamtanzahl Schlüsselteile (n)	4
Anzahl zur Rekonstruktion benötigte Schlüsselteile (k)	3
Anzahl Replikate	1
Simulationsläufe	jeweils 30

Tabelle 6.15: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit von MPX

Abbildung 6.27 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für steigende Knotenanzahl bei Verwendung der Parametermenge Tiny Security (TS), in Abbildung 6.27 wird entsprechend Full Security (FS) verwendet. Der Verlauf der Lookup-Wahrscheinlichkeit bei steigendem Anteil von Angreifern sowie der Lookup-Wahrscheinlichkeit bei steigender Knotenanzahl entsprecht den Erwartungen und den Ergebnissen der Untersuchungen der Lookup-Operation mittels SPX. Ein genauer Vergleich zu SPX wird in Abschnitt 6.1.4.3 vorgenommen. Auch bei der Lookup-Operation mit MPX nimmt die durchschnittliche Lookup-Wahrscheinlichkeit bei steigendem Anteil von Angreifern ab. Ebenfalls bewirkt auch bei MPX eine größere Anzahl an Knoten eine geringere Lookup-Wahrscheinlichkeit bei gleichem Anteil von Angreifern, da die durchschnittliche Länge der Overlay-Pfade wächst.

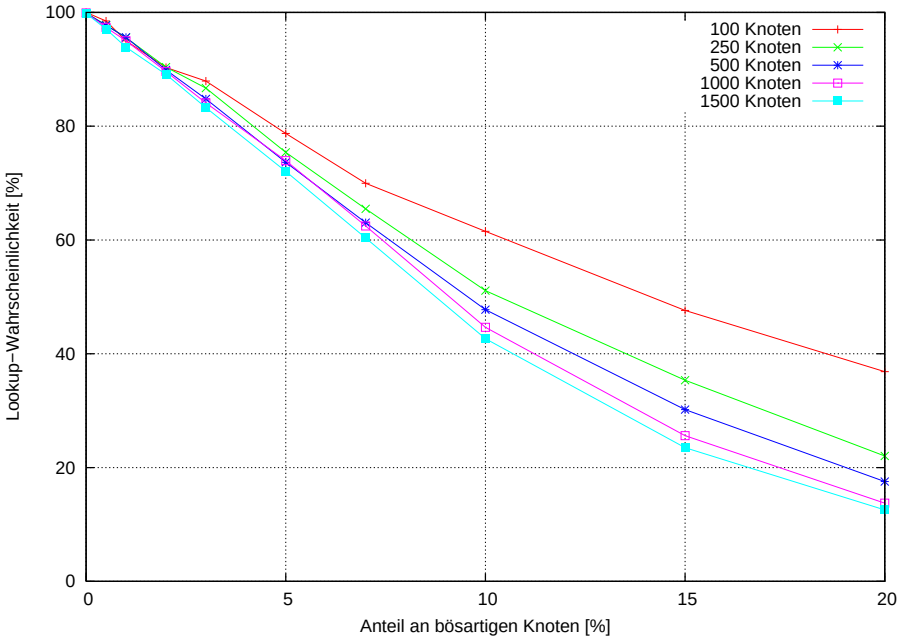


Abbildung 6.27: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von MPX mit der Parametermenge Tiny Security (TS)

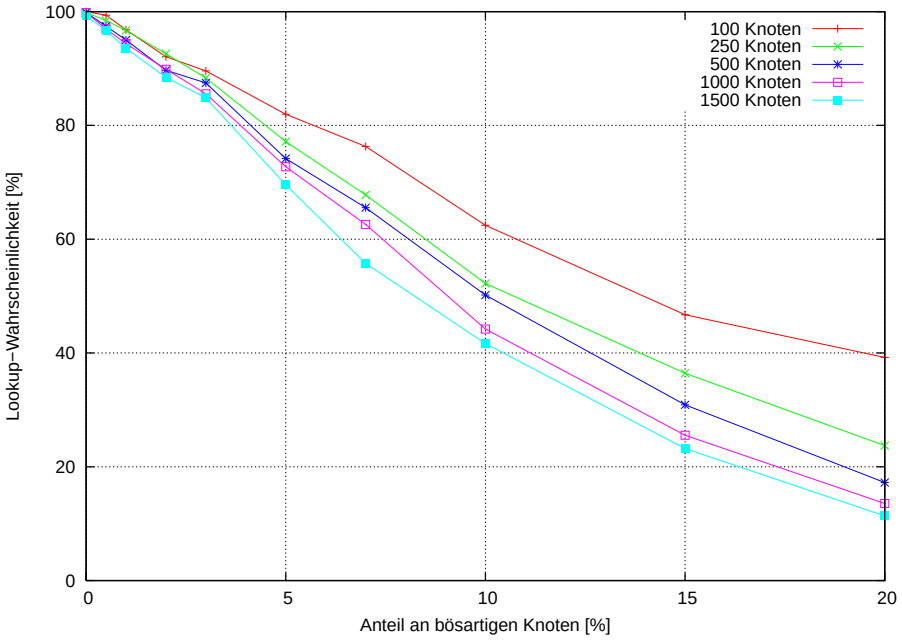


Abbildung 6.28: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von MPX mit der Parametermenge Full Security (FS)

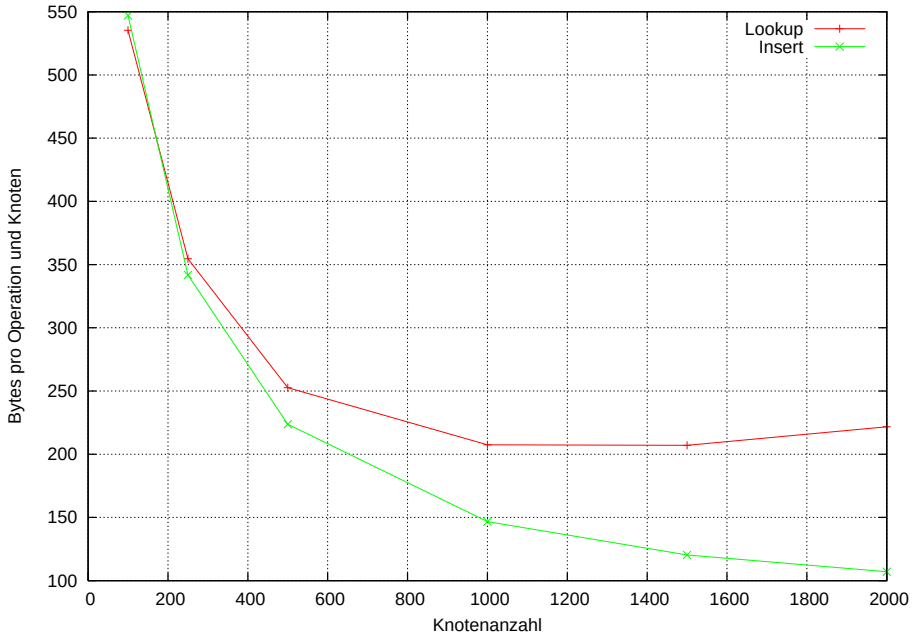


Abbildung 6.29: Durchschnittlicher Kommunikationsaufwand der Lookup- und Insert-Operation mit MPX: relative Werte

Abbildung 6.29 und Abbildung 6.30 zeigen den durchschnittlichen Kommunikationsaufwand der Lookup- und Insert-Operation bei steigender Knotenanzahl bei Verwendung von MPX. Wie schon bei der Betrachtung der Lookup-Operation mit SPX wird der Kommunikationsaufwand in Anzahl im Overlay gesendeter Bytes pro Lookup- bzw. Insert-Operation pro Knoten (in Abbildung 6.29) bzw. absolut (in Abbildung 6.30) angegeben. Auch bei MPX steigt die Anzahl an im Overlay gesendeten Bytes pro Lookup-Operation und Knoten ab einer gewissen Knotenanzahl wieder an. War dies bei der Lookup-Operation mittels SPX bereits bei 500 Knoten der Fall, so erfolgt dieser Anstieg bei der Lookup-Operation mittels MPX erst ab 1500 Knoten. Dies ist der Tatsache geschuldet, dass die wesentlich längeren Dienstelisten bei höheren Knotenanzahlen im Fall von MPX erst später eine deutliche Auswirkung haben, weil der Kommunikationsaufwand von MPX allgemein größer ist.

Genau wie beim Einsatz von SPX soll auch bei MPX untersucht werden, welchen Einfluss Replikation auf die durchschnittliche Lookup-Wahrscheinlichkeit hat. Dazu wurde ein Experiment mit den in Tabelle 6.16 angegebenen Parametern durchgeführt.

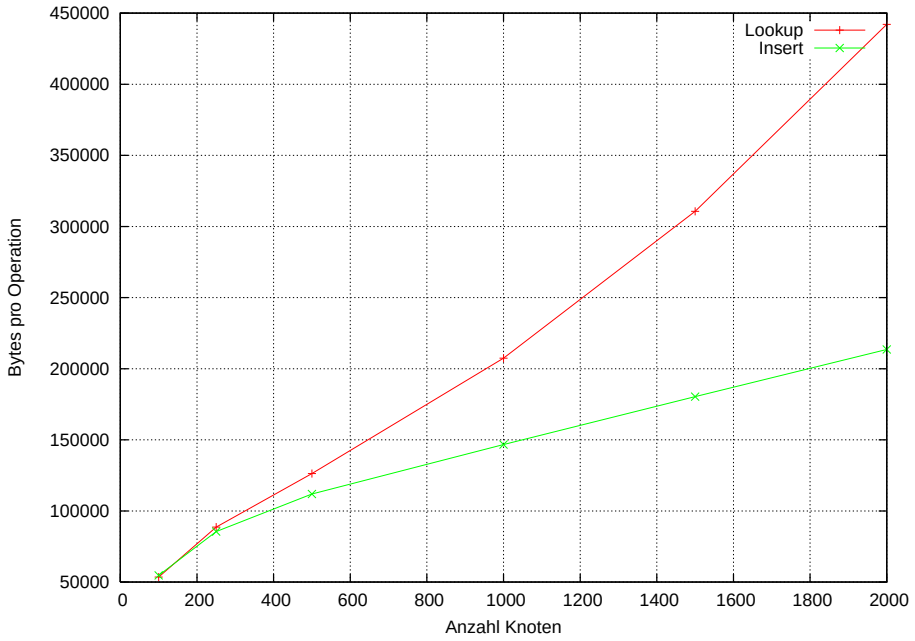


Abbildung 6.30: Durchschnittlicher Kommunikationsaufwand der Lookup- und Insert-Operation mit MPX: absolute Werte

Parameter	Einstellung
Gesamtknotenanzahl	500
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer(m)	$0\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	MPX
Gesamtanzahl Schlüsselteile (n)	4
Anzahl zur Rekonstruktion benötigte Schlüsselteile (k)	3
Anzahl Replikate	1,3,5,7
Simulationsläufe	jeweils 10

Tabelle 6.16: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit von MPX mit Replikation

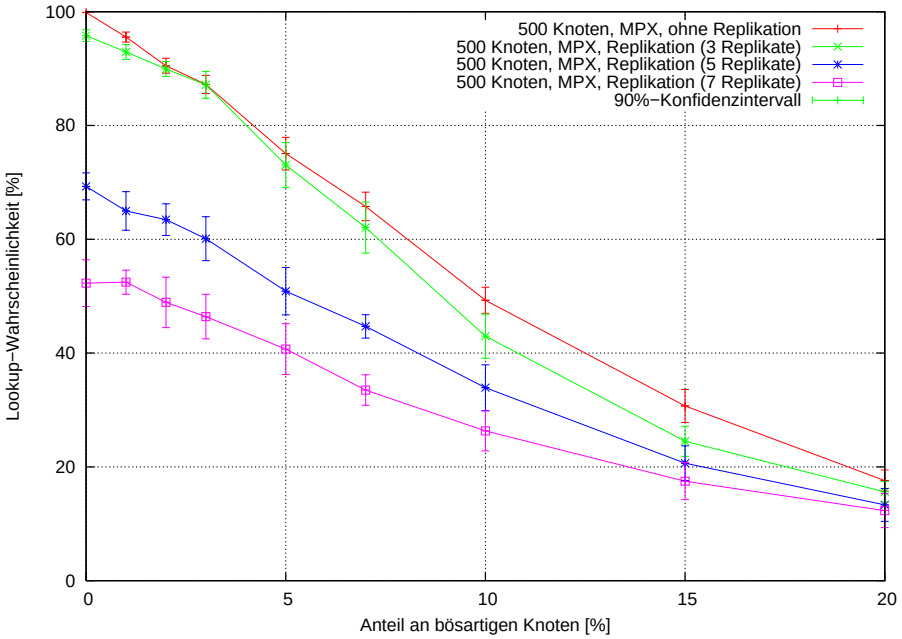


Abbildung 6.31: Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit unter Verwendung von MPX mit und ohne Replikation für ein SCAN mit 500 Knoten

Bei MPX ergibt sich, im Gegensatz zu SPX, dass durch Verwendung von Replikation keine Verbesserung der durchschnittlichen Lookup-Wahrscheinlichkeit bei 500 Knoten erfolgt, wie Abbildung 6.31 zeigt. Vielmehr führt der Einsatz von Replikation zu einer deutlichen Verschlechterung der Lookup-Wahrscheinlichkeit. Dies ist durch eine Überlastung des Netzes verursacht. Dies wird an der gesunkenen Delivery Ratio bei Verwendung von Replikation und MPX deutlich (siehe Abbildung 6.32). Da die Parametermenge Full Security im Vergleich zu Tiny Security mit einer weiteren Erhöhung des Kommunikationsaufwands einhergeht wurde Full Security mit Replikation bei MPX nicht mehr betrachtet.

In Abbildung 6.33 wird die Lookup-Wahrscheinlichkeit in einem SCAN mit 250 Knoten beim Einsatz von MPX gezeigt. Hier ergibt sich noch ein deutlicher Zugewinn durch die Verwendung von Replikation. Bei MPX kann Replikation also gewinnbringend bei geringer Knotenanzahl eingesetzt werden, da ansonsten die durchschnittliche Lookup-Wahrscheinlichkeit wegen einer Überlastsituation, erkennbar an der gesunkenen Delivery Ratio, sinkt. Dieses Ergebnis fußt auf dem gewählten Simulationsmodell.

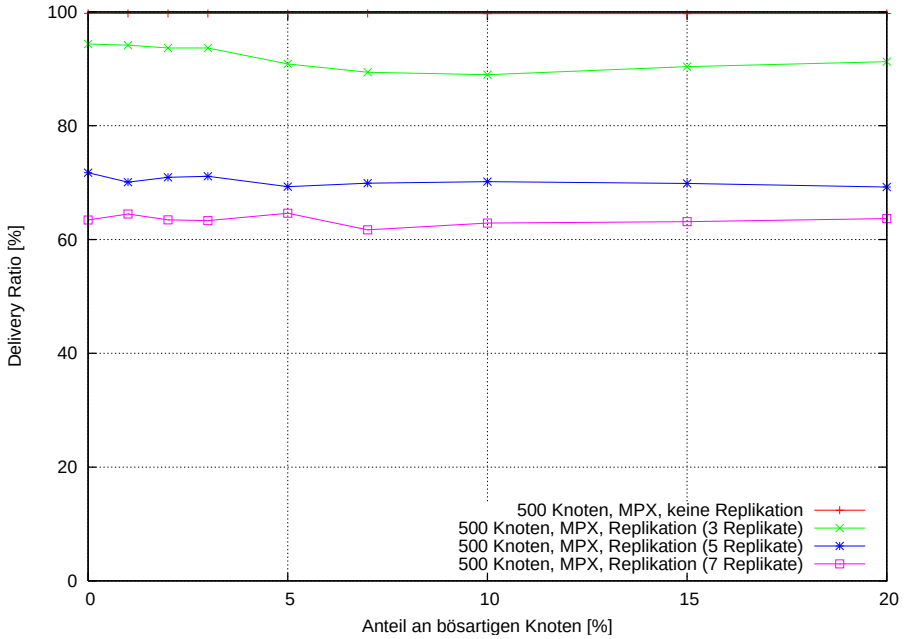


Abbildung 6.32: Delivery Ratio bei Verwendung von Replikation und MPX

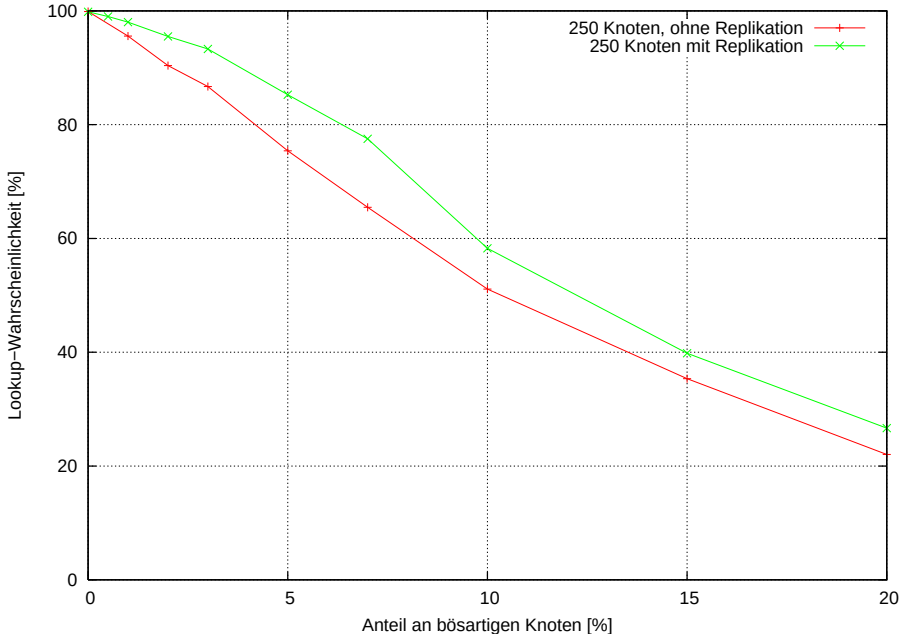


Abbildung 6.33: Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit unter Verwendung von MPX mit und ohne Replikation für ein SCAN mit 250 Knoten

Um zu untersuchen, welchen Einfluss das gewählte Simulationsmodell auf die Ergebnisse letzten Experiments haben, wurde ein Experiment durchgeführt, in jeder Knoten lediglich einen Dienst anbietet und einen Dienst nutzt. Dabei wurde untersucht, wie sich MPX bei Verwendung von 5 und 7 Replikaten in diesem Szenario verhält. Tabelle 6.17 zeigt die Parameter des Experiments.

Parameter	Einstellung
Gesamtknotenanzahl	500
Anzahl abzufragender Dienste pro Knoten	1
Anzahl angebotener Dienste pro Knoten	1
Anteil Angreifer(m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	MPX
Gesamtanzahl Schlüsselteile (n)	4
Anzahl zur Rekonstruktion benötigte Schlüsselteile (k)	3
Anzahl Replikate	5,7
Simulationsläufe	jeweils 10

Tabelle 6.17: Parameter der Simulationen zur Untersuchung des Einflusses des gewählten Simulationsmodells

Die durchschnittliche Lookup-Wahrscheinlichkeit für dieses Experiment zeigt Abbildung 6.34. Deutlich ist erkennbar, dass es nicht zu dem Einbruch der Lookup-Wahrscheinlichkeit kommt, der in Abbildung 6.33 zu sehen war, womit der Einfluss der Anzahl von Diensten gezeigt ist. Werden wenige Dienste angeboten und genutzt, so kann Replikation in Verbindung mit MPX also sehr wohl genutzt werden.

Zusammenfassend kann man sagen, dass MPX bei steigender Knotenanzahl gut skaliert. MPX eignet sich also für den Einsatz in Sensornetzen. Eine weitere Steigerung der Lookup-Wahrscheinlichkeit kann durch Einsatz von Replikation erreicht werden, allerdings geht die Verbesserung der Lookup-Wahrscheinlichkeit mit einem deutlich erhöhten Kommunikationsaufwand einher, der bei MPX so ausgeprägt ist, dass sich MPX mit Replikation vor allem für Sensornetze mit bis zu 250 Knoten eignet, in denen wenige Lookup- und Insert-Operationen gleichzeitig ablaufen (z.B. weil wenige Dienste im Netz vorhanden sind oder weil Sensorknoten zeitlich stark versetzt Dienste anbieten oder abfragen). Für größere Sensornetze ist MPX mit Replikation nur dann geeignet, wenn sehr wenige Dienste angeboten werden.

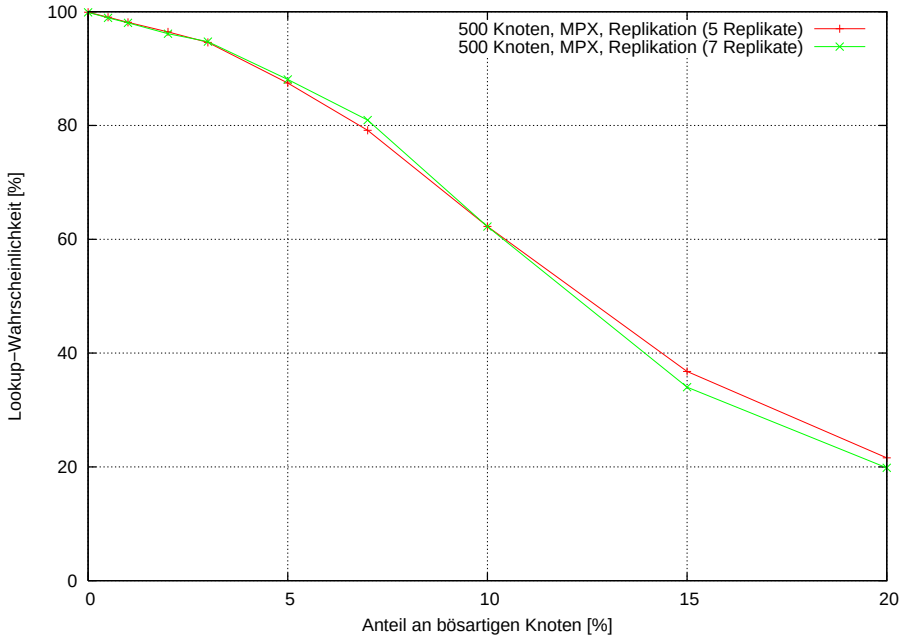


Abbildung 6.34: Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit von MPX mit Replikation (5 und 7 Replikate) für ein SCAN mit 500 Knoten, wobei jeder Knoten einen Dienst anbietet und einen Dienst nutzt.

6.1.4.3 Vergleich SPX und MPX

Um SPX, SPX mit Replikation und MPX zu vergleichen wurde ein Experiment mit 500 und 1000 Knoten durchgeführt. Tabelle 6.18 zeigt die Parameter des Experiments.

Abbildung 6.35 zeigt einen Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit bei Verwendung der Lookup-Operation mit SPX, SPX mit Replikation bzw. MPX unter steigendem Anteil an Angreifern in einem SCAN mit 500 Knoten. Dabei ist zu sehen, dass MPX bei einem moderaten Anteil von Angreifern eine Verbesserung der Lookup-Wahrscheinlichkeit von ca. 5% gegenüber SPX erreicht. SPX mit 3 oder 5 Replikaten liefert jedoch eine bessere Lookup-Wahrscheinlichkeit als MPX. Allerdings muss beim Vergleich von SPX und MPX berücksichtigt werden, dass MPX mehr Funktionalität bietet als SPX: Mit MPX ist es z. B. möglich, eine Position im SCAN zu authentifizieren, was mit SPX nicht möglich ist. Abbildung 6.36 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für 1000 Knoten. Hierbei fällt die durchschnittliche Lookup-Wahrscheinlichkeit von SPX mit 5 Replikaten deutlich unter die durchschnittliche Lookup-Wahrscheinlichkeit aller anderen Kurven, was mit dem durch die Verwendung von 5 Replikaten verursachten Kommunikationsaufwand zu erklären ist.

In Abbildung 6.37 wird der Kommunikationsaufwand der Lookup- und Insert-Operation für SPX, SPX mit Replikation (3 und 5 Replikate) sowie MPX im Experiment mit 500 Knoten gezeigt. In Abbildung 6.38 ist zum Vergleicht der Kommunikati-

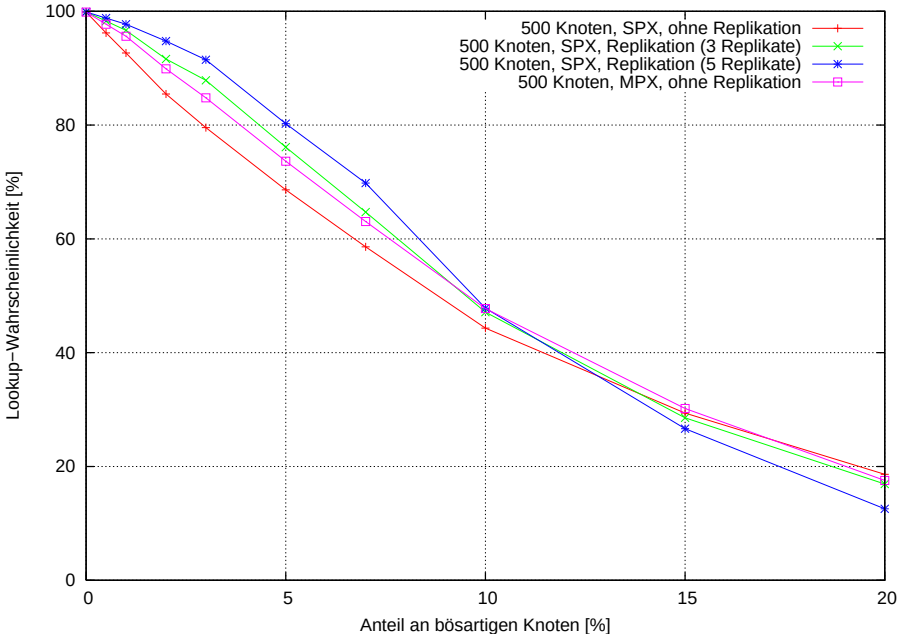


Abbildung 6.35: Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit unter Verwendung von SPX und MPX für ein SCAN mit 500 Knoten

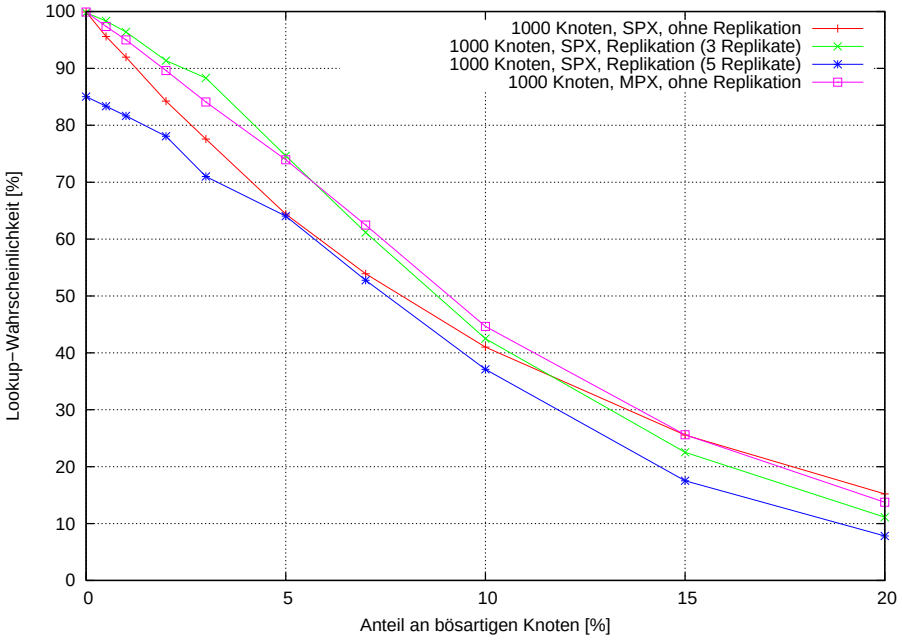


Abbildung 6.36: Vergleich der durchschnittlichen Lookup-Wahrscheinlichkeit unter Verwendung von SPX und MPX für ein SCAN mit 1000 Knoten

Parameter	Einstellung
Gesamtknotenanzahl	500,1000
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Anteil Angreifer (m)	$0, 0.5\%, 1\%, 2\%, 3\%, 5\%, 7\%, 10\%, 15\%, 20\%$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX, MPX
Anzahl Replikate	1 bei MPX; 1, 3 und 5 bei SPX
Simulationsläufe	Jeweils 30

Tabelle 6.18: Parameter der Simulationen zum Vergleich von SPX, SPX mit Replikation und MPX

onsaufwand der Lookup- und Insert-Operation für 1000 Knoten dargestellt. Da die Lookup-Wahrscheinlichkeit von MPX und SPX unter Verwendung von 3 Replikaten relativ ähnlich ist, bietet es sich an, für größere Szenarien eher MPX zu verwenden und für kleinere Szenarien (mit 500 Knoten und weniger) SPX mit 3 Replikaten.

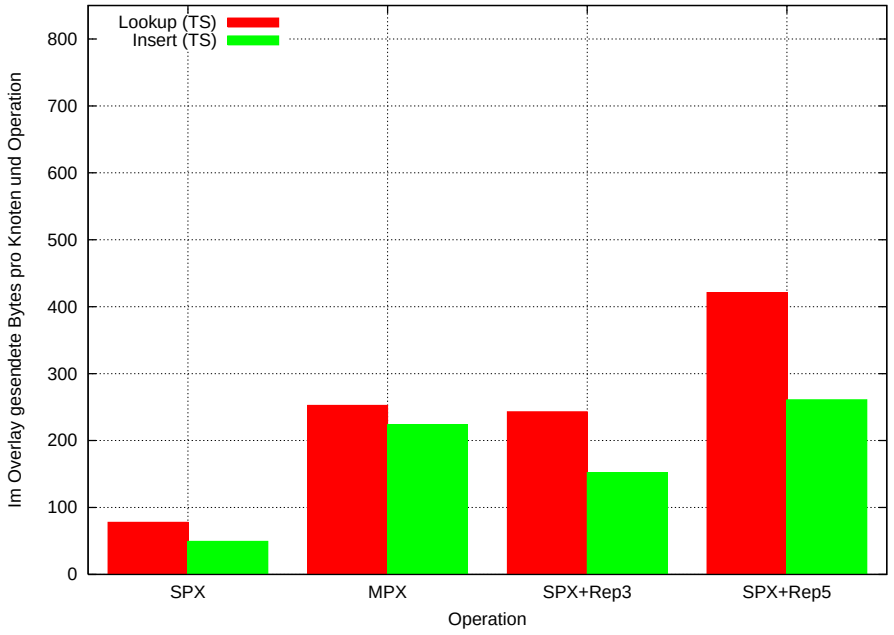


Abbildung 6.37: Vergleich des durchschnittlichen Kommunikationsaufwands für SPX, MPX, SPX mit 3 Replikaten (SPX+Rep3) und SPX mit 5 Replikaten (SPX+5) für 500 Knoten

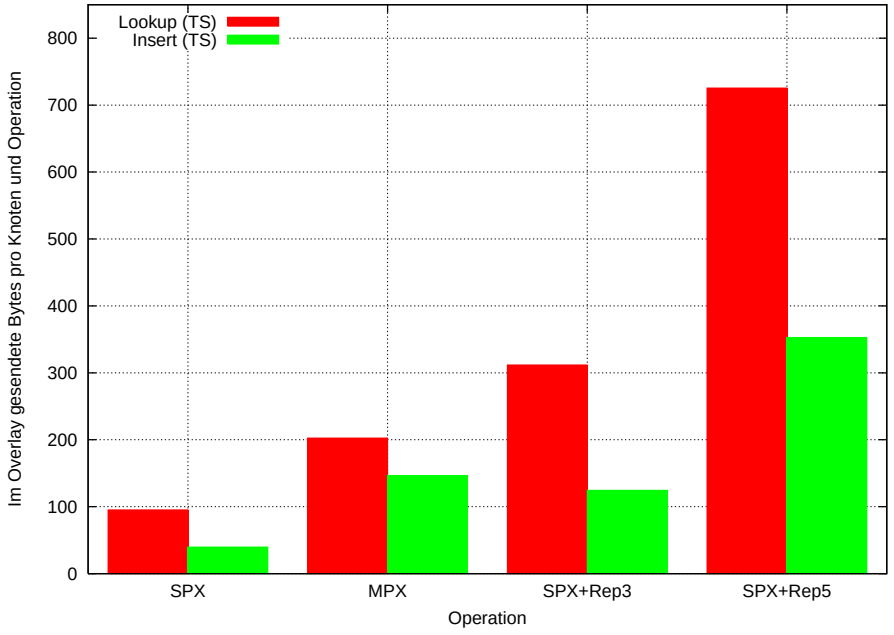


Abbildung 6.38: Vergleich des durchschnittlichen Kommunikationsaufwands für SPX, MPX, SPX mit 3 Replikaten (SPX+Rep3) und SPX mit 5 Replikaten (SPX+5) für 1000 Knoten

6.1.5 Die Join-Operation

Um den Kommunikationsaufwand der Join-Operation zu untersuchen wurde ein Experiment ausgeführt, dessen Parameter in Tabelle 6.19 zu sehen sind.

Parameter	Einstellung
Gesamtknotenanzahl	$100, 250, 500, 1000, 1500$
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS), Full Security (FS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Anzahl Replikate	1

Tabelle 6.19: Parameter der Simulationen zur Untersuchung des Kommunikationsaufwands der Join-Operation

Abbildung 6.39 zeigt den durchschnittlichen Kommunikationsaufwand der Join-Operation in im Overlay gesendeten Bytes pro Operation und Knoten für die Parametermengen Tiny Security (TS) und Full Security (FS) bei steigender Knotenanzahl. Der steigende Kommunikationsaufwand der Join-Operation geht dabei auf die steigende durchschnittliche Länge der Overlay-Routing-Pfaden zurück, die bereits in Tabelle 6.11 gezeigt wurde. Die Join-Operation erzeugt im Vergleich zur Lookup- und Insert-Operation einen größeren Kommunikationsaufwand. Allerdings wird die Join-Operation auch nur ein einziges Mal für jeden Knoten aufgerufen, weswegen der höhere Aufwand gerechtfertigt ist.

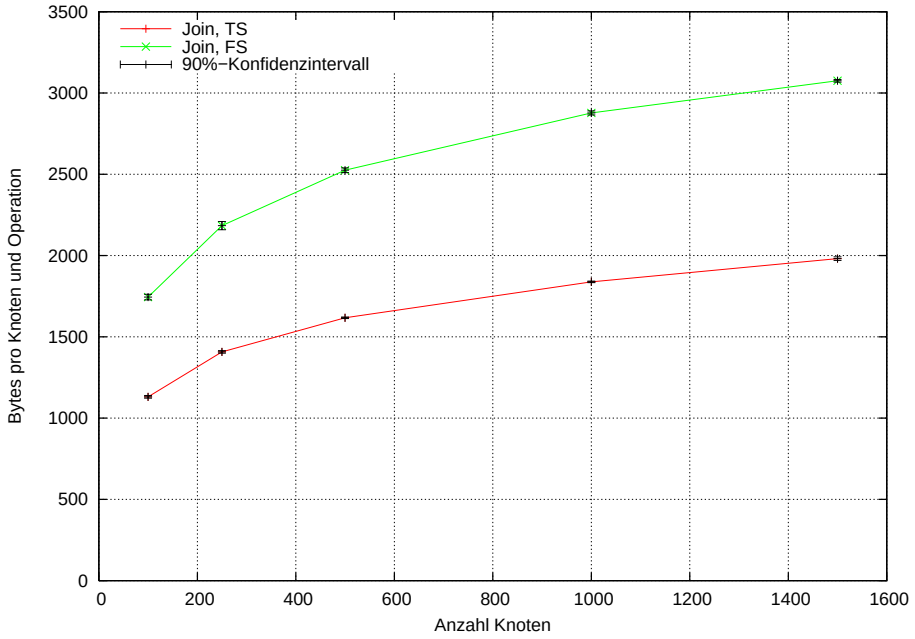


Abbildung 6.39: Vergleich des durchschnittlichen Kommunikationsaufwands der Join-Operation für die Parametermengen Tiny Security (TS) und Full Security (FS)

6.1.6 Leistungsanalyse der Behandlung von Knotenausfällen

Zur Untersuchung der Leistungsfähigkeit der expliziten Zonenübergabe wurde ein Experiment mit 500 Knoten und 0%, 1%, 3%, 5%, 7%, 10% und 15% Angreifern durchgeführt, in dem 1%, 2%, 3%, bzw. 5% der Knoten innerhalb ihrer Zeit im SCAN ausfielen. Tabelle 6.20 fasst die Parameter des Experiments noch einmal zusammen.

Abbildung 6.40 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für verschiedene Anteile von Knotenausfällen. Dabei fällt auf, dass die Lookup-Wahrscheinlichkeit bei Knotenausfall leicht höher ist. Dies hängt einerseits damit zusammen, dass durch die Zonenübernahme die durchschnittliche Overlay-Routing-Pfad-Länge sinkt, da ein Knoten nun mehrere Zonen verwaltet und dadurch ein Hop auf dem Knoten selbst, ohne jegliche Kommunikation, erfolgt. Bei 500 Knoten und 0% Knotenausfall beträgt die durchschnittliche Pfadlänge im Overlay 4.19 Overlay-Hops, während sie bei 500 Knoten und 5% Knotenausfall nur noch 3.87 Overlay-Hops beträgt. Dabei ist allerdings zu beachten, dass ein Knoten, der eine Zone übernimmt, stärker belastet ist, da er z. B. mehr Dienstbeschreibungen speichern muss, d. h. die leichte Verbesserung der Lookup-Wahrscheinlichkeit geht mit einem erhöhten Aufwand für die Sensorknoten einher. In Abbildung Abbildung 6.40 wird außerdem sichtbar, dass sich die Lookup-Wahrscheinlichkeit bei verschiedenen Anteilen an Angreifern nicht wesentlich ändert. Dies war zu erwarten, da Angreifer lediglich dann einen Einfluss auf das Verfahren zur expliziten Zoneübergabe haben, wenn eine Zone an einen bereits feindlichen Knoten übergeben wird.

Parameter	Einstellung
Gesamtknotenanzahl	500
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Verfahren zur Behandlung von Knotenausfällen	Explizite Zonenübergabe
Anteil ausfallender Knoten (f)	0%,1%,2%,3%,5%
Anteil Angreifer (m)	0%,1%,3%,5%,7%,10% und 15%
Simulationsläufe	jeweils 10

Tabelle 6.20: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit der expliziten Zonenübergabe

Im Experiment wurde auch der Kommunikationsaufwand der expliziten Zonenübergabe gemessen. Der Kommunikationsaufwand, gemessen in im Overlay gesendeten Bytes pro Zonenübergabe beträgt im Experiment durchschnittlich 574 Bytes. Auch dies spricht für die Effizienz des Verfahrens. Es bietet sich also auf jeden Fall an, ein vorausschauendes Energiemanagement (wie z. B. [33]) mit explizite Zonenübergabe im Rahmen von SCAN einzusetzen.

In einem weiteren Experiment mit 1000 Knoten wurde die Leistungsfähigkeit von RMPX untersucht. Die Parameter des Experiments sind in Tabelle 6.21 aufgeführt.

Parameter	Einstellung
Gesamtknotenanzahl	1000
Anzahl abzufragender Dienste pro Knoten	$\in \{1, ..., 10\}, \in \mathbb{N}$
Anzahl angebotener Dienste pro Knoten	$\in \{1, ..., 7\}, \in \mathbb{N}$
Parametermenge für Sicherheitsverfahren	Tiny Security (TS)
Schlüsselaustauschverfahren für Insert und Lookup	SPX
Verfahren zur Behandlung von Knotenausfällen	RMPX
Anteil ausfallender Knoten (f)	0%,1%,2%,3%,5%
Anteil Angreifer (m)	0%,1%,3%,5%,7% und 10%
Simulationsläufe	jeweils 10

Tabelle 6.21: Parameter der Simulationen zur Untersuchung der Leistungsfähigkeit der expliziten Zonenübergabe

Im Experiment wurde ein zusammenarbeitender Angreifer folgendermaßen simuliert: Gelingt es einem Angreifer, einen der Schlüsselaustauschvorgänge zu korrumpieren, oder ist ein Angreifer Koordinator, so wird die ausgefallene Zone als feindliche Zone markiert. Alle Overlay-Nachrichten, die über diese Zone weitergeleitet werden, werden als korrumpiert markiert. Abbildung 6.41 zeigt die durchschnittliche Lookup-Wahrscheinlichkeit für einen steigenden Anteil von Angreifern. Ebenso

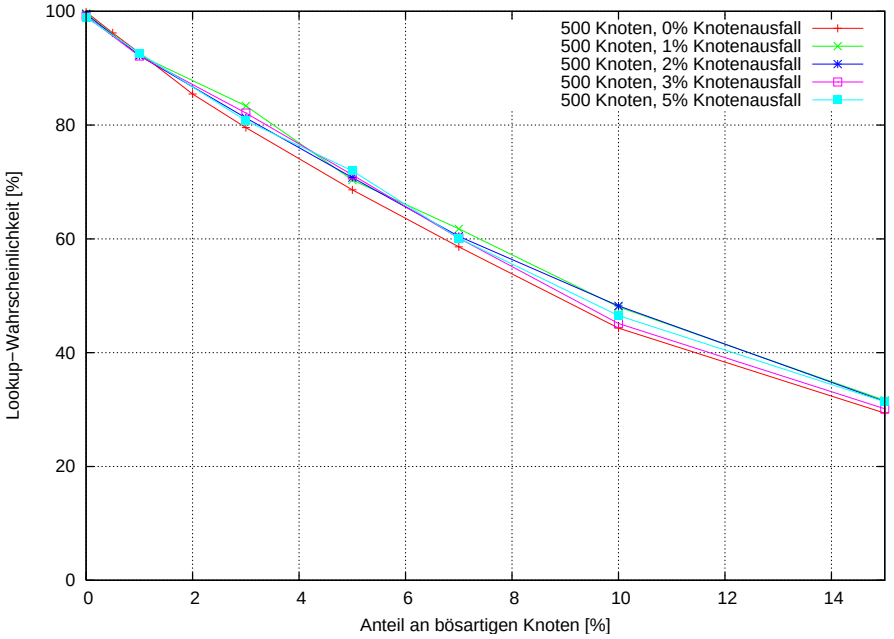


Abbildung 6.40: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX mit expliziter Zonenübergabe

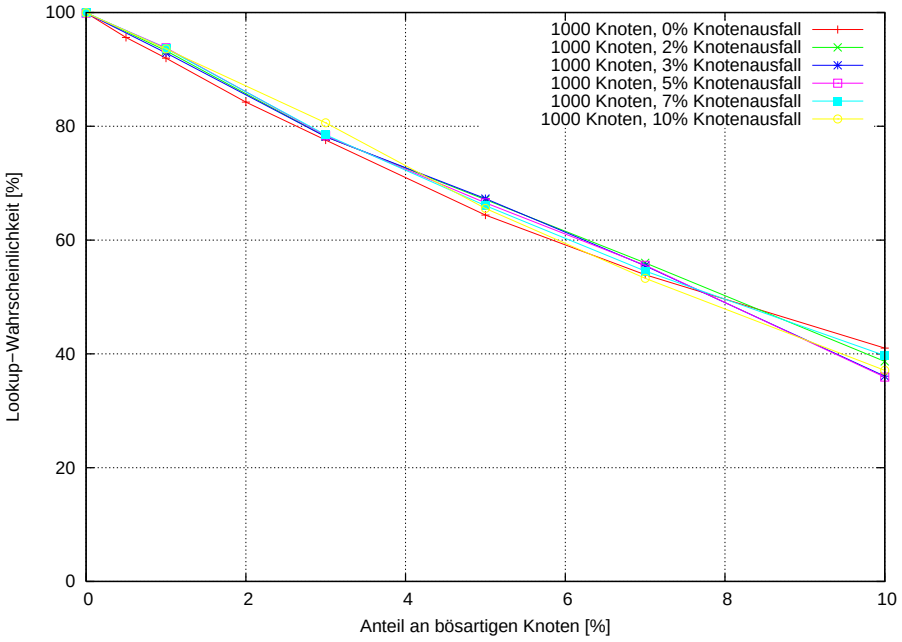


Abbildung 6.41: Durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von RMPX zur Behandlung eines Zonenausfalls

wie bei der expliziten Zonenübergabe resultiert die Übernahme von Zonen in einer Verkürzung der durchschnittlichen Overlay-Pfadlänge bedingt dadurch, dass ein Knoten jetzt mehrere Zonen verwalten kann. Dadurch steigt die durchschnittliche Lookup-Wahrscheinlichkeit leicht an. Allerdings ist bei der Verwendung von RMPX deutlich der Einfluss von Angreifern zu sehen: Ab 8% Angreifern gelingt es den zusammenarbeitenden Angreifern zunehmen, Zonen während einer Zonenreparatur zu übernehmen, indem sie den Schlüsselaustausch infiltrieren, weswegen die durchschnittliche Lookup-Wahrscheinlichkeit bei Knotenausfällen unter die durchschnittliche Lookup-Wahrscheinlichkeit ohne Knotenausfälle sinkt. Auch der Kommunikationsaufwand wurde im Rahmen des Experiments ermittelt: Eine Zonenreparatur mit RMPX verursacht im Durschnitt einen Kommunikationsaufwand von 4542 im Overlay versendet Bytes pro Zonenreparatur.

6.2 Implementierung und Leistungsanalyse auf MICAz-Sensorknoten

SCAN wurde im Rahmen dieser Arbeit nicht nur für den Netzwerksimulator GloMoSim implementiert, sonder prototypisch auf MICAz-Sensorknoten [104] realisiert [47], um die Machbarkeit von SCAN insbesondere im Bezug auf den für Sensorknoten kritischen Speicherverbrauch zu zeigen und zu Aussagen über zeitliche Abläufe zu kommen, die anhand einer Implementierung im Simulator nicht bzw. nicht aussagekräftig getroffen werden können. Besonders interessant sind die Dauer der Join-Operation, da diese eine Benutzerinteraktion erfordert, sowie die Dauer der Lookup-Operation, da Sensorknoten diese unter Umständen aufrufen, um schnell Ersatz für ausgefallene Dienste zu finden. Beide Operationen sollten also in möglichst kurzer Zeit ausgeführt werden. Die Insert-Operation ist dagegen weniger zeitkritisch, weswegen sie im Weiteren nicht betrachtet wird.

Die Implementierung des Prototypen basiert auf dem Sensornetz-Betriebssystem TinyOS 1.1 [2]. Für die Implementierung wurde im Rahmen eines Demonstrators das Szenario einer Gärtnerei ausgewählt, für das eine verteilte Sensornetz-Anwendung zur Pflege von Pflanzen entwickelt wurde. Die verteilte Sensornetz-Anwendung basiert dabei auf der ebenfalls am Institut für Telematik entwickelten dienstorientierten Programmierabstraktion „Talassa“ [49], die es ermöglicht, verteilte Anwendungen aus Diensten aufzubauen. Abschnitt 6.2.2 stellt die Randbedingungen der Prototyp-Implementierung vor, die vor allem durch die Beschränkung der verwendeten Sensorknoten sowie durch TinyOS 1.1 induziert sind. Die folgenden Abschnitte beschreiben die Implementierung auf Sensorknoten (Abschnitt 6.2.3) sowie die Realisierung des Master-Device (Abschnitt 6.2.4) und einer Visualisierungskomponente (Abschnitt 6.2.5). Daran schließt in Abschnitt 6.2.6 die Evaluierung der Prototyp-Implementierung, insbesondere die Untersuchung des Speicherbedarfs, der Dauer der Join- sowie der Dauer der Lookup-Operation, an.

6.2.1 Szenario

Als Szenario wurde eine „intelligente Gärtnerei“ betrachtet. Den schematischen Aufbau des Szenario zeigt Abbildung 6.42. Ziel der dienstorientierten Sensornetz-An-

wendung für die intelligenten Gärtnerei ist es, die Pflege der Pflanzen in einem Gewächshaus zu organisieren. Dabei soll jede Pflanze mit einem Sensorknoten versehen sein. Dieser Sensorknoten ermittelt die Feuchtigkeit der Pflanzenerde und die aktuelle Helligkeit. Weiterhin kann jeder Sensorknoten eine Pumpe ansteuern, um eine Pflanze mit Wasser zu versorgen. Darüber hinaus verfügt jeder Sensorknoten über eine Pflegeanweisung, in welcher die Umgebungsbedingungen angeben sind, unter welchen die Pflanze gedeiht. Im Prototyp wird z. B. eine untere und eine obere Schranke für die Helligkeit angegeben, welche die Pflanzen für ein optimales Wachstum benötigt. Weitere Sensorknoten sind zur Lichtsteuerung in das Gewächshaus eingebaut. Diese steuern mittels Lampen die Helligkeit im Gewächshaus so, dass die obere und untere Schranke für die Helligkeit eingehalten sind. Eine „Sonne“, im Prototyp eine Schreibtischlampe, simuliert äußere Einflüsse, die eine automatische Anpassung der Helligkeit notwendig machen.

Im Rahmen der Anwendung „intelligente Gärtnerei“ wurden folgende Interaktionen der Sensornetz-Anwendung gezeigt:

- *Feuchtigkeitsregelung:* Ein Sensorknoten steuert eine Pumpe an, sobald die Pflanzenerde zu trocken wird

- *Lichtsteuerung:* Im Gewächshaus existiert eine obere und eine untere globale Schranke für die Helligkeit. Basierend auf der aktuellen Helligkeit wird das Licht im Gewächshaus unter Berücksichtigung der unteren und oberen globalen Schranke gesteuert. Kommt eine neue Pflanze ins Gewächshaus, so werden die untere und obere globale Schranke entsprechend den Pflegeanweisungen der Pflanze angepasst. Sollte jedoch festgestellt werden, dass eine Pflanze nicht kompatibel mit den aktuellen Schranken ist, so wird sie nicht ins Sensornetz integriert.

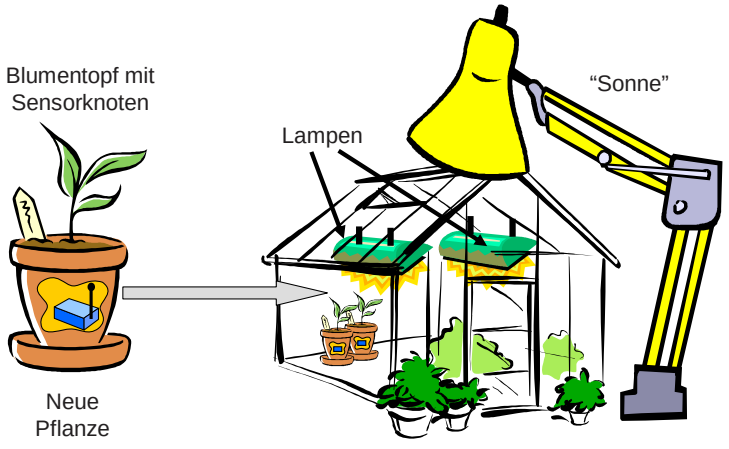


Abbildung 6.42: Schematische Darstellung des Szenario „intelligente Gärtnerei“

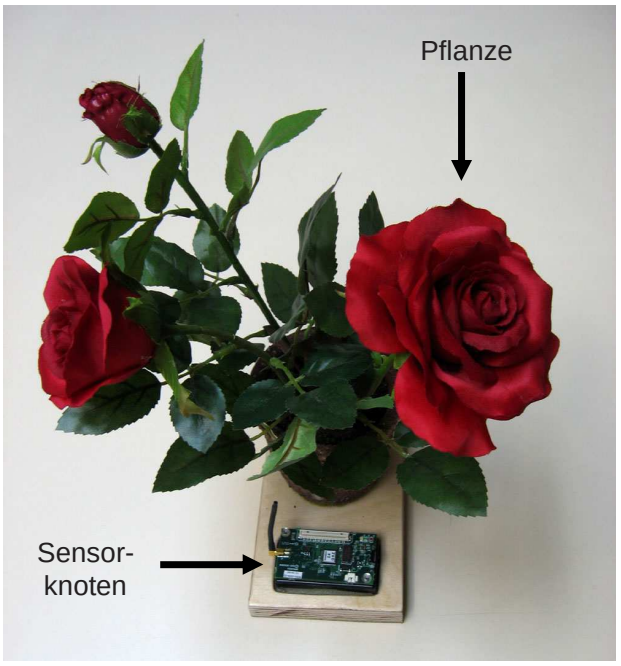


Abbildung 6.43: MICAz Sensorknoten mit Pflanze

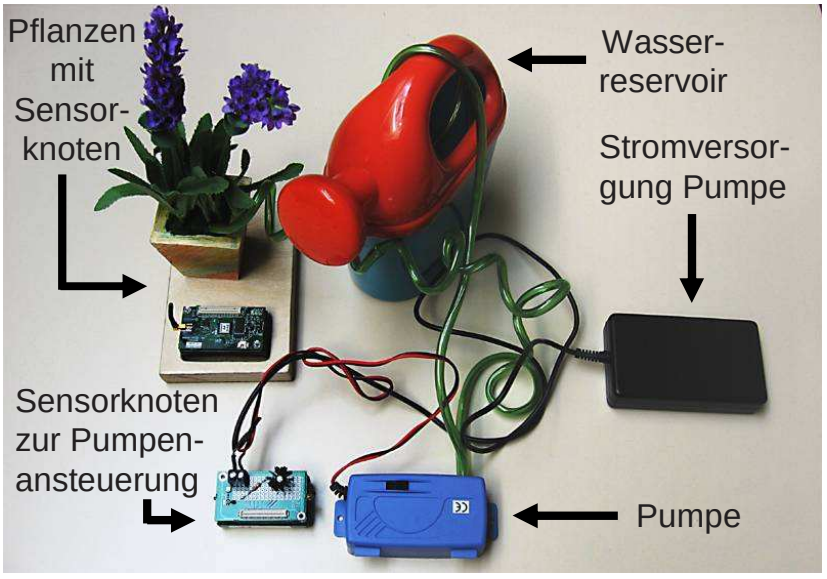


Abbildung 6.44: Im Demonstrator eingesetzte Wasserpumpe

Abbildung 6.43, Abbildung 6.44 und Abbildung 6.45 zeigen die Realisierung des Prototyps: Abbildung 6.43 zeigt einen MICAz Sensorknoten, wie er im Rahmen des Prototyps verwendet wurde: der Sensorknoten ist an einer Pflanze befestigt. Abbildung 6.44 zeigt eine Pumpe, die zur Bewässerung der ebenfalls dargestellten Pflanze verwendet wird. Die Pumpe wird von einem Sensorknoten angesteuert, der

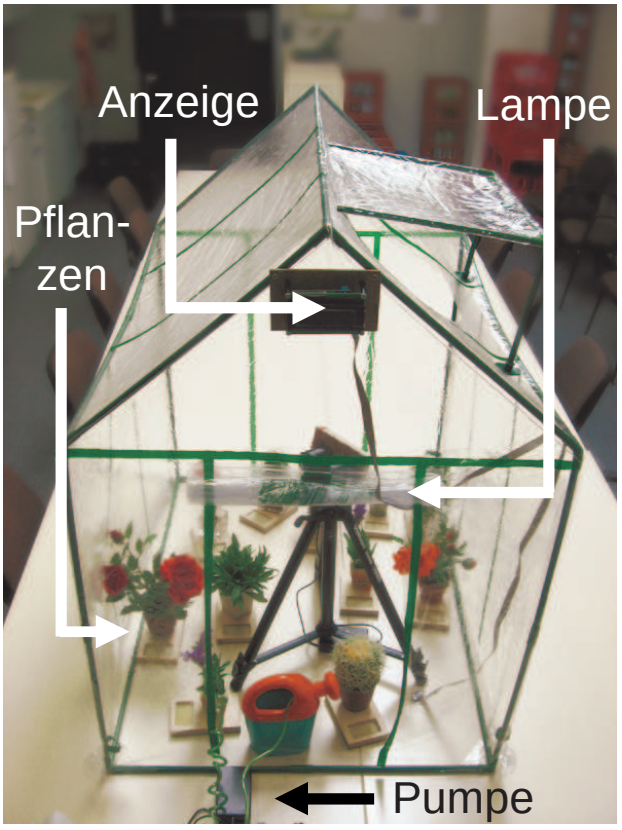


Abbildung 6.45: Gewächshaus mit Pflanzen und Pumpe

über ein Prototyp-Board die Stromzufuhr der Pumpe regelt und die Pumpe somit an- und abschalten kann. Die Pumpe bezieht ihr Wasser aus einem Wasserreservoire (im Bild eine Gießkanne). Abbildung 6.45 zeigt den Aufbau als Ganzes: 9 Pflanzen sind bereits in das Gewächshaus integriert. Am Giebel des Gewächshauses ist zu Visualisierungszwecken eine Anzeige angebracht, welche die aktuell gültige obere und untere Helligkeitsschranke zeigt. Zur Helligkeitsbeeinflussung dient eine Lampe, die je nach gemessenem Helligkeitswert und oberer und unterer Helligkeitsschranke an- oder abgeschaltet werden kann.

6.2.2 Randbedingungen

Im Prototypen kommen als Sensorknoten MICAz Motes [104] zum Einsatz. Diese verfügen lediglich über sehr geringe Ressourcen und sind nicht sehr leistungsfähig:

- 4 Kilobyte RAM (Hauptspeicher)
- 128 Kilobyte Flash-Speicher (Programmspeicher)
- 8 Bit RISC-CPU mit 8 MHz getaktet
- Bandbreite von 250 kbit/s (IEEE 802.15.4/Zigbee)

Dabei ist zu beachten, dass SCAN sich diese Ressourcen mit der Implementierung von Talassa teilt und ebenfalls Speicher vorgesehen werden muss, um die Dienstbeschreibungen als solche zu speichern. Auf den Sensorknoten kommt das Sensornetz-Betriebssystem TinyOS 1.1 [2] zum Einsatz. Dadurch ergaben sich folgende Einschränkungen:

- Die Nutzdatenmenge ist auf 29 Bytes pro versendeter Nachricht beschränkt. Es ist zwar möglich die Nachrichtenlänge und damit auch die Nutzdatenmenge pro Paket zu erhöhen, allerdings wurde in der Implementierung darauf verzichtet, da viele für TinyOS verfügbaren Module von der voreingestellten Nachrichtenlänge ausgehen.

- Die dynamische Speicherverwaltung von TinyOS 1.1 stellte sich bei der Implementierung als fehlerbehaftet heraus, da sich verschiedene Module gegenseitig beeinflussen konnten, so dass nicht zu verhindern ist, dass Speicherbereiche von zwei verschiedenen Modulen reserviert und gegenseitig überschrieben werden. Für die Implementierung von SCAN wurde deshalb auf dynamische Speicherreservierung verzichtet und der benötigte Speicher wurde statisch mittels Arrays reserviert.

6.2.3 Implementierung des Prototypen in TinyOS auf den Sensorknoten

Bei der Prototyp-Implementierung musste die Zahl der in SCAN verwendeten Dimensionen festgelegt werden. Die Dimension d des SCAN ist ein wichtiger Design-Parameter, der unter anderem die Leistungsfähigkeit und den Speicherverbrauch einer Implementierung von SCAN bestimmt. Der Speicherverbrauch wird vor allem durch die Nachbarliste erzeugt, deren Größe von der Dimension des SCAN abhängt. Da für den Prototyp eine Gesamtknotenanzahl bis zu 100 Knoten angepeilt war, ist $d = 3$ völlig ausreichend. Eine höhere Dimension stellt prinzipiell kein Problem dar, war aber für die Implementierung der hier vorgestellten Szenario nicht notwendig. So würde die Wahl von $d = 7$ den Speicherverbrauch lediglich um 180 Byte erhöhen.

6.2.3.1 Dienstbeschreibungen

Im vorliegenden Szenario haben Dienstbeschreibungen lediglich das verpflichtende Attribut „Adresse“, das die Vermittlungsschichtadresse des Dienstanbieters beinhaltet. Weitere Attribute werden von der Talassa-Implementierung nicht genutzt. Vereinfacht können die Dienstbeschreibung also als Liste von Vermittlungsschichtadressen gespeichert werden. Diesen Weg geht die prototypische Implementierung, um Speicher zu sparen. Dabei hat eine Dienstbeschreibung folgenden Aufbau (C-Datenstruktur):

```
typedef struct{
   uint8_t key[KEY_SIZE];
   uint8_t value[VALUE_SIZE];
}
```

Dabei ist `key` der Hash-Wert des Dienstnamens und `value` die Vermittlungsschichtadresse des Dienstanbieters. In der Prototyp-Implementierung wurde `KEY_SIZE=3` sowie `VALUE_SIZE=2` verwendet. Die Wahl von `KEY_SIZE` ist dadurch motiviert, dass im verwendeten 3-dimensionalen SCAN jeweils ein Byte pro Dimension für Koordinaten verwendet wird. Die Wahl von `VALUE_SIZE` ist dadurch bedingt, dass Vermittlungsschichtadressen in TinyOS 1.1 zwei Byte lang sind.

6.2.3.2 Aufbau des Prototypen

In diesem Abschnitt wird die Implementierung des Prototypen in TinyOS beschrieben. Für TinyOS ist ein modulbasierter Aufbau von Programmen üblich. Module bieten Interfaces an, die dazu verwendet werden, mehrere Module zu verbinden. Ein Interface definiert Kommandos (gekennzeichnet durch das Schlüsselwort `command`) und Events (gekennzeichnet durch das Schlüsselwort `event`), wobei das das Interface anbietende Modul lediglich die Kommandos implementiert. Die Events werden von dem Modul implementiert, welches das Interface benutzt. Der modulbasierte Aufbau von TinyOS, sowie die im Folgenden verwendeten Notationen, werden ausführlich in [2] beschrieben.

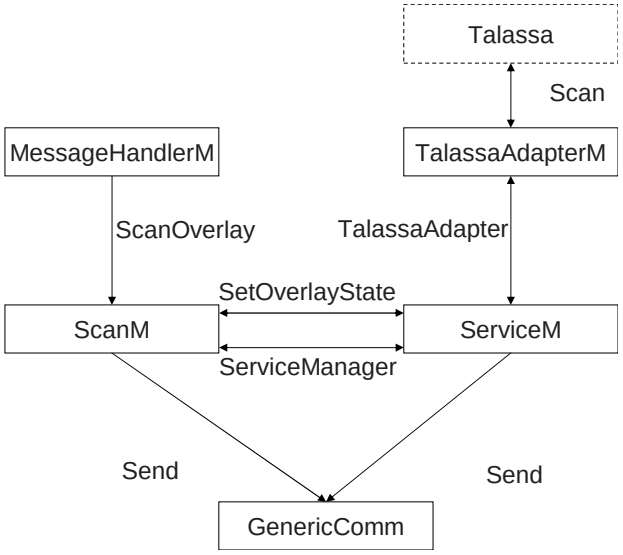


Abbildung 6.46: Aufbau des Prototypen aus TinyOS-Modulen

Abbildung 6.46 zeigt den Aufbau des Prototyps aus TinyOS-Modulen. Im Folgenden werden die Module `MessageHandlerM`, `ScanM`, `ServiceM` sowie `TalassaAdapterM` beschrieben. Das Modul `GenericComm` gehört zu TinyOS 1.1 und wird in [2] beschrieben. `GenericComm` bietet anderen Modulen die Möglichkeit, Nachrichten über

das TinyOS 1.1 eigene Routing zu versenden. Dazu bietet `GenericComm` das Interface `SendMsg`, das sowohl vom Modul `ServiceM` als auch vom Modul `ScanM` benutzt wird.

Auch die zu Talassa gehörigen Module (im Bild als gestrichelter Kasten zusammengefasst) werden in dieser Arbeit nicht dargestellt. Details zu Talassa können in [49] nachgelesen werden.

- Das Modul `MessageHandlerM` dient zur Behandlung eingehender Nachrichten. Dazu implementiert das Modul das Interface `ReceiveMsg`, welches mit dem `GenericComm` Modul von TinyOS verbunden wird:

  ```
  MessageHandlerM.ReceiveMsg -> Comm.ReceiveMsg[AM_SCAN];
  ```

  Dieses so genannte „Wireing“ dient in TinyOS dazu, Module miteinander zu verbinden. Dabei ist `AM_SCAN` der Nachrichten-Typ, der durch das Modul `MessageHandlerM` bearbeitet werden soll. Als Typ lassen sich `DIRECT_PACKET` oder `OVERLAY_PACKET` wählen. `DIRECT_PACKET` dient dabei für SCAN-Nachrichten, die auf Vermittlungsschicht versendet werden, während `OVERLAY_PACKET` Nachrichten kennzeichnet, die über das Overlay-Routing versendet werden. Das Modul `MessageHandlerM` bietet das Interface `ScanOverlay` an:

  ```
  interface ScanOverlay{
    event result_t handleOverlayPkt(TOS_MsgPtr m);
    event result_t handleDirectPkt(TOS_MsgPtr m);
    event result_t mdHelloRcvd();
    event result_t assignJPRvcd(scanPoint *point);
    event result_t zoneSplitRcvd
                   (scanPoint *point,scanZone *zone,uint8_t neighbor);
    event result_t zoneSplitNRcvd
                   (uint16_t addr,scanZone *zone,uint8_t neighborLeft);
    event result_t joinReqRcvd(uint16_t addr);
    event result_t updateRcvd(TOS\_MsgPtr m);
  }
  ```

  Empfängt das Modul `MessageHandlerM` eine SCAN-Nachricht, so löst es einen der folgenden Events aus:

  ```
  event result_t handleOverlayPkt(TOS_MsgPtr m);
  event result_t handleDirectPkt(TOS_MsgPtr m);
  ```

  Entscheidend ist lediglich, ob es sich um eine Nachricht handelt, die im Overlay versendet wird (`OVERLAY_PACKET`) oder um eine Nachricht, die von SCAN

auf Vermittlungsschicht versendet wird (DIRECT_PACKET). Während bei Aufruf des Events handleDirectPkt klar ist, dass die weitere Verarbeitung durch den Knoten erfolgt, der diese Nachricht erhalten hat, hängt beim Event handleOverlayPkt die weitere Verarbeitung davon ab, ob die Nachricht an einen Punkt in der Zone des verarbeitenden Knoten adressiert ist oder nicht. Die folgenden Events dienen der Verarbeitung einzelner SCAN-Nachrichten, die für den jeweiligen Knoten bestimmt sind.

```
event result_t mdHelloRcvd();
event result_t assignJPRvcd(scanPoint *point);
event result_t zoneSplitRcvd(scanPoint *point,scanZone
 *zone,uint8_t neighbor);
event result_t zoneSplitNRcvd(uint16_t addr,scanZone *zone,
               uint8_t neighborLeft);}
event result_t joinReqRcvd(uint16_t addr);
event result_t updateRcvd(TOS_MsgPtr m);
```

Dabei dient mdHelloRcvd lediglich zur Erkennung des Master Device, das sich im Prototyp dadurch ankündigt, dass es regelmäßig MD_HELLO Nachrichten im Location Limited Channel versendet. Die restlichen Events dienen der Verarbeitung der Nachrichten ASSIGN_JP (assignJPRvcd), ZONE_SPLIT (zoneSplitRcvd und zoneSplitNRcvd), JOIN_REQ (joinReqRcvd) und UPDATE (updateRcvd).

- Das Modul ScanM dient der Verwaltung der Nachbarlisten und Nachbarschlüssellisten. Weiterhin erledigt ScanM das Routing von Overlay-Paketen. Um diese Aufgaben zu erfüllen benutzt ScanM das Interface ScanOverlay, welches vom Modul MessageHandlerM angeboten wird. Das Modul ScanM implementiert also alle im Interface ScanOverlay definierten Events. Dadurch erhält ScanM vom Modul MessageHandlerM alle Nachrichten und kann diese weiter verarbeiten. Der Event handleDirectPkt behandelt alle Nachrichten, die nicht über das Overlay gesendet werden. Die Implementierung des Event leitet die Bearbeitung der Nachricht an die entsprechenden internen Funktionen des Moduls weiter. Der Event handleOverlayPkt prüft, ob die Nachricht an einen Punkt im SCAN adressiert ist, der in der eigenen Zone des Knotens liegt. Falls dies der Fall ist, ruft der Knoten die entsprechende interne Funktion des Moduls auf, um die Nachricht weiter zu verarbeiten. Ansonsten wird durch den Overlay-Routing-Algorithmus bestimmt, an welchen der Nachbarknoten die Nachricht weitergeleitet wird. Der Overlay-Routing-Algorithmus wurde bereits in Abschnitt 5.7.2 in Pseudocode vorgestellt und wird von der Prototyp-Implementierung analog implementiert. Die für den Overlay-Routing-Algorithmus benötigte Nachbarliste besteht aus folgenden Einträgen:

```
typedef struct {
  scanNeighb neighbor;
  uint8_t flags;
  uint8_t abutDim;
}neighbEntry;
```

Die Datenstruktur `scanNeighb` dient zur Speicherung der Zone eines Nachbarn mitsamt seiner Vermittlungsschichtadresse. Die Einträge `flags` und `abutDim` dienen der Verwaltung der Nachbarn: `flags` beinhaltet ein Bit das angibt, ob ein Nachbar noch gültig ist oder bei der nächsten Bereinigung der Nachbarliste entfernt werden kann. Der Eintrag `abutDim` gibt an, in welcher Dimension der Nachbar angrenzt und erleichtert damit die Verarbeitung beim Overlay-Routing, da nicht jedes mal erneut auf Grund der Zonenbeschreibung in `neihbor` ermittelt werden muss, in welcher Dimension ein Nachbar angrenzt. Hier ist ein Tradeoff zwischen Verarbeitungsgeschwindigkeit und Speicherverbrauch möglich: wird der Eintrag `abutDim` weggelassen, so könnten in der aktuellen Implementierung auf jedem Knoten noch 17 Byte Speicher eingespart werden, allerdings auf Kosten einer höheren Verarbeitungszeit.

`ScanM` bietet das Interface `ServiceMangager` an, welches Events zur Verarbeitung aller Nachrichten beinhaltet, die Informationen über Dienstbeschreibungen benötigen.

```
interface ServiceManager{
  event result_t zoneTransferRcvd(TOS_MsgPtr m);
  event result_t zoneTransferNRcvd(TOS_MsgPtr m);
  event result_t sendZone(unit16_t dest, scanZone zone);
  event result_t insertReqRcvd(TOS_MsgPtr m);
  event result_t lookupReqRcvd(TOS_MsgPtr m);
}
```

- Im `ServiceM` Modul werden die Dienstbeschreibungen, die ein Sensorknoten in seiner Zone speichert, in einer Diensteliste verwaltet. Auch hierzu kommen statische Arrays zum Einsatz. Das `ServiceM` Modul benutzt das Interface `ServiceManager`, das von `ScanM` angeboten wird. `ServiceM` implementiert also folgende Events:

```
event result_t zoneTransferRcvd(TOS_MsgPtr m);
event result_t zoneTransferNRcvd(TOS_MsgPtr m);
event result_t sendZone(unit16_t dest, scanZone zone);
event result_t insertReqRcvd(TOS_MsgPtr m);
event result_t lookupReqRcvd(TOS_MsgPtr m);
event result_t lookupRepRcvd(TOS_MsgPtr m);
```

Dabei dient der `zoneTransferRcvd` Event zur Verarbeitung von `ZONE_TRANSFER`-Nachrichten. Da die Nachrichtenlänge in TinyOS wie gesagt begrenzt ist, können weitere `ZONE_TRANSFER_N`-Nachrichten folgen, die von dem Event `zoneTransferNRcvd` verarbeitet werden. Der Event `sendZone` wird ausgelöst, um den Zonentransfer zu beginnen. Der Event `insertReqRcvd` dient zur Verarbeitung von empfangenen `INSERT`-Nachrichten. Die Events `lookupReqRcvd` und `lookupRepRcvd` dienen entsprechend zur Verarbeitung der `LOOKUP_REQ`- bzw. `LOOKUP_REP`-Nachrichten.

Darüber hinaus bietet `ServiceM` das Interface `SCAN` an:

```
interface SCAN{
  command result_t insert(uint8_t key[],uint8_t value[]);
  command result_t lookup(uint8_t key[]);
  event result_t insert_done();
  event result_t lookup_done
                (uint8_t lookup_result[][],uint8_t result_number);
}
```

Die Kommandos `insert` und `lookup` stoßen die Insert- bzw. Lookup-Operation an. Dem Kommando `insert` wird der Hash-Wert des Dienstnamens in `key` übergeben und die entsprechende Dienstbeschreibung in `value`. Die Events `insert_done` und `lookup_done` werden von den Modulen implementiert, die das Interface benutzen. Beide Events dienen zur Rückmeldung, dass die Insert- bzw. Lookup-Operation abgeschlossen ist. Im Fall von `lookup_done` wird außerdem eine Liste mit gefundenen Diensten (`lookup_result`) sowie die Anzahl der gefundenen Dienst (`result_number`) übergeben.

- Das `TalassaAdapterM` Modul dient als Adapter zwischen SCAN und Talassa: Durch das Modul erfolgt die Anpassung der Schnittstellen. Dies ist insbesondere notwendig, da die aktuelle Talassa-Implementierung Dienste von Dienstanbietern anhand von eindeutigen IDs erkennt, die SCAN nicht voraussetzt. Um die SCAN-Schnittstelle nicht anpassen zu müssen, wurden die Talassa-IDs in die Dienstbeschreibungen integriert und das `TalassaAdapterM` Modul nimmt die Wandlung vor. Das Modul `TalassaAdapterM` bietet ebenfalls das Interface `SCAN` an, allerdings unter dem Namen `TalassaAdapter`. Das Interface `TalassaAdapter` wird von Talassa genutzt, das entsprechend die beiden Events `insert_done` und `lookup_done` implementiert. Über diese beiden Events wird an Talassa der Abschluss der Insert- bzw. Lookup-Operation signalisieren und, im Fall von `lookup_done`, die Diensteliste zurückgeliefert. Talassa ruft das Kommando `insert` auf, um eine Dienstbeschreibung (enthalten in `value`) unter dem Hash-Wert `key` einzutragen. Die eigentliche Adapter-Funktionaliät findet sich in dem Kommando `lookup` und dem Event `lookup_done`: In `key` übergibt Talassa nicht nur den Hash-Wert des Dienstnamens an die Lookup-Operation, sondern auch die entsprechende Talassa-ID. Die Talassa-ID wird im Modul `TalassaAdapterM` gespeichert. Ist der Lookup abgeschlossen, was durch den

Event `lookup_done` angezeigt wird, so wird über `lookup_done` in `result` lediglich eine Liste derjenigen Dienstbeschreibungen zurückgeliefert, welche die Talassa-ID in ihrer Dienstbeschreibung beinhalten, die Talassa in `lookup` angegeben hat.

6.2.3.3 SPX und MPX

Für [47] kamen SPX und MPX noch nicht zum Einsatz, da der Schwerpunkt auf der Evaluierung der Kommunikation sowie des Speicherverbrauch lag. Mit [76] stehen aber Implementierungen von SPX und MPX zur Verfügung. Die entsprechenden kryptographischen Algorithmen lieferte [54]. Die Implementierung benötigt weniger als 4 Kilobyte Hauptspeicher und weniger als 128 Kilobyte ROM, ist also für die MICAz Sensorknoten [104] geeignet.

6.2.4 Implementierung des Master Device

Als Master Device wird ein PDA verwendet, genauer gesagt ein HP iPaq der hx4700 Familie. Um einen Location Limited Channel herzustellen wird der PDA über seine serielle Schnittstelle mit einem Kabel mit den Sensorknoten verbunden. Sowohl iPaq als auch die Sensorknoten bieten eine serielle Schnittstelle, so dass die entsprechende Verbindung kein Problem darstellt. Allerdings war eine Pegelanpassung notwendig, da die seriellen Schnittstelle des iPaq einen anderen Spannungspegel als die serielle Schnittstelle der Sensorknoten verwendet. Kommunikation über die serielle Schnittstelle ist ein Location Limited Channel (siehe Abschnitt 5.2), da sie die drei Eigenschaften eines Location Limited Channel besitzt: es handelt sich um einen Seitenkanal, dieser ist authentisch und vertraulich und bietet Demonstrative Identification, da klar ist, mit welchem Sensorknoten das Master Device über ein Kabel verbunden ist.

Das Benutzer-Interface des Master Device ist über Karteireiter zweigeteilt: Während der Karteireiter „Main“ alle für die für die Benutzung des Master Device notwendigen Funktionen bietet, dient der Karteireiter „Log“ dem Debugging und der Visualisierung. Die beiden Ansichten sind in Abbildung 6.47 und 6.48 zu sehen.

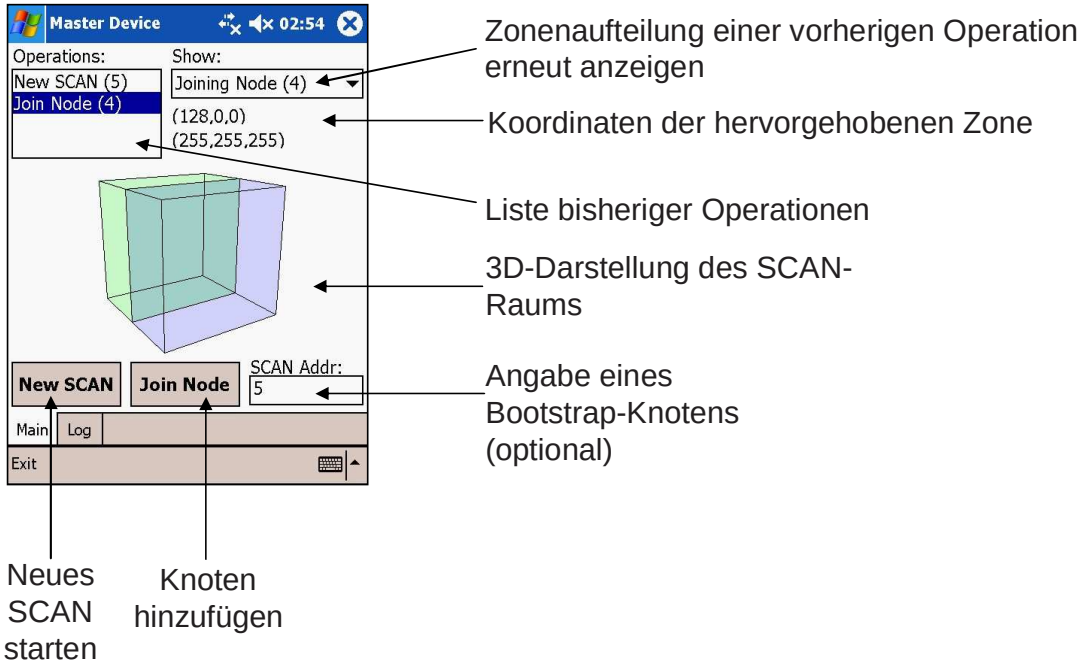


Abbildung 6.47: Der Karteireiter „Main“ des Master Device

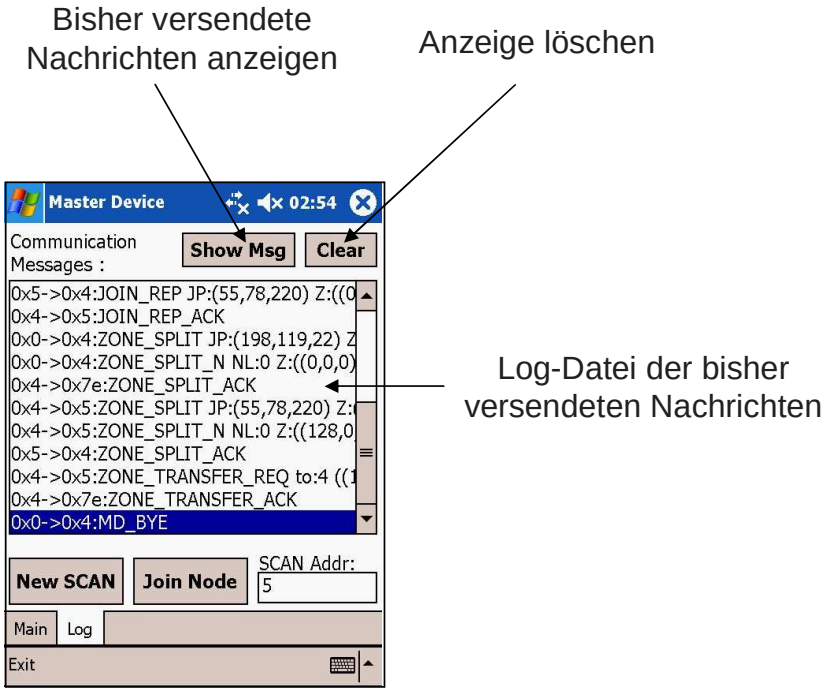


Abbildung 6.48: Der Karteireiter „Log“ des Master Device

6.2.5 Visualisierung von SCAN im Prototyp

Zu Visualisierungszwecken wurde auf einem iPaq das Programm „Node Control“ entwickelt,um den inneren Zustand eines Sensorknotens darzustellen. So kann z. B. der Teil der verteilte Hash-Tabelle angezeigt werden, welcher auf einem Knoten verwaltet wird. Der iPaq zur Visualisierung kommuniziert, ebenso wie das Master Device, über die serielle Schnittstelle mit einem Sensorknoten. „Node Control“ verteilt die anzuzeigenden Informationen auf 5 Karteireiter: „Zone“, „SCAN“, „Services“, „Hardware“ und „Log“. Für die Visualisierung von SCAN sind vor allem die Karteireiter „Zone“ und „SCAN“ verantwortlich, die in Abbildung 6.49 und in Abbildung 6.50 gezeigt werden.

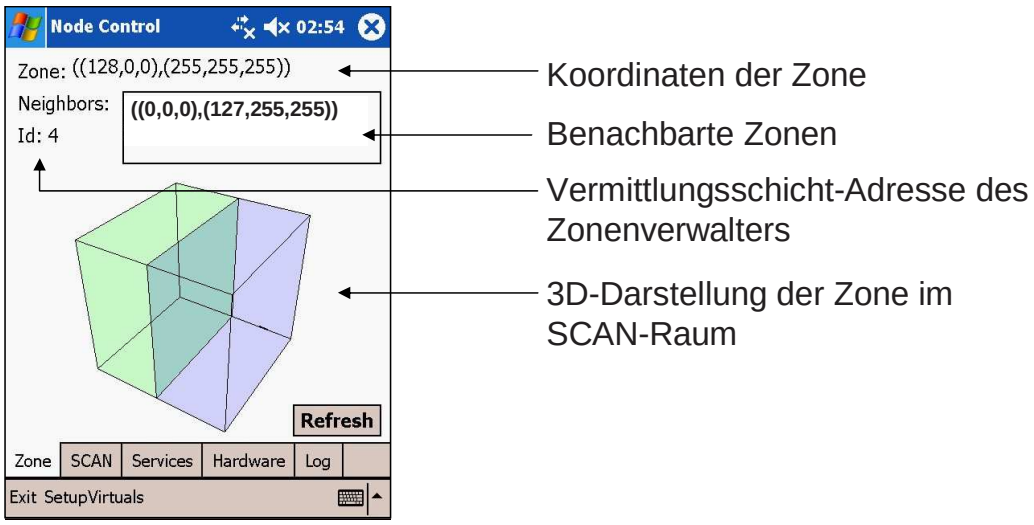


Abbildung 6.49: Der Karteireiter „Zone“ des Programms „Node Control“

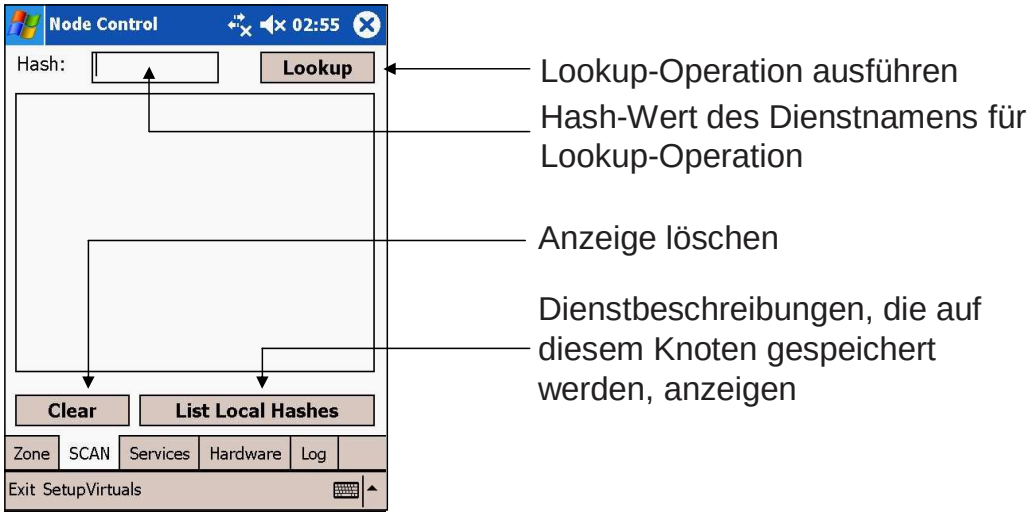


Abbildung 6.50: Der Karteireiter „SCAN“ des Programms „Node Control“

6.2.6 Evaluierung

Mit der Prototyp-Implementierung konnte die Funktionalität von SCAN auf leistungs- und ressourcenschwachen Sensorknoten gezeigt werden. Darüber hinaus wurden im Szenario „intelligente Gärtnerei“ einige Messungen vorgenommen, um zu quantitative Aussagen zu kommen, die durch Simulationen nur schwer zu erhalten sind, insbesondere zum Speicherverbrauch und zu zeitlichen Aspekten.

6.2.6.1 Speicherverbrauch

Die Implementierung von SCAN benötigt in der getesteten Konfiguration 1906 Bytes RAM. Damit bleibt der Speicherverbrauch deutlich unter den auf MICAz Sensorknoten vorhandenen 4 Kilobyte RAM. Auch im Flash-Speicher benötigt die Implementierung von SCAN nur einen Teil der vorhandenen Ressourcen: Die Implementierung benötigt 18.35 Kilobyte von 128 Kilobyte Flash-Speicher. Die Sensorknoten konnten bis zu 30 Dienstbeschreibungen pro Knoten speichern. Diese Beschränkung ist vor allem durch die statischen Arrays bedingt. Diese Werte beziehen sich auf ein SCAN der Dimension $d = 3$. Würde ein SCAN der Dimension $d = 7$ verwendet werden, so würde sich der Speicherbedarf im RAM vor allem dadurch erhöhen, dass mehr Nachbarn in der Nachbarliste enthalten sind. Im Fall $d = 7$ ist mit einem Speicherverbrauch von 2086 Bytes RAM zu rechnen, eine Änderung im Flash-Speicher ergibt sich nicht.

6.2.6.2 Zeitliche Aspekte der Kommunikation

An der Prototyp-Implementierung wurden zeitliche Aspekte untersucht, die anhand der Implementierung im Simulator nur unzureichend geklärt werden können. So wurde z. B. untersucht, wie lange die Join-Operation dauert in Abhängigkeit von der Anzahl von Knoten, die bereits Mitglied im SCAN sind. Abbildung 6.51 zeigt

Mittelwert sowie das 95%-Konfidenzintervall der Dauer der Join-Operation von 10 Messungen. Bei jeder Messung wurden 28 Knoten nacheinander ins SCAN integriert. Deutlich wird sichtbar, dass die Zeitdauer, nach einer gewissen Anzahl von Join-Operationen, nur noch leicht steigt. Die Join-Operation dauert ca. 3 bis 4 Sekunden. Die Dauer der Join-Operation wächst in ähnlichem Maße wie die durchschnittliche Pfadlänge zunimmt. Für 0 bis 28 Knoten ist die theoretische durchschnittliche Pfadlänge in Overlay-Hops in Abbildung 6.52 dargestellt. Dies deutet darauf hin, dass die Kommunikationsvorgänge, die im Rahmen der Join-Operation ausgeführt werden, zu einem großen Teil für die Dauer der Join-Operation verantwortlich sind. Die Schwankungen bei der Dauer der Join-Operation sind vor allem durch Übertragungswiederholungen bedingt, die in einzelnen Messreihen aufgetraten. Da die entsprechenden Timeouts auf relativ hohe Werte gesetzt wurden (500 ms), fallen die Ausreißer stark ins Gewicht.

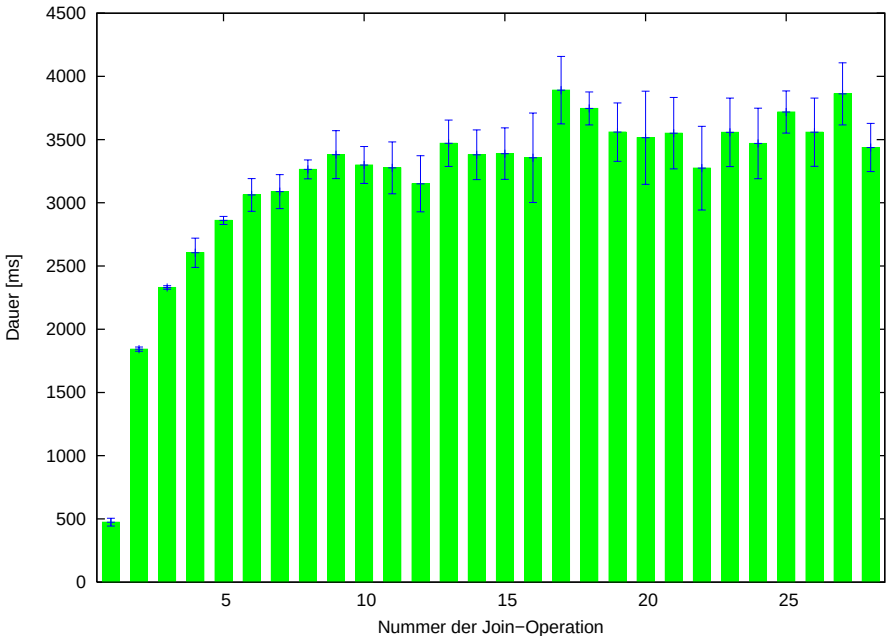


Abbildung 6.51: Zeitdauer einer Join-Operation in einem SCAN mit 1-28 Knoten

Auch die Zeitdauer der Lookup-Operation wurde untersucht: dazu wurden in einem Sensornetz, das aus 28 Knoten bestand, unter vier verschiedenen Dienstnamen jeweils eine Dienstbeschreibung abgelegt. Dabei wurde durch die Wahl der Dienstnamen darauf geachtet, dass die Werte über den gesamten SCAN-Raum verteilt sind. Anschließend wurde auf jedem Knoten für jeden Dienstnamen eine Lookup-Operation ausgeführt. In Abbildung 6.53 ist die Verteilung der Zeitdauer der Lookup-Operationen zu sehen. Alle Lookup-Operationen waren innerhalb 120 bis 200 ms abgeschlossen. Dies dürfte für die meisten Sensornetzanwendungen völlig ausreichend sein.

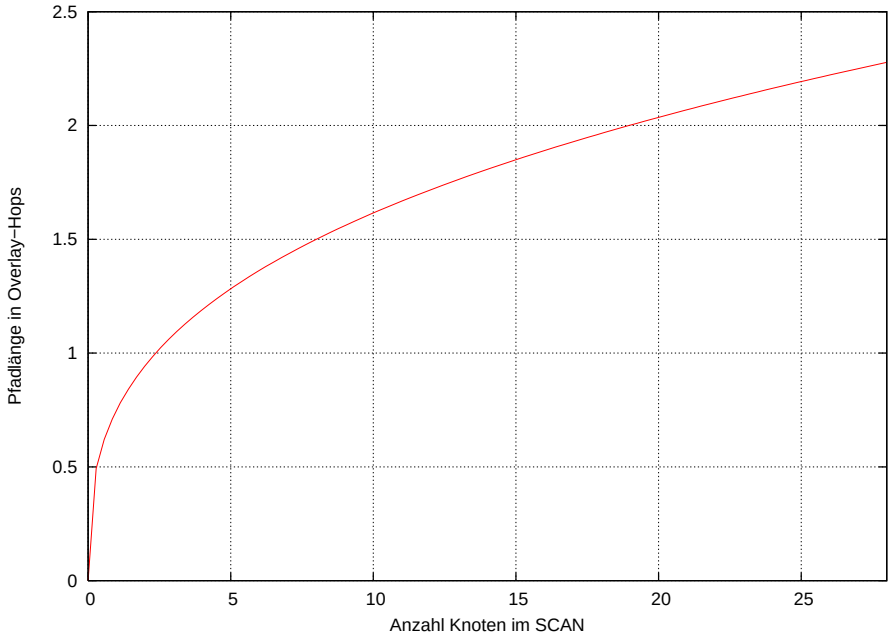


Abbildung 6.52: Durchschnittliche Pfadlänge im Overlay ($d = 3$) bei steigender Knotenanzahl

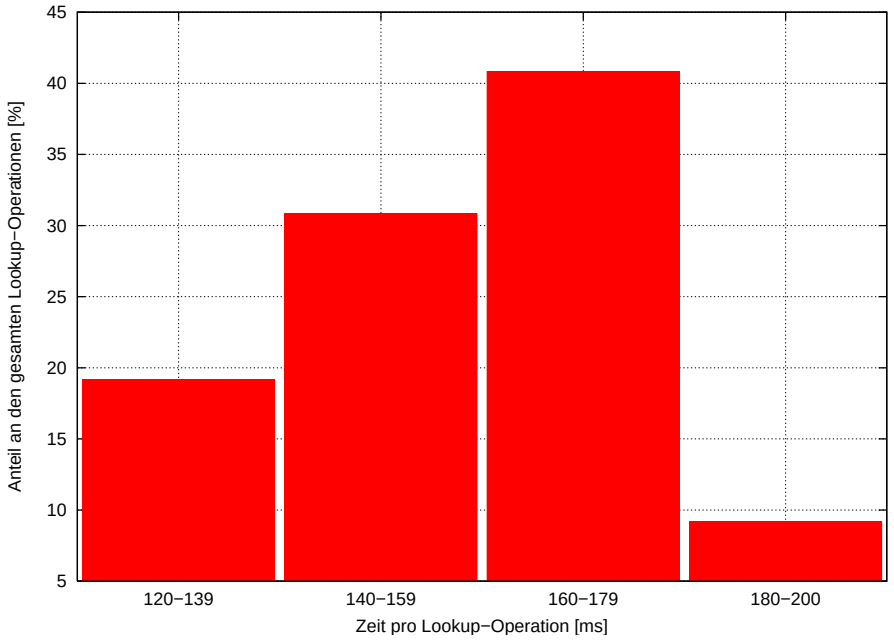


Abbildung 6.53: Verteilung der Zeitdauer einer Lookup-Operation

6.3 Zusammenfassung

Die Leistungsanalyse sowohl im Simulator GloMoSim als auch der Prototyp-Implementierung auf MICAz Sensorknoten hat gezeigt, dass SCAN für die sichere Dienstesuche in Sensornetzen eingesetzt werden kann, da die Lookup- und Insert-Operation pro Knoten und Operation nur wenige hundert Byte Kommunikationsaufwand erzeugen, die Implementierung auch auf leistungs- und ressourcenschwachen Sensorknoten möglich ist, und die Join- und Lookup-Operation innerhalb akzeptabler Zeitdauer ausgeführt werden. SPX und MPX können bei der Lookup- und Insert-Operation zum Einsatz kommen, um die Wahrscheinlichkeit zu erhöhen, eine gesuchte Dienstbeschreibung zu erhalten, die nicht von einem Angreifer manipuliert wurde. Diese so genannte Lookup-Wahrscheinlichkeit ist bei Verwendung von MPX unter einer moderaten Anzahl von Angreifern höher als bei SPX. Bei SPX kann die Lookup-Wahrscheinlichkeit noch einmal deutlich durch die Verwendung von Replikaten gesteigert werden. Bei MPX hängt der Erfolg des Einsatzes von Replikation davon ab, wie viele Dienstbeschreibungen im SCAN hinterlegt werden, da die Gefahr besteht, zu viel Kommunikationsaufwand zu erzeugen. Die Join-Operation erzeugt deutlich mehr Kommunikations-Overhead als die Insert- und die Lookup-Operation, allerdings wird die Join Operation lediglich ein einziges Mal für jeden Knoten des Overlays aufgerufen, während die Lookup- und Insert-Operationen ständig eingesetzt werden.

7. Zusammenfassung und Ausblick

Im Zuge der fortschreitenden Miniaturisierung von Elektronikbauteilen werden zunehmend kleinere Miniatur-Computer möglich. Werden diese Miniatur-Computer drahtlos miteinander verbunden, so entstehen Netze mit einem hohen Anwendungspotential, z. B. im Bereich Krankenpflege, Home Automation oder Umweltüberwachung. Drahtlose Sensornetze sind eine Unterklasse dieser Netze. Ein drahtloses Sensornetz besteht aus Sensorknoten, die drahtlos, üblicherweise über Funk, miteinander kommunizieren, um durch Sensoren gemessene Werte zu übermitteln oder um Aktoren anzusteuern. Dienstorientierte Sensornetze sind wiederum eine Unterklasse von Sensornetzen, die sich dadurch auszeichnet, dass eine Dienstabstraktion verwendet wird, welche die Bereitstellung jedweder Funktionalität in Form von Diensten ermöglicht. Um in solchen dienstorientierten Sensornetzen eine verteilte Anwendung flexibel erstellen zu können, muss insbesondere das Problem gelöst werden, wie verfügbare Dienste aufgefunden werden können. Ein weiterer wichtiger Aspekt ist die Sicherheit dieser Dienste-Suche: es muss verhindert werden, dass ein Angreifer eine verteilte Anwendung, die aus Diensten zusammengesetzt ist, manipuliert, indem er die Dienste-Suche dazu benutzt, einen korrumpierten Dienst unberechtigt einzuschleusen. Sensorknoten zeichnen sich vor allem durch ihre beschränkten Ressourcen aus. So ist z. B. der Speicher knapp bemessen und den Sensorknoten steht nur ein geringer Vorrat an Energie zur Verfügung, da Sensorknoten meist batteriebetrieben sind. Diese Randbedingungen stellen eine anspruchsvolle Herausforderung für jedes Verfahren für Sensornetze dar.

Bisherige Lösungsansätze behandeln die Thematik nur unzureichend: keine der vorhandenen Lösungen zur Dienste-Suche erfüllt in Sensornetzen die Anforderungen Funktionalität, Effizienz, Robustheit, Skalierbarkeit, Dezentralität und Sicherheit. Besonders die Anforderung Sicherheit wurde bisher nur unzureichend betrachtet.

7.1 Ergebnisse der Arbeit

Im Rahmen dieser Arbeit wurde mit Secure Content Addressable Network (SCAN) ein sicheres, verteiltes Diensteverzeichnis vorgestellt, mit Hilfe dessen in dienstorientierten Sensornetzen verfügbare Dienste sicher aufgefunden werden können. Dazu ermöglicht es SCAN einem Dienstanbieter, Dienstbeschreibungen zu den von ihm angebotenen Diensten unter einem Dienstnamen zu veröffentlichen. Diese Dienstbeschreibungen werden verteilt gespeichert. Ein Dienstnutzer kann zu einem Dienstnamen eine Liste von Dienstbeschreibungen anfordern. Die Integrität und Vertraulichkeit der Dienstbeschreibungen werden sowohl während der Übertragung durch das Sensornetz als auch während der verteilten Speicherung geschützt. SCAN stellt sicher, dass nur vertrauenswürdige Knoten am verteilten Diensteverzeichnis teilnehmen, indem Knoten vor dem Beitritt authentifiziert und zum Beitritt autorisiert werden. Darüber hinaus etabliert SCAN Schlüssel zwischen diesen Knoten, mit deren Hilfe Integrität und Vertraulichkeit der Kommunikation zwischen Knoten geschützt werden können. Dazu dienen unter anderem die Schlüsselaustauschprotokolle Single-Path-Key-Exchange (SPX) und Multi-Path-Key-Exchange (MPX). SCAN verhindert, dass Outsider-Angreifer, also Angreifer, die nicht dem SCAN angehören, während dem Veröffentlichen oder Auffinden von Dienstbeschreibungen manipulierte Dienstbeschreibungen einschleusen, um so eine verteilte, aus Diensten aufgebaute, Sensornetz-Anwendung zu infiltrieren. Gegen zusammenarbeitende Insider-Angreifer, also Angreifer, die zu SCAN gehören, bietet SCAN mit hoher Wahrscheinlichkeit Schutz. Außerdem schützt SCAN die Integrität und Vertraulichkeit von Dienstbeschreibungen während der Speicherung gegen Outsider- und Insider-Angreifer.

SCAN nimmt besondere Rücksicht auf die speziellen Eigenschaften von Sensorknoten, insbesondere auf die sehr beschränkten Ressourcen wie z. B. Speicher (RAM). SCAN verwendet nur symmetrische kryptographische Verfahren, da Implementierungen symmetrischer kryptographischer Verfahren auf Sensorknoten, im Vergleich zu Implementierungen von asymmetrischen Verfahren, wesentlich effizienter sind und somit die zur Verfügung stehenden Ressourcen optimal nutzen.

Die Leistungsfähigkeit von SCAN in Sensornetzen mit bis zu 1500 Knoten wurde anhand einer Simulation in GloMoSim gezeigt. Darüber hinaus wurde SCAN auf den handelsüblichen MICAz Sensorknoten als Prototyp implementiert. Hier zeigte sich, dass SCAN auf ressourcen- und leistungsschwachen Sensorknoten realisiert werden kann, ohne diese Knoten zu überfordern. Eine Untersuchung der zeitlichen Aspekte, insbesondere der Dauer des Beitritts eines Knotens zum SCAN sowie der Dauer des Auffindens einer Liste verfügbarer Dienstbeschreibungen, zeigte, dass SCAN sich für dienstorientierte Sensornetze eignet.

7.2 Weiterführende Arbeiten

In dieser Arbeit wurde das Problem der sicheren Dienste-Suche in dienstorientierten Sensornetzen betrachtet und mit SCAN gelöst. Ein nächster Schritt hin zu sicheren

dienstorientierten Sensornetzen ist der sichere Zugriff auf Dienste. SCAN legt hierzu bereits einen Grundstein mit den Schlüsselaustauschprotokollen SPX und MPX: Diese eignen sich grundsätzlich auch dazu, einen Schlüssel zwischen einem Dienstnutzer und einem Dienstanbieter auszutauschen, solange beide Mitglied des SCAN sind. Dieser Schlüssel kann für den an die Dienste-Suche anschließenden Zugriff auf Dienste verwendet werden, um Integrität und Vertraulichkeit der Kommunikation zwischen dem Dienstanbieter und dem Dienstnehmer zu sichern. Eine weitere Möglichkeit zum sicheren Zugriff auf Diensten ist, Schlüsselmaterial zum Zugriff auf Dienste in den Dienstbeschreibungen zu verankern. Dies wäre insbesondere in Kombination mit vertraulichen Dienstbeschreibungen ein vielversprechender Ansatz. Dieser Aspekt wurde bereits in [93] angerissen, verdient aber eine weitere Ausarbeitung.

Um die Effizienz von SCAN insbesondere in sehr großen Netzen mit 10000 und mehr Knoten zu steigern wäre es möglich, eine Cluster-Architektur einzusetzen. Bisher wurde davon ausgegangen, dass alle Knoten des Sensornetzes auch Mitglied im SCAN sind. Es wäre jedoch auch möglich, SCAN mit einer Cluster-Architektur zu kombinieren, so dass ein Clusterhead pro Cluster dem SCAN beitritt und die Mitglieder des Clusters nur über den Clusterhead mit SCAN kommunizieren. Dieser Ansatz wurde bereits in [114] untersucht. Um die Sicherheit innerhalb eines Clusters zu gewährleisten, können herkömmliche sichere Cluster-Architekturen verwendet werden wie z. B. [14].

Eine weitere Senkung des Kommunikationsaufwands von SCAN wäre durch eine semantische Anfragesprache möglich, die es einem Dienstnehmer ermöglicht, beim Auffinden von Diensten neben dem Hash-Wert des Dienstnamens noch weitere Angaben zum gewünschten Dienst zu machen. Der speichernde Knoten kann diese Angaben auswerten und nur die Dienstbeschreibungen zurück liefern, die den Angaben entsprechen. Dazu ist es allerdings notwendig, dass SCAN Dienstbeschreibungen interpretieren kann, da die Entscheidung auf Basis von Einträgen in den Dienstbeschreibungen geschehen muss.

In dieser Arbeit wurde SCAN speziell für den Einsatz in Sensornetzen entworfen. Jedoch lassen sich die im Rahmen dieser Arbeit entwickelten Mechanismen und Protokolle auch in anderen Netzen einsetzen. Insbesondere die Schlüsselaustauschprotokolle Single-Path-Key-Exchange und Multi-Path-Key-Exchange eignen sich nicht nur für die Dienste-Suche in Sensornetzen, sondern können in vielen anderen sicherheitsrelevanten Anwendungen, auch in anderen Netzen, eingesetzt werden, falls dort die Verwendung des Peer-to-Peer-Netzes CAN möglich ist und ein Schüsselaustausch zwischen beliebigen Knoten benötigt wird.

A. Übersicht über in dieser Arbeit verwendete Symbole

Tabelle A.1 listet Symbole auf, die kapitelübergreifend verwendet werden und für die Arbeit von besonderer Bedeutung sind.

Im Rahmen der Arbeit wird der SCAN-Raum immer zweidimensional dargestellt wie in Abbildung A.1 zu sehen. Dabei ist zu beachten, dass lediglich ein Ausschnitt des SCAN-Raums zu sehen ist, so dass das Overlay-Routing nie über die Ränder hinweg geschieht.

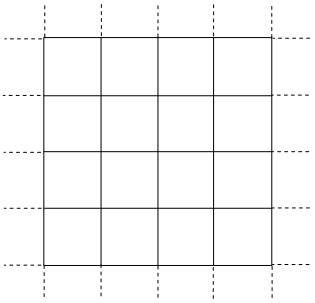


Abbildung A.1: Darstellung des SCAN-Raums

Das Overlay-Routing in diesem Raum wird wie in Abbildung A.2 zu sehen ist mittels einfacher Pfeile dargestellt.

Existiert zwischen den Zonenverwalter von zwei benachbarten Zonen ein gemeinsamer Schlüssel, so wird dies durch einen dicken Pfeil dargestellt wie in Abbildung A.3 zu sehen ist. Ist die Benennung des Schlüssels von Bedeutung, so wird der Name des Schlüssels im Pfeil notiert (*key* im Beispiel).

Abbildung A.4 zeigt einen Punkt JP im SCAN-Raum. Dabei wird die Koordinate des Punkts grafisch mit einem Stern dargestellt.

Symbol	Bedeutung
d	Dimension des vom Overlay (CAN oder SCAN) verwendeten virtuellen Raums.
N	Anzahl von Sensorknoten, aus denen das Overlay (CAN oder SCAN) gebildet wird.
k	Anzahl der Schlüsselteile die benötigt werden, um den Schlüssel zu rekonstruieren.
n	Anzahl der Schlüsselteile, die aus einem Schüssel erzeugt werden.
m	Anteil Angreifer bzw. Anteil bösartiger Knoten. Der Anteil wird entweder ohne Einheit ($\in [0,1]$) oder als Prozentsatz ($\in [0\%, 100\%]$) angegeben.
f	Anteil ausfallender Knoten. Der Anteil wird entweder ohne Einheit ($\in [0,1]$) oder als Prozentsatz ($\in [0\%, 100\%]$) angegeben.
r	Anzahl Replikate. Gibt an, wie oft eine Dienstbeschreibung im SCAN hinterlegt/abgefragt wird.
p_{bez}	Wahrscheinlichkeit, die über den Bezeichner *bez* genauer bestimmt wird.
$key_{A,B}$	Symmetrischer Schlüssel zwischen einem Knoten A und einem Knoten B.
key_{bez}	Schlüssel, der nicht für die Kommunikation zwischen zwei Knoten dient. Dies kann einerseits ein Schlüssel sein, der auf einem Knoten lokal verwendet wird oder andererseits ein Schlüssel, der in einer Gruppe von mehr als zwei Knoten verwendet wird. Der Schlüssel wird durch *bez* bezeichnet. Die Bezeichnung eines Gruppenschlüssels beginnt immer mit „grp“.
$E_{key}()$	Verschlüsselungsfunktion, die mit dem Schlüssel *key* parametrisiert ist.
$hash()$	Hash-Funktion.
mac	Message Authentication Code.
uid_{bez}	Nonce mit der Bezeichnung *bez*.
$A\|B$	Anhängen des Wertes B an den Wert A (Konkatenation).
K_i	Bezeichnung eines von mehreren Sensorknoten. Sensorknoten werden immer mit einem oder mit zwei Buchstaben bezeichnet sowie optional mit einem Index.
$\neg A$	Negierung der Aussage A.
$\oplus$	Exklusives oder.
$x \prec y$	x befindet sich in einer Ordnung vor y.
$\lfloor x \rfloor$	Gauß-Klammer: liefert die größte ganze Zahl kleiner oder gleich x zurück.

Tabelle A.1: Verwendete Symbole

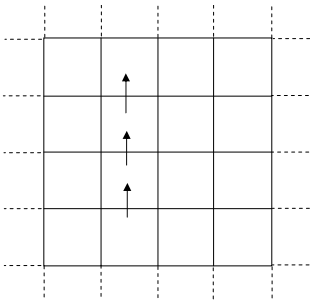


Abbildung A.2: Darstellung des Overlay-Routings im SCAN-Raum

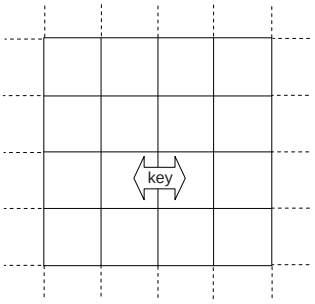


Abbildung A.3: Darstellung gemeinsamer Schlüssel

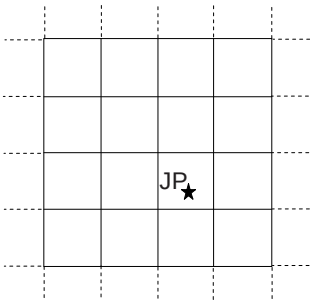


Abbildung A.4: Darstellung eines Punktes im SCAN-Raum

B. Test auf Normalverteilung der Simulationsergebnisse

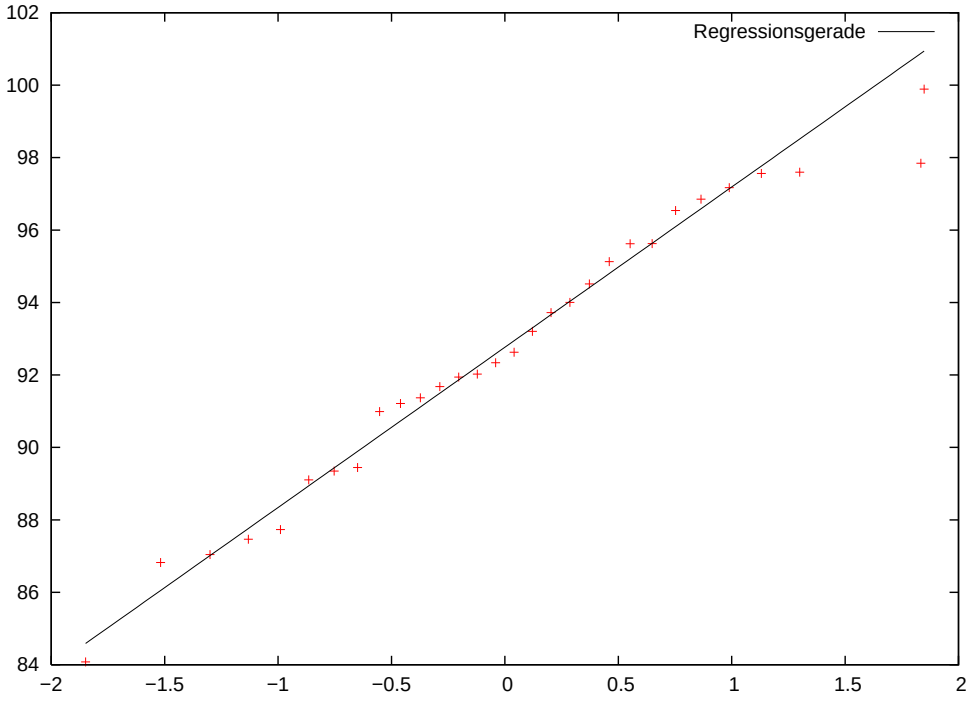


Abbildung B.1: Test auf Normalverteilungsannahme der Simulationsergebnisse für die Lookup-Wahrscheinlichkeit bei Verwendung von SPX und 500 Knoten

Wie in Kapitel 6 erwähnt, wurden die Simulationen zum größten Teil mit 30 Simulationsläufen durchgeführt. Zur statistischen Absicherung der erhaltenen Ergebnisse ist es sinnvoll, Konfidenzintervalle für die erhaltenen Mittelwerte, z. B. die durchschnittliche Lookup-Wahrscheinlichkeit, zu bestimmen. Dabei wurde angenommen, dass die

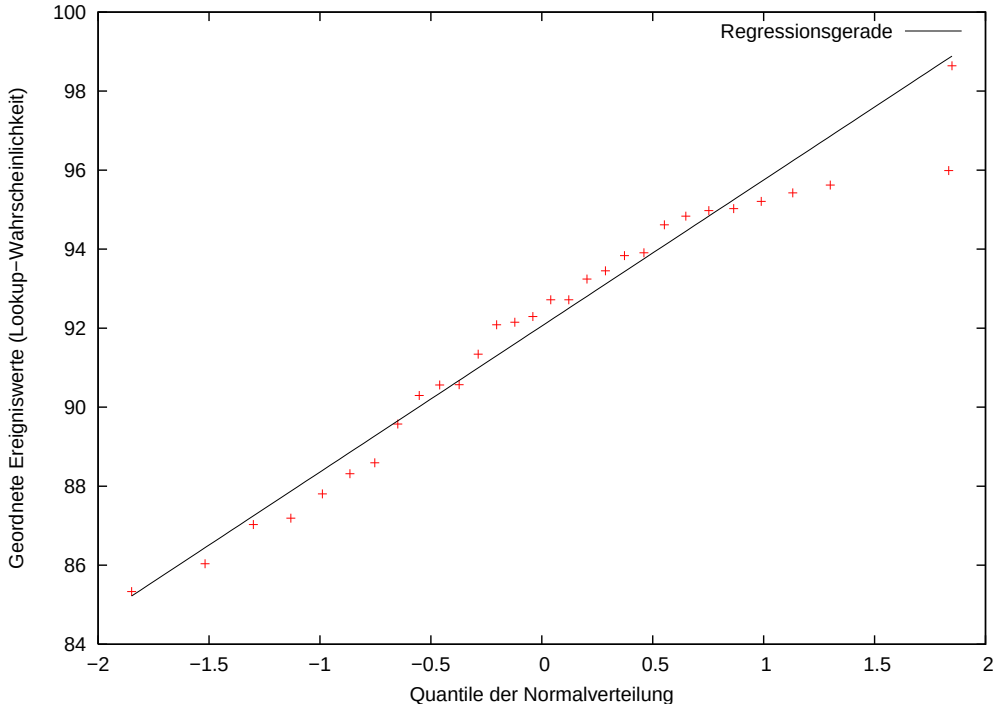


Abbildung B.2: Test auf Normalverteilungsannahme der Simulationsergebnisse für die Lookup-Wahrscheinlichkeit bei Verwendung von SPX und 1000 Knoten

durch die Simulationsläufe ermittelten Stichprobenwerte normalverteilt sind. Durch einen Regressionstest der n Simulationsergebnisse (als geordnete Stichprobe) mit den $\frac{k}{n-1}$-Quantilen der Standardnormalverteilung kann nach [78] die Normalverteilungsannahme überprüft werden. Abbildung B.1 zeigt einen Quantil-Quantil-Plot mit Regressionsgerade für die durchschnittliche Lookup-Wahrscheinlichkeit bei Verwendung von SPX für die Lookup- und Insert-Operation in einem SCAN mit 500 Knoten und einem Prozent Angreifern. Abbildung B.2 zeigt dasselbe für ein SCAN mit 1000 Knoten. Für 500 Knoten beträgt der Korrelationskoeffizient $r = 0.9869$, für 1000 Knoten gilt $r = 0.9788$. Aus den Abbildungen und dem Korrelationskoeffizienten geht hervor, dass es keinen Grund gibt, die Normalverteilungsannahme abzulehnen.

C. Bewertung der Eignung kryptographischer Bausteine

Ver- und Entschlüsselung, Digitale Signaturen, Hash-Funktionen und Message Authentication Codes sind kryptographische Bausteine, die oft in sicheren Protokollen eingesetzt werden. Bedingt durch die besonderen Bedingungen in Sensornetzen, insbesondere die beschränkten Ressourcen, muss deren Einsatzfähigkeit in Sensornetzen überdacht werden. Für eine Einsatzfähigkeit in Sensornetzen spielen unter anderem die Faktoren Rechenzeit, Energieeffizienz und Speicherverbrauch eine große Rolle. Um eine Bewertung vornehmen zu können, wurden Forschungsarbeiten betrachtet, die eine Aussage über die entsprechenden Bausteine auf für Sensornetzen typischen Hardware-Plattformen vornehmen [37, 38, 59, 64, 75, 109–111]. Die angegebenen Werte beziehen sich immer auf eine konkrete Implementierung der Verfahren, so dass Fortschritte bei in der Implementierung der entsprechenden Verfahren durchaus deutliche Leistungsverbesserungen mit sich bringen können. Es wurde deshalb auch versucht, aus den Arbeiten eine allgemeine Aussage bezüglich Größenordnungen herzuleiten.

C.1 Bewertungskriterien

Für die Beurteilung der Ressourcen-Sparsamkeit wurden folgende Bewertungskriterien angelegt:

- *Rechenzeit:* Die eingesetzten Algorithmen sollen nicht zu rechenintensiv sein, da die in dieser Arbeit betrachteten Sensorknoten nur über wenig Rechenleistung verfügen. Benötigt ein Sensorknoten z. B. mehrere Sekunden für eine Berechnung, so ist dieser Algorithmus nur für Aufgaben geeignet, die selten ausgeführt werden müssen.

- *Energieeffizienz:* Dicht verwandt mit der Rechenzeit ist der Energieverbrauch eines kryptographischen Algorithmus. Sensorknoten sind üblicherweise batteriebetrieben, d. h. sie verfügen nur über einen begrenzten Energievorrat, der die Lebenszeit der Sensorknoten maßgeblich bestimmt. Deshalb sollen die in Sensornetzen eingesetzten kryptographischen Bausteine möglichst energieeffizient sein.

- *Speicherverbrauch:* Die eingesetzten Algorithmen sollen nur wenig RAM benötigen, da die in dieser Arbeit betrachteten Sensorknoten nur über wenig Hauptspeicher (z. B. 10 Kilobyte) verfügen. Ebenfalls sollten die Algorithmen nicht zu viel Programmspeicher (Flash-Speicher) verbrauchen, da auch dieser in den hier betrachteten Sensorknoten begrenzt ist (z. B. 128 Kilobyte), wobei die Beschränkung beim Flash-Speicher üblicherweise wesentlich weniger ins Gewicht fällt als die Beschränkung des Hauptspeichers.

Bei der folgenden Bewertung der Implementierung einzelner Algorithmen liegt das Interesse weniger auf den genauen Werten für verbrauchten Speicherplatz und Laufzeit. Vielmehr soll herausgefunden werden, ob Gemeinsamkeiten einzelner Algorithmen-Klassen gefunden werden können, hinsichtlich Größenordnung, in denen sich Laufzeit und Speicherverbrauch bewegen.

C.2 Verschlüsselung und Digitale Signatur

Verschlüsselungs-Algorithmen (so genannte Chiffren) sind grundlegende Bausteine von Protokollen der Kryptographie. Verschlüsselung mit einer Chiffre kann z. B. zum Schutz der Vertraulichkeit eingesetzt werden. In diesem Abschnitt werden verschiedene Implementierungen von Chiffren untersucht und auf Basis der oben angegebenen Bewertungskriterien beurteilt. Die Tabellen C.1 und C.2 geben einen Überblick über die Ergebnisse. Wegen dem kostengünstigen Aufbau von Sensorknoten ist nicht damit zu rechnen, dass dedizierte Hardware zur Unterstützung von Chiffren eingesetzt wird. Möglich ist allerdings, dass solche Hardware in andere Komponenten integriert ist. So ist z. B. in Zukunft zu erwarten, dass Sensorknoten serienmäßig mit einer Hardware-Implementierung der symmetrischen Chiffre AES [30] ausgestattet sind, da die Verwendung von AES im ZigBee-Standard vorgeschrieben wird und damit zu rechnen ist, dass dieser Algorithmus zusammen mit dem ZigBee-Protokoll-Stack in einem separaten Kommunikations-Chip in Hardware integriert werden wird. Erste ZigBee-Chips mit AES in Hardware existieren bereits, z. B. [107]. Es ist damit zu rechnen, dass eine dedizierte AES-Hardware-Implementierung wesentlich energieeffizienter arbeiten wird als eine Software-Implementierung. Da heute diese dedizierte Hardware jedoch noch nicht flächendeckend eingesetzt wird, wird im Folgenden untersucht, wie effizient *Software-Implementierungen von Chiffren und Signatur-Algorithmen* in Sensornetzen sind.

Man unterscheidet grundsätzlich zwei Klassen von Chiffren: *symmetrische Chiffren* und *asymmetrische Chiffren*. Dabei verwenden symmetrische Chiffren zum Ver- und

Entschlüsseln denselben Schlüssel, während asymmetrische Chiffren einen öffentlichen Schlüssel zum Verschlüsseln und einen privaten Schlüsseln zum Entschlüsseln verwenden. In *hybriden Verschlüsselungsverfahren* wird mittels eines asymmetrischen Algorithmus ein Schlüssel zwischen zwei Knoten ausgetauscht, der dann zur Verschlüsselung mittels einer symmetrischer Chiffre verwendet werden kann. Dieses Vorgehen kombiniert die Vorteile von asymmetrischen und symmetrischen Chiffren.

In [109] wird der Energieverbrauch der symmetrischen Chiffre AES auf einem MICA2Dot [28] Sensorknoten, der mit 4 MHz betrieben wird, untersucht. Bei 128 Bit Schlüssellänge des AES werden für die Verschlüsselung pro Byte im Durchschnitt $1.62\mu J$ Energie benötigt. Die Entschlüsselung eines Bytes benötigt im Durchschnitt $2.49\mu J$. In [109] wird die Leistung des Mikroprozessors eines MICA2Dot Sensorknoten im Active Mode (also während Berechnungen) mit $13.8mW$ angegeben. Daraus errechnet sich (nach $P = W/t$, P Leistung, W Arbeit, t Zeit), dass die Verschlüsselung eines Bytes mit AES und einer Schlüssellänge von 128 Bit im Durchschnitt $117\mu s$ benötigt. Die entsprechende Entschlüsselung benötigt im Durchschnitt $180\mu s$.

In [54] wird TinySec vorgestellt, ein System zum Schutz von Integrität und Vertraulichkeit auf Sicherungsschicht. Dabei werden die beiden symmetrischen Chiffren RC5 [83] und Skipjack [99] auf MICA2 Sensorknoten mit einer Taktfrequenz von 8 MHz untersucht. RC5 benötigt bei einer Schlüssellänge von 80 Bit pro Byte $32.5\mu s$. Skipjack mit einer Schlüssellänge von 80 Bit benötigt für die Verschlüsselung eines Bytes $47.5\mu s$. In [64] werden verschiedene Berechnungen auf der MICA2-Plattform vorgenommen und deren Laufzeit und Energieverbrauch gemessen. Aus Laufzeit und Energieverbrauch lässt sich errechnen, dass die Leistung eines MICA2 Motes mit 8 MHz Taktfrequenz bei ca. $23.9mW$ liegt. Legt man diesen Wert zu Grunde, so lässt sich damit errechnen, dass RC5 bei einer Schlüssellänge von 80 Bit pro Byte $0.77\mu J$ Energie verbraucht. Skipjack benötigt für die Verschlüsselung eines Bytes $1.14\mu J$. Für TinySec als Ganzes wird ein Speicher-Overhead von 728 Byte RAM und 7146 Byte Flash-Speicher angegeben. Da Skipjack und RC5 beide in TinySec eingesetzt werden, arbeiten diese Algorithmen also mit höchstens dem gleichen Speicher-Overhead.

In [64] wird die Leistungsfähigkeit von TinySec im Hinblick auf den Energieverbrauch analysiert. Die Autoren stellen fest, dass eine mit TinySec geschützte 29 Byte lange Nachricht lediglich zusätzliche $1244\mu s$ für die Übertragung benötigt. Eine Verschlüsselung mit TinySec benötigt auf einem MICA2 Mote bei einer Taktfrequenz von $7.3828MHz$ $75,51\mu s$ pro Byte. Damit werden $1.8\mu J$ an Energie verbraucht. Weiterhin wurde festgestellt, dass die Implementierung von TinySec 730 Byte RAM und 7076 Bytes Flash-Speicher benötigt. Da Skipjack und RC5 Bestandteile von TinySec sind, und in TinySec keine Programmteile dynamisch nachgeladen oder ausgelagert werden, sind diese Zahlen eine Obergrenze für den Speicher-Overhead von Skipjack und RC5.

In [59] werden verschiedene symmetrische Verfahren für deren Verwendung in Sensornetzen evaluiert. Die betrachteten Verfahren sind Skipjack [99], RC5 [83], RC6 [80], Rijndael [30], Twofish[91], MISTY [66], KASUMI [4] und Camellia [7]. Bereits

im Vorfeld der Untersuchung wurden die Algorithmen DES [70], IDEA [58], SAFER++ [65], MARS [21] und Serpent [6] von der Untersuchung ausgeschlossen, da sie von den Autoren als nicht für Sensornetze geeignet befunden wurden. Für die Untersuchungen wurden die EYES Sensorknoten [44] eingesetzt. Diese Sensorknoten verfügen über 60 KB Flash Speicher und 2 KB RAM. Der 16-Bit Prozessor ist mit 8 MHz getaktet. Von der Rechenleistung und der Ausstattung mit RAM und Flash sind die EYES Sensorknoten durchaus mit den MICA Sensorknoten vergleichbar, allerdings verwenden die EYES Sensorknoten einen wesentlich energieeffizienteren Mikrokontroller, weswegen die gemessenen Energiewerte mit anderen Arbeiten, die z. B. die MICA Sensorknoten einsetzen, nicht direkt vergleichbar sind. Die Arbeit vergleicht den Flash-Speicher, der von den einzelnen Chiffre-Implementierungen benötigt wird. Die Werte bewegen sich für alle Algorithmen im Bereich von wenigen Kilobyte ($0.85kB$ bis $29.5kB$). So benötigt eine geschwindigkeitsoptimierte Version des Rijndael z. B. $4.5kB$ Flash-Speicher. Im direkten Vergleich der einzelnen Algorithmen stellen die Autoren fest, dass sich die Algorithmen Skipjack, Rijndael und MYSTY1 am besten für Sensornetze eigenen.

Zusammenfassung

Die Ergebnisse zeigen, wie sich die einzelnen Implementierungen in ihrer Effizienz unterscheiden. Für symmetrische Verschlüsselungs-Algorithmen wurden Laufzeiten von $32.5\mu s$ bis $287.5\mu s$ pro verschlüsseltes Byte gemessen. Dabei wurden für die Verschlüsselung eines Bytes zwischen $0.77\mu J$ und $2.9\mu J$ Energie benötigt. Die genauen Werte sind in Tabelle C.1 noch einmal aufgeführt.

Im Folgenden wird betrachtet, inwiefern sich asymmetrische Verfahren (so genannte Public-Key-Algorithmen) für Sensornetze eignen. Public-Key-Algorithmen können z. B. zur Verschlüsselung zum Einsatz kommen. Ebenfalls ist es möglich, asymmetrische Verfahren nur zum Schlüsselaustausch einzusetzen (z. B. über den Diffie-Hellman-Algorithmus [53]) um den ausgetauschten Schlüssel anschließend mit einer symmetrischen Chiffre zu benutzen (hybrides Verfahren). Schließlich können asymmetrische Verfahren auch zum Signieren von Nachrichten verwendet werden bzw. zum Verifizieren einer Signatur. In frühen Arbeiten zur Sicherheit in Sensornetzen wie z. B. [23, 73, 74] wird davon ausgegangen, dass asymmetrische Chiffren wie z. B. RSA [81] in Sensornetzen nicht praktikabel sind, da sie im Vergleich zu symmetrischen Verfahren wesentlich aufwendiger sind und konkrete Implementierungen an den äußerst begrenzten Ressourcen typischer Sensorknoten scheitern. Aktuelle Arbeiten zeigen jedoch, dass asymmetrische Algorithmen durchaus auch auf Sensorknoten realisierbar sind. Dabei kommt üblicherweise ein hybrides Verfahren zum Einsatz.

In [64] werden Implementierungen asymmetrischer Chiffren auf einem MICA2 Sensorknoten betrachtet, der mit $7.3828MHz$ getaktet ist. Für die in Diffie-Hellman benutzte Potenzierung $2^x \ mod \ p$ bei einem Exponent von 160 Bit und einer Primzahl p mit 1024 Bit dauert die Berechnung im Durchschnitt $54,9s$ und benötigt 1.185

Alg.	Parameter	Laufzeit [μs/Byte]	Energieverbrauch [μJ/Byte]	RAM	Flash
AES	MICADot @4MHz K:128 [109]	E:117 D:180	E:1.62 D:2.49	<4 KB	<128 KB
RC5	MICA2 @8MHz K:80 [54]	E/D:32.5	E/D:0.77	<728 B	<7146 B
Skipjack	MICA2 @8MHz K:80 [54]	E/D:47.5	E/D:1.14	<728 B	<1467 B
Skipjack	EYES @8MHz K:80 [59]	E/D:<69	E/D:<0.7	<2 KB	<2610 B
RC5	EYES @8MHz K:128 [59]	E/D:<150	E/D:<1.512	<2 KB	<7502 B
RC6	EYES @8MHz K:128 [59]	E/D:<287.5	E/D:<2.9	<2 KB	<3174 B
AES	EYES @8MHz K:128 [59]	E/D:<131	E/D:<1.32	<2 KB	<9984 B
Twofish	EYES @8MHz K:128 [59]	E/D:<162.5	E/D:<1.64	<2 KB	<12538 B
MISTY	EYES @8MHz K:128 [59]	E/D:<62.5	E/D:<0.63	<2 KB	<8596 B
KASUMI	EYES @8MHz K:128 [59]	E/D:<69	E/D:<0.7	<2 KB	<11078 B
Camellia	EYES @8MHz K:128 [59]	E/D:<225	E/D:<2.27	<2 KB	<29516 B

Tabelle C.1: Zusammenfassung Performance symmetrische Chiffren. Dabei steht K für die Schlüssellänge in Bit, E für Verschlüsselung und D für Entschlüsselung. Die Werte sind in Byte (B) oder Kilobyte (KB) angegeben.

J Energie. Die Parameter wurden so gewählt, dass die Sicherheit der Verschlüsselung äquivalent ist zu einer symmetrischen Verschlüsselung mit einem Schlüssel von 80 Bit Länge (man spricht in diesem Fall auch von „80 Bit Sicherheit“, siehe [18]). Die oben angesprochenen Potenzierung stellt im Diffie-Hellman-Algorithmus, neben der Kommunikation den größten Aufwand dar, weswegen die angegebenen Zahlen geeignet ist, um den Aufwand des Diffie-Hellman-Verfahrens zum Schlüsselaustausch abzuschätzen. Die Implementierung des Algorithmus benötigt ca. 1250 Bytes RAM und ca. 11 kB Flash.

Neben der traditionellen Diffie-Hellman-Implementierung wurde in [64] auch eine Implementierung des Algorithmus betrachtet, die auf Elliptischen Kurven [40] basiert. Bei dieser Implementierung kommt ein signifikant kürzerer Schlüssel mit nur 163 Bit zum Einsatz. Es wird ein ähnliches Sicherheitsniveau wie oben erreicht. Für die Erzeugung des benötigten Schlüsselpaars, bestehend aus öffentlichem und geheimem Schlüssel, benötigt die Implementierung im Durchschnitt 34.161 Sekunden. Die Energiekosten für die Erzeugung dieses Schlüsselpaars werden dabei mit $821.49mJ$ angegeben. Um bei Kenntnis eines öffentlichen Schlüssels eines Kommunikationspartners einen Schlüssel herzustellen werden 34.173 Sekunden benötigt. Der entsprechende Energieverbrauch beläuft sich auf $821.78mJ$. Die Implementierung benötigt ca. 1140 Bytes RAM und 33,5 kB Flash-Speicher

In [75] werden die Performance der Public Key Algorithmen RSA [81] und eines Signaturverfahrens auf Basis von elliptischen Kurven für verschiedene Plattformen untersucht. Für MICA2 Sensorknoten wurde errechnet, dass für die Signaturerzeugung mittels RSA mit einem 1024 Bit Schlüssel 359.87 mJ benötigt werden und diese Generierung 12.04 Sekunden dauert. Die Verifikation einer Signatur benötigt 14.05 mJ und dauert 0.47 s. Wird ein Schlüssel mit einer Schlüssellänge von 2048 Bit eingesetzt, so benötigt die Erzeugung einer Signatur 2725.66 mJ und die Verifikation der Signatur benötigt 63.55 mJ und dauert 2.13 Sekunden. Der entsprechenden Algorithmus mit elliptischen Kurven benötigt bei 160 Bit Schlüssellänge zur Generierung einer Signatur 26.96 mJ. Die Generierung der Signatur dauert 0.89 Sekunden. Die Überprüfung der Signatur benötigt 53.42 J und dauert 1.77 Sekunden. Bei einem Schlüssel mit 224 Bit werden bei dem gleichen Algorithmus für die Generierung der Signatur 72.85 mJ benötigt und für die Verifikation 144.40 J. Die Generierung einer Signatur dauert 2.41 Sekunden und die Überprüfung ist in 4.78 Sekunden erledigt. Andere Arbeiten wie z. B. [38, 109–111] kommen zu ähnlichen Ergebnissen in der gleichen Größenordnung.

In [18] wird ein Vergleich von Schlüssellängen von symmetrischen und asymmetrischen Verfahren mit gleichem Sicherheitsniveau gegeben. Dabei zeigt sich deutlich, dass asymmetrische Verfahren eine deutlich größere Schlüssellänge benötigen, um auf das gleiche Sicherheitsniveau wie vergleichbare symmetrische Chiffren zu kommen. So entspricht einem 128-Bit Schlüssel eines symmetrischen Verfahrens z. B. ein 3214-Bit langer RSA-Schlüssel bzw. ein 256-Bit langer ECC-Schlüssel. Da bei Sicherheitsprotokollen oft Schlüssel auf Sensorknoten gespeichert werden müssen ist es wichtig, dass diese so kurz wie möglich sind, da der Speicher der Sensorknoten

Alg.	Para-meter	Lauf-zeit $[ms]$	Energie-verbrauch $[mJ]$	RAM	Flash
RSA	MICA @7.4MHz K:1024 [64]	P:54900	1186	11 KB	1250 B
ECC	MICA @7.4MHz K:163 [64]	G:34161 P:34173	G: 821.49 P:821.78	33.5 KB	1140 B
RSA	MICA @7.4MHz K:1024 [75]	S:12040 V:470	S:359.87 V:14.05	<4KB	<128KB
RSA	MICA @7.4MHz K:2048 [75]	S:91180 V:2130	S:2725.66 V:63.55	<4KB	<128KB
ECC	MICA @7.4MHz K:160 [75]	S:890 V:1770	S:26.96 V:53.42	<4KB	<128KB
ECC	MICA @7.4MHz K:224 [75]	S:2410 V:4780	S:72.85 V:144.40	<4KB	<128KB

Tabelle C.2: Zusammenfassung Performance asymmetrischer Verfahren. Dabei steht K für die Schlüssellänge in Bit, S für die Operation Signieren und V für Operation Verifizieren. G steht für die Erzeugung der Schlüsselpaare. P steht für die Potenzierung bzw. Punktmultiplikation, was bei Diffie-Hellman den größten Teil der Berechnungen darstellt. Die Werte sind in Byte (B) oder Kilobyte (KB) angegeben.

sehr beschränkt ist. Weiterhin werden in Sicherheitsprotokollen Schlüssel übertragen. Eine größere Schlüssellänge bewirkt hier, dass längere Nachrichten übertragen werden müssen und somit der Energieverbrauch steigt.

Zusammenfassung Zusammenfassend kann man sagen, dass die Energiekosten für das Erzeugen einer digitale Signatur mittels eines asymmetrischen Verfahrens auf einem MICA2 Sensorknoten zwischen $26.96mJ$ und $2725.66mJ$ betragen. Die Überprüfung einer Signatur benötigt zwischen $14.05mJ$ und $144.40mJ$. Dabei dauert die Signaturerstellung zwischen $890ms$ und $12040ms$ und die Signaturüberprüfung zwischen $470ms$ und $4780ms$. Die Ergebnisse sind in Tabelle C.2 noch einmal aufgeführt.

Dieser Abschnitt hat gezeigt, dass asymmetrische Verfahren deutlich aufwendiger als symmetrische Verfahren sind. Auch die deutlich größere Schlüssellänge bei gleichem Sicherheitsniveau spricht gegen den Einsatz von Public-Key-Algorithmen in Sensornetzen. Es muss also genau abgewogen werden, ob sich der zusätzliche Aufwand beim Einsatz eines Public-Key-Verfahrens rechnet. Da damit zu rechnen ist,

dass zukünftige Generationen von Sensorknoten ohnehin die symmetrische Chiffre AES in Hardware mitbringen (siehe oben) bietet es sich an, symmetrische Chiffren den asymmetrischen Verfahren vorzuziehen.

Basierend auf den oben besprochenen Untersuchungen wird in dieser Arbeit davon ausgegangen, dass beim Entwurf eines Verfahrens zum sicheren Auffinden von Diensten in Sensornetzen lediglich auf symmetrische Chiffren wie z. B. AES zurückgegriffen werden sollte.

C.3 Hash-Funktionen

In [37] wird die Leistungsfähigkeit der Hash-Algorithmen MD5 und SHA-1 auf verschiedenen Mikrokontrollern untersucht. Unter anderem wird auch der Atmel ATmega 128 Mikrokontroller betrachtet, der auf den in dieser Arbeit betrachteten Sensorknoten zum Einsatz kommt. Allerdings wird dieser Prozessor in der Untersuchung mit $16MHz$ getaktet und ist im Vergleich zur standardmäßigen Ausstattung der MICAz bzw. MICA2 Motes, die vielen der hier vorgestellten Arbeiten zugrunde liegen, doppelt so schnell.

Für SHA-1 wird die Laufzeit bei Eingabe von einem Byte als $3.8ms$ angegeben. Dies ist als obere Schranke zu verstehen, da SHA-1 eine Blocklänge von 64 Byte hat und das eine Byte Eingabe auf die Blocklänge erweitern muss. Für eine Eingabe von 64 Byte benötigt SHA-1 $7.8ms$, also pro Byte $122\mu s$. Bei MD5 ergibt sich entsprechend eine Laufzeit von $34\mu s$. MD5 benötigt etwas mehr als $10kB$ Flash-Speicher, während SHA-1 lediglich $4kB$ Flash Speicher benötigt.

In einer späteren Arbeit [109] wird der Energieverbrauch des Hash-Algorithmus SHA-1 gemessen. Pro Eingabe-Byte werden gemittelt $5.9\mu J$ Energie verbraucht. Mit der in der Arbeit angegebenen Leistung von $13.8mW$ errechnet sich daraus eine Laufzeit von $427.5\mu s$ pro Eingabebyte.

Zusammenfassung

Die untersuchten Implementierungen von Hash-Funktion haben eine Laufzeit zwischen $427.5\mu s$ und $3812\mu s$ pro Eingabebyte bei einem Energieverbrauch von $5.9\mu J$.

C.4 Message Authentication Codes

Message Authentication Codes dienen zur Sicherung der Integrität von Nachrichten sowie zur Authentifizierung des Absenders. Generell ist es möglich, aus Hash-Funktionen einen Algorithmus zur Erzeugung von Message Authentication Codes aufzubauen, siehe z. B. HMAC [56], wobei die zusätzlich verwendeten Operationen relativ einfach sind. Es ist also zu erwarten, dass sich Message Authentication Codes für Sensornetze eignen.

In [64] wird die Leistungsfähigkeit von TinySec untersucht. In TinySec kommt ein Message Authentication Code zum Schutz der Integrität von Nachrichten auf der

Alg.	Parameter	Lauf-zeit [μs/Byte]	Energieverbrauch [μJ/Byte]	RAM	Flash
SHA-1	MICADot @4MHz [109]	427.5	5.9	<4 KB	<128 KB
SHA-1	MICA2 @16MHz [37]	<3812	-	<4 KB	4 KB
MD5	MICA2 @16MHz [37]	<1466	-	<4 KB	10 KB

Tabelle C.3: Zusammenfassung Performance Hash-Funktionen

Sicherungsschicht zum Einsatz. Die Berechnung eines Message Authentication Code für eine Nachricht mit 29 Byte Nutzdaten benötigt $3049\mu s$ und $72,87\mu J$. Pro Byte werden also $105.14\mu s$ und $2.51\mu J$ benötigt.

Zusammenfassung

Message Authentication Codes können also beim Entwurf eines Verfahrens zum sicheren Auffinden von Diensten in Sensornetzen verwendet werden.

C.5 Zusammenfassende Bewertung kryptographischer Bausteine

Im Rahmen dieses Kapitels wurden untersucht, inwiefern verschiedene kryptographische Verfahren in Sensornetzen eingesetzt werden können. Es stellte sich heraus, dass symmetrische Verschlüsselungs-Verfahren, Hash-Funktionen und Message Authentication Codes mit relativ geringem Aufwand eingesetzt werden können, und damit bei geeignetem Einsatz insbesondere die Anforderung „Energie-Effizienz“ erfüllen. Der Einsatz von asymmetrischen Verschlüsselungs-Verfahren und Digitalen Signaturen kann in Sensornetzen nur eingeschränkt erfolgen, da diese Verfahren auf Sensorknoten im Vergleich zu symmetrischen Verfahren wesentlich mehr Rechenzeit erfordern und entsprechend auch deutlich mehr Energie benötigen. Um das gleiche Sicherheitsniveau wie bei einer symmetrischen Chiffre zu erreichen ist zudem bei asymmetrischen Verfahren ein wesentlich längerer Schlüssel notwendig. Werden Schlüssel übertragen, kann dies zu längeren Nachrichten führen. Asymmetrische Verfahren erfüllen die Anforderung „Energie-Effizienz“ also nicht.

Literatur

[1] *Crossbow Homepage*. http://www.xbow.net

[2] *TinyOS Documentation*. http://www.tinyos.net/tinyos-1.x/doc/

[3] *ZigBee Alliance Homepage*. http://www.zigbee.org/

[4] 3rd Generation Partnership Project: Specification of the 3GPP Confidentiality and Integrity Algorithms - Document 2: KASUMI Specification (Release 6). URL http://www.3gpp.org/ftp/Specs/archive/35_series/35.202/35202-610.zip, September 2005 (3GPP TS 35.202 V6.1.0 (2005-09)). – Forschungsbericht

[5] Almenarez, Florina ; Campo, Celeste: SPDP: A Secure Service Discovery Protocol for Ad-hoc Networks. In: Halász, Edit (Hrsg.): *Proceedings of EUNICE 2003, 9th EUNICE Open European Summer School and IFIP WG6.3 Workshop on Next Generation Networks*. Balatonfüred, Ungarn, September 2003. – ISBN 963-421-576-9

[6] Anderson, R. ; Biham, E. ; Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. In: *Proceedings of the First AES Candidate Conference*. Ventura, CA, USA, Juni 1998

[7] Aoki, K. ; Ichikawa, T. ; Kanda, M. ; Matsui, M. ; Moriai, S. ; Nakajima, J. ; Tokita, T.: Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. In: Stinson, D.R. (Hrsg.) ; Tavares, S.E. (Hrsg.): *Selected Areas in Cryptography (SAC 2000)* Bd. 2012/2001. Waterloo, Ontario, Kanada : Springer Berlin/Heidelberg, August 2000, S. 39–56. – ISSN 0302-9743

[8] Bagrodia, G. ; Meyer, R. ; Takai, M. ; Chen, Y. ; Martin, J. ; Song, H.: Parsec: A Parallel Simulation Environment for Complex Systems. In: *IEEE Computer* 31 (1998), Oktober, S. 77–85

[9] Bajaj, Lokesh ; Takai, Mineo ; Ahuja, Rajat ; Tang, Ken ; Begrodia, Rajive ; Gerla, Mario: GloMoSim: A Scalable Network Simulation Environment / UCLA Computer Science Department. Los Angeles, CA, USA, 1999 (Nr. 990027). – Forschungsbericht

[10] Baldoni, R. ; Beraldi, R. ; Mian, A.: Survey of Service Discovery Protocols in Mobile Ad Hoc Networks / Dip. Informatica e Sistemistica Antonio Ruberti, Universita di Roma La Sapienza. Rom, Italien, Juni 2006 (Midlab 7/06). – Forschungsbericht. – URL http://www.dis.uniroma1.it/~midlab

[11] Balfanz, Dirk ; Smetters, D. K. ; Stewart, Paul ; Wong, H. C.: Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In: *In Proceedings of Network and Distributed System Security Symposium 2002*. San Diego, CA, USA : Internet Society, Februar 2002. – ISBN 1-891562-14-2

[12] Baumgart, Ingmar: *Analyse, Optimierung und Evaluierung eines sicheren verteilten Dienstverzeichnisses für drahtlose Sensornetze*. Karlsruhe, Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Diplomarbeit, April 2005

[13] Becher, Alexander ; Benenson, Zinaida ; Dornseif, Maximillian: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In: *Proceedings of International Conference on Security in Pervasive Computing*. York, UK : Springer Berlin/New York, April 2006. – ISBN 3540333762

[14] Bechler, Marc ; Hof, Hans-Joachim ; Kraft, Daniel ; Pählke, Frank ; Wolf, Lars: A Cluster-Based Security Architecture for Ad Hoc Networks. In: *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*. Hong Kong, China : IEEE Computer Society, März 2004, S. 2393 – 2403

[15] Blakley, Robert ; Kabatiansky, Gregory: *Encyclopedia of Cryptography and Security*. Kap. Secret Sharing Shemes, S. 544–545, Springer, 2005. – ISBN 0-387-23473-X

[16] Blass, Erik-Oliver ; Hof, Hans-Joachim ; Zitterbart, Martina: S-CAN: Sicheres Overlay fuer Sensornetze. In: *2. GI/ITG KuVS Fachgespraech Drahtlose Sensornetze*. Karlsruhe, Deutschland, März 2004. – URL http://doc.tm.uka.de/2004/blass-fach2004.pdf

[17] Bless, Roland ; Mink, Stefan ; Blass, Erik-Oliver ; Conrad, Michael ; Hof, Hans-Joachim ; Kutzner, Kendy ; Schoeller, Marcus ; Bless, Roland (Hrsg.): *Sichere Netzwerkkommunikation*. Springer-Verlag Berlin/Heidelberg, Juni 2005 (X.system.press). – ISBN 3-540-21845-9

[18] Bless, Roland ; Mink, Stefan ; Blass, Erik-Oliver ; Conrad, Michael ; Hof, Hans-Joachim ; Kutzner, Kendy ; Schoeller, Marcus: *Sichere Netzwerkkommunikation*. Kap. 3.8.1 Schlüssellängen, S. 89. In: Bless, Roland (Hrsg.): *Sichere Netzwerkkommunikation*, Springer-Verlag Berlin/Heidelberg, Juni 2005 (X.system.press). – ISBN 3-540-21845-9

[19] Bluetooth SIG (Veranst.): *Bluetooth Specification Version 2.0*. November 2004. – URL http://german.bluetooth.com/NR/rdonlyres/1F6469BA-6AE7-42B6-B5A1-65148B9DB238/840/Core_v210_EDR.zip

[20] Boeyen, Sharon: *Encyclopedia of Cryptography and security.* Kap. Trust Models, S. 628–637, Springer, 2005. – ISBN 0-387-23473-X

[21] Burwick, C. ; Coppersmith, D. ; Avignon, E.D. ; Gennaro, R. ; Halevi, S. ; Jutla, C. ; Jr., S.M. M. ; O'Connor, L. ; Peyravian, M. ; Safford, D. ; Zunic, N.: MARS–a candidate cipher for AES. In: *Proceedings of the First AES Candidate Conference.* Ventura, CA, USA, Juni 1998

[22] Campo, C. ; Munoz, M. ; Perea, J.C. ; A.Mann ; Garcia-Rubio, C.: PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems. In: *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops).* Kauai Island, Hawaii, USA : IEEE Computer Society, März 2005, S. 178–182. – ISBN 0-7695-2300-5

[23] Carman, David W. ; Kruus, Peter S. ; Matt, Brian J.: Constraints and Approaches for Distributed Sensor Network Security (Final) / Network Associates Inc. Glenwood, MD, USA, September 2000 (Nr. 00-010). – NAI Labs Technical Report

[24] Chen, Bor-Rong ; Muniswamy-Reddy, Kiran-Kumar ; Welsh, Matt: Ad-hoc multicast routing on resource-limited sensor nodes. In: *International Symposium on Mobile Ad Hoc Networking & Computing: Proceedings of the second international workshop on Multi-hop ad hoc networks: from theory to reality.* Florenz, Italien : ACM Press, Mai 2006, S. 87 – 94. – ISBN 1-59593-360-3

[25] Clark, D.: The design philosophy of the DARPA internet protocols. In: *ACM SIGCOMM Computer Communication Review* Bd. 18, August 1988, S. 106–114. – ISBN 0-89791-279-9

[26] Cramer, Ronald ; Damgard, Ivan: *Contemporary Cryptology (Advanced Courses in Mathematics - CRM Barcelona).* Kap. Multiparty Computation, an Introduction, S. 41–88, Birkhäuser, Juli 2005. – ISBN 3-7643-7294-X

[27] Cronin, Eric: *Encyclopedia of Cryptography and security.* Kap. "Denial of Service", S. 143–144, Springer, 2005. – ISBN 0-387-23473-X

[28] Crossbow Technology Inc.: *Mica2Dot - Wireless Microsensor Mote (Data Sheet).* – URL http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf

[29] Czerwinski, Steven E. ; Zhao, Ben Y. ; Hodes, Todd D. ; Joseph, Anthony D. ; Katz, Randy H.: An architecture for a secure service discovery service. In: *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking.* Seattle, Washington, USA : ACM Press New York, August 1999, S. 24–35. – ISBN 1-58113-142-9

[30] Daemen, Joan ; Rijmen, Vincent: *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag New York, März 2002. – ISBN 3-540-425580-2

[31] Dolev, D. ; Yao, A.: On the Security of Public Key Protocols. In: *IEEE Transactions on Information Theory* 29 (1983), März, Nr. 2, S. 198–208. – ISSN 0018-9448

[32] Douceur, John R.: The Sybil Attack. In: *IPTPS '01:Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Cambridge, MA, USA : Springe Berlin/New York, März 2002

[33] Eck, Holger: *Adaptives Energiemanagement in Sensorknoten*. Karlsruhe, Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Studienarbeit, 2006

[34] Eckert, Claudia: *IT-Sicherheit*. 3.Auflage. Oldenburg Verlag München/Wien, 2004. – ISBN 3-486-20000-3

[35] Edwards, W. K.: Discovery Systems in Ubiquitous Computing. In: *Pervasive Computing* 5 (2006), April-Juni, Nr. 2, S. 70–77

[36] Frank, Christian ; Handziski, Vlado ; Karl, Holger ; Wolisz, Prof. Dr.-Ing. A. (Hrsg.): Service Discovery in Wireless Sensor Networks / Technical University Berlin, Telecommunication Networks Group. Berlin, Deutschland, März 2004 (TKN-04-006). – TKN Technical Report

[37] Ganesan, P. ; Venugopalan, R. ; Peddabachagari, P. ; Dean, A. ; Mueller, F. ; Sichitiu, M.: Analyzing and Modeling Encryption Overhead for Sensor Network Nodes. In: *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. San Diego, CA, USA : ACM Press New York, 2003, S. 151–159. – ISBN 1-58113-764-8

[38] Gura, Nils ; Pate, Arun ; Wander, Arvinderpal ; Eberle, Hans ; Shantz, Sheueling C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: *Cryptographic Hardware and Embedded Systems - CHES 2004* Bd. 3146/2004. Cambridge, MA, USA : Springer Berlin/Heidelberg, Juli 2004, S. 119–132. – ISBN 978-3-540-22666-6

[39] Guttman, E. ; Perkins, C. ; Veizades, J. ; Day, M.: Service Location Protocol, Version 2 / Internet Engineering Task Force (IETF). RFC Editor, Juni 1999 (RFC 2608). – RFC

[40] Hankerson, Darrel ; Menezes, Alfred: *Encyclopedia of Cryptography and security*. Kap. "Elliptic Curve", S. 183–197, Springer, 2005. – ISBN 0-387-23473-X

[41] Helal, S. ; Desai, N. ; Verma, V. ; Choonhwa, Lee: Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks. In: *Wireless Communications and Networking, 2003 (WCNC 2003)* Bd. 3. New Orleans, USA : IEEE Computer Society, März 2003, S. 2107– 2113. – ISBN 0-7803-7700-1

[42] Hermann, R. ; Husemann, D. ; Moser, M. ; Nidd, M. ; Rohner, C. ; Schade, A.: DEAPspace–Transient Ad hoc Networking of Pervasive Devices. In: *Computer Networks* 35 (2001), Februar, Nr. 4, S. 411–428

[43] Hildrum, K. ; Kubiatowicz, J.D. ; Rao, S. ; Zhao, B.Y.: Distributed Object Location in a Dynamic Network. In: *SPAA '02: Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures.* Winnipeg, Manitoba, Kanada : ACM Press New York, August 2002, S. 41–52

[44] Hoesel, L. V. ; Dulman, S. ; Havinga, P. ; Kip, H.: Design of a low-power testbed for wireless sensor networks and verification / Centre for Telematics and Information Technology, University of Twente, The Netherlands. Twente, Niederlande, September 2003 (TR-CTIT-03-45). – Forschungsbericht

[45] Hof, Hans-Joachim ; Blass, Erik-Oliver ; Fuhrmann, Thomas ; Zitterbart, Martina: Design of a Secure Distributed Service Directory for Wireless Sensor Networks. In: Karl, Holger (Hrsg.) ; Willig, Andreas (Hrsg.) ; Wolisz (Hrsg.): *Proceedings of First European Workshop on Wireless Sensor Networks* Bd. 2920/2004. Berlin, Deutschland : Springer, Januar 2004, S. 276–290. – ISBN 978-3-540-20825-9

[46] Hof, Hans-Joachim ; Blass, Erik-Oliver ; Zitterbart, Martina: Secure Overlay for Service Centric Wireless Sensor Networks. In: Castelluccia, Claude (Hrsg.) ; Hartenstein, Hannes (Hrsg.) ; Paar, Christof (Hrsg.) ; Westhoff, Dirk (Hrsg.): *Security in Ad-hoc and Sensor Networks: First European Workshop, ESAS 2004.* Heidelberg, Deutschland : Springer-Verlag, August 2004. – ISBN 3-540-243-96-8

[47] Hof, Hans-Joachim ; Hurler, Bernhard ; Hergenröder, Anton ; Haas, Christian ; Conrad, Michel: *Programming and Securing Service-oriented Wireless Sensor Networks.* 5th International Conference on Mobile Systems, Applications, and Services (MobiSys 2007), Puerto Rico, USA. Juni 2007. – Demo Abstract

[48] Hof, Hans-Joachim ; Zitterbart, Martina: SCAN: A secure service directory for service-centric wireless sensor networks. In: *Computer Communications* 28 (2005), Juli, Nr. 13, S. 1517–1522. – ISSN 0140-3664

[49] Hurler, Bernhard: *Dienstorientierte Sensornetze: Programmierabstraktionen und Dienstkomposition.* Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Dissertation (in Arbeit), erscheint voraussichtlich 2008

[50] Intanagonwiwat, C. ; Govindan, R. ; Estrin, D. ; Heidemann, J. ; Silva, F.: Directed Diffusion for Wireless Sensor Networking. In: *IEEE /ACM Transactions on Networking* 11 (2003), Februar, Nr. 1, S. 2–16

[51] Jacquet, P. ; Muhlethaler, P. ; Clausen, T. ; Laouiti, A. ; Qayyum, A. ; Viennot, L.: Optimized link state routing protocol for ad hoc networks. In: *Proceedings of IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.* Lahore, Pakistan : IEEE Computer Society, Dezember 2001, S. 62–68

[52] Ji, Ping ; Ge, Zihui ; Kurose, Jim ; Towsley, Don: A comparison of hard-state and soft-state signaling protocols. In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* Pittsburgh, Pennsylvania, USA : ACM Press New York, August 2003, S. 251–262. – ISBN 1-58113-735-4

[53] Just, Mike: *Encyclopedia of Cryptography and security.* Kap. "Diffie-Hellman Key Agreement", S. 154–158, Springer, 2005. – ISBN 0-387-23473-X

[54] Karlof, Chris ; Sastry, Naveen ; Wagner, David: TinySec: a link layer security architecture for wireless sensor networks. In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems.* Baltimore, Maryland, USA : ACM Press New York, November 2004, S. 162–175. – ISBN 1-58113-879-2

[55] Klein, M. ; Konig-Ries, B. ; Obreiter, P.: Service rings - a semantic overlay for service discovery in ad hoc networks. In: Marik, Vladimir (Hrsg.) ; Retschitzegger, Werner (Hrsg.) ; Stepankova, Olga (Hrsg.): *Database and Expert Systems Applications, 14th International Conference, DEXA 2003, Proceedings* Bd. 2003/2736. Prag, Tschechien : Springer Berlin/Heidelberg/-New York, September 2003, S. 180–185. – ISBN 0-7695-1993-8

[56] Krawczyk, H. ; Bellare, M. ; Canetti, R.: HMAC: Keyed-Hashing for Message Authentication / Internet Engineering Task Force (IETF). Februar 1997 (RFC 2104). – Forschungsbericht

[57] Kuntz, Andreas ; Zitterbart, Martina: ServiceCast: Eine Architektur zur dienstorientierten Kommunikation in selbstorganisierenden Sensor-Aktor-Netzen. In: *6. Fachgespräch Sensornetze der GI/ITG Fachgruppe Kommunikation und Verteilte Systeme.* Aachen, Deutschland : Distributed Systems Group, RWTH Aachen University, Juli 2007, S. 39–42. – ISSN 0935-3232

[58] Lai, Xuejia ; Massey, James L. ; Murphy, Sean: Markov Ciphers and Differential Cryptanalysis. In: *Advances in Cryptology - EUROCRYPT '91: Workshop on the Theory and Application of Cryptographic Techniques* Bd. 547/1991. Brighton, UK : Springer Berlin/Heidelberg, April 1991, S. 17–39. – ISSN 0302-9743

[59] Law, Yee W. ; Doumen, Jeroen ; Hartel, Pieter: Survey and Benchmark of Block Ciphers for Wireless Sensor Networks. In: *ACM Transactions on Sensor Networks (TOSN)* 2 (2006), Februar, S. 65–93. – ISSN 1550-4859

[60] Linga, Prakash ; Gupta, Indranil ; Birman, Ken: A Churn-resistant Peer-to-Peer Web Caching System. In: *Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems: in association with 10th ACM Conference on Computer and Communications Security.* Fairfax, VA, USA : ACM Press New York, 2003, S. 1 – 10. – ISBN 1-5811-3784-2

[61] Luo, Jun ; Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In: *Proceedings of the IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Bd. 3. Miami, CA, USA, März 2005, S. 1735–1746. – ISBN 0-7803-8968-9

[62] Mahlmann, Peter ; Schindelhauer, Christian: *Peer-to-Peer-Netzwerke.* Springer-Verlag Berlin Heidelberg, 2007 (eXamen.press). – ISBN 978-3-540-33991-5

[63] Mainwaring, Alan ; Culler, David ; Polastre, Joseph ; Szewczyk, Robert ; Anderson, John: Wireless sensor networks for habitat monitoring. In: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.* Atlanta, Georgia, USA : ACM Press, 2002, S. 88–97. – ISBN 1-58113-589-0

[64] Malan, D.J. ; Welsh, M. ; Smith, M.D.: A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In: *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks.* Santa Clara, CA, USA : IEEE Computer Society, Oktober 2004, S. 171–80. – ISBN 0-7803-8796-1

[65] Massey, J.L. ; Khachatrian, G.H. ; Kuregian, M.K.: Nomination of SAFER++ as candidate algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) / NESSIE. 2000. – Forschungsbericht

[66] Matsui, M.: Block Encryption Algorithm MISTY. In: Biham, E. (Hrsg.): *Fast Software Encryption (FSE 1997)* Bd. 1267. Haifa, Israel : Springer-Verlag, Januar 1997, S. 64–74. – ISBN 3-540-63247-6

[67] Meier, René ; Cahill, Vinny ; Nedos, Andronikos ; Clarke, Siobhán: Proximity-Based Service Discovery in Mobile Ad Hoc Networks. In: Kutvonen, Lea (Hrsg.) ; Alonistioti, Nancy (Hrsg.): *Distributed Applications and Interoperable Systems* Bd. 3543/2005. Athen, Griechenland : Springer Berlin/Heidelberg, Juni 2005, S. 115–129. – ISBN 978-3-540-26262-6

[68] Microsoft: *Understanding UPnP: A White Paper.* http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc. 2000. – Redmond, USA

[69] Moore, Gordon E.: Cramming More Components onto Integrated Circuits. In: *Electronics* 38 (1965), April, Nr. 8, S. 114–117

[70] NIST: Data Encryption Standard (DES) / National Institute of Standards and Technology. Januar 1979 (46). – FIPS Publication

[71] NIST: Secure Hash Standard / National Institute of Standards and Technology. April 1995 (180-1). – FIPS Publication

[72] Perkins, Charles E. ; Royer, Elizabeth M.: Ad-hoc On-Demand Distance Vector Routing. In: *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. New Orleans, Louisiana, USA : IEEE Computer Society, Februar 1999, S. 90–100

[73] Perrig, Adrian ; Stankovic, John ; Wagner, David: Security in wireless sensor networks. In: *Communications of the ACM* 47 (2004), Nr. 6, S. 53–57. – ISSN 0001-0782

[74] Perrig, Adrian ; Szewczyk, Robert ; Tygar, J. D. ; Wen, Victor ; Culler, David E.: SPINS: security protocols for sensor networks. In: *Wireless Networks* 8 (2002), September, Nr. 5, S. 521–534. – ISSN 1022-0038

[75] Piotrowski, Krzysztof ; Langendoerfer, Peter ; Peter, Steffen: How public key cryptography influences wireless sensor node lifetime. In: *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*. Alexandria, VA, USA : ACM Press New York, 2006, S. 169–176. – ISBN 1-59593-554-1

[76] Plötner, Johannes W. M.: *Implementierung zweier Schlüsselaustauschverfahren für Sensornetze*. Studienarbeit am Institut für Telematik, Universität Karlsruhe (TH), Deutschland. Februar 2006

[77] Ratnasamy, S. ; Francis, P. ; Handley, M. ; Karp, R. ; Shenker, S.: A Scalable Content Addressable Network. In: *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM '01* Bd. 31. San Diego, CA, USA : ACM Press New York, August 2001, S. 161–172

[78] Rice, John A.: *Mathematical Statistics and Data Analysis*. 2. Auflage. Wadsworth Group - Duxbury Press, 1995. – ISBN 0534209343

[79] Richard, Golden G.: *Service and Device Discovery*. Mcgraw-Hill Professional, 2002 (Telecom Developer). – ISBN 0-0713-7959-2

[80] Rivest, R. L. ; Robshaw, M.J.B ; R.Sidney ; Yin, Y.L.: The RC6 Block Cipher. In: *The First Advanced Encryption Standard (AES) Candidate Conference*. Ventura, CA, USA, August 1998

[81] Rivest, R. L. ; Shamir, A. ; Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. In: *Communications of the ACM* 21 (1978), Nr. 2, S. 120–126. – ISSN 0001-0782

[82] Rivest, R.L.: The MD5 Message-Digest Algorithm / Internet Engineering Task Force (IETF). April 1992 (RFC 1321). – Forschungsbericht

[83] Rivest, Ronald L.: The RC5 Encryption algorithm. In: Preneel, Bart (Hrsg.): *Proceedings Second International Workshop on Fast Software Encryption*. Leuven, Belgien : Springer Berlin/Heidelberg, 1995, S. 86–96

[84] Römer, K. ; Mattern, F.: The design space of wireless sensor networks. In: *IEEE Wireless Communications* 11 (2004), Dezember, Nr. 6, S. 54–61. – ISSN 1536-1284

[85] Rowstron, A. ; Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-Peer Systems. In: *Proceedings of the International Conference on Distributed Systems Platforms (IFIP/ACM)* Bd. 2218. Heidelberg, Deutschland : Springer Berlin/New York, November 2001, S. 329–350

[86] Sailhan, Francoise ; Issarny, Valerie: Scalable Service Discovery for MANET. In: *Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*. Los Alamitos, CA, USA : IEEE Computer Society, März 2005, S. 235–244. – ISBN 0-7695-2299-8

[87] Schiele, Gregor ; Becker, Christian ; Rothermel, Kurt: Energy-efficient cluster-based service discovery for Ubiquitous Computing. In: *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*. Leuven, Belgien : ACM Press New York, September 2004

[88] Schneier, Bruce: *Applied Cryptography*. John Wiley & Son, 1995. – ISBN 0-471-117-099

[89] Schneier, Bruce: Attack Trees - Modeling security threats. In: *Dr. Dobb's Journal* (1999), Dezember. – http://www.schneier.com/paper-attacktrees-ddj-ft.html

[90] Schneier, Bruce: *Secrets and Lies: Digital Security in a Networked World* . Kap. 21. Attack Trees, S. 318–333, Wiley, Januar 2004. – ISBN 0-4714-5380-3

[91] Schneier, Bruce ; Kelsey, J. ; Whiting, D. ; Wagner, D. ; Hall, C. ; Ferguson, N.: Twofish: A 128-bit Block Cipher. In: *Proceedings of the First AES Candidate Conference*, August 1998

[92] Simmons, G. J.: *Contemporary Cryptology - The Science of Information Integrity*. Kap. Chapter 9: An Introduction to Shared Secret and/or Shared Control Schemes and their Application, S. 441–497, IEEE Press New York, 1992. – ISBN 0-7803-5352-8

[93] Simon, Volker: *Sicherheit in Sensornetzen durch Einbeziehung von Kontext-Informationen.* Karlsruhe, Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Studienarbeit, 2007

[94] Singh, Atul ; Ngan, Tsuen-Wan J. ; Druschel, Peter ; Wallach, Dan S.: Eclipse Attacks on Overlays: Threats and Defenses. In: *Proceedings of the 25th Conference on Computer Communications (IEEE InfoCom 2006).* Barcelona, Spanien : IEEE Computer Society, April 2006

[95] Sit, Emil ; Morris, Robert: Security Considerations for Peer-to-Peer Distributed Hash Tables. In: Druschel, Peter (Hrsg.) ; Kaashoek, M. F. (Hrsg.) ; Rowstron, Antony I. T. (Hrsg.): *Peer-to-Peer Systems, First International Workshop, IPTPS 2002* Bd. 2429. Cambridge, MA, USA : Springer, März 2002. – ISBN 3-540-44179-4

[96] Srivatsa, M. ; Liu, L.: Vulnerabilities and Security Threats in Structured Overlay Networks: a Quantitative Analysis. In: *Annual Computer Security Applications Conference.* Miami, CA, USA : IEEE Computer Society, Dezember 2004, S. 252–261. – ISBN 0-7695-2252-1

[97] Stajano, Frank: *Security for Ubiquitous Computing.* John Wiley & Son, 2002 (Wiley series in Communications Networking & Distributed Systems). – 96 S. – ISBN 0-470-84493-0

[98] Stajano, Frank ; Anderson, R.J.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: *Security Protocols* Bd. 1796/2000, Springer Berlin/Heidelberg, April 1999, S. 172–182. – ISBN 978-3-540-67381-1

[99] Standards, National I. of ; (NIST), Technology: SKIPJACK and KEA Algorithm Specifications Version 2.0 / National Institute of Standards and Technology (NIST). Mai 1998. – Forschungsbericht

[100] Stinson, Douglas ; Wei, Ruizhong: *Bibliography on Secret Sharing Schemes.* http://www.cacr.math.uwaterloo.ca/ dstinson/ssbib.html. Oktober 1998

[101] Stoica, I. ; Morris, R. ; Karger, D. ; Kaashoek, F. ; Balakrishnan, H.: Chord: A scalable Peer-To-Peer Lookup Service for Internet Applications. In: *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)* Bd. 31. San Diego, CA, USA : ACM Press New York, 2001, S. 149–160

[102] Stutz, Daniel: *Service Cast, ein neues Kommunikationsparadigma für Sensor-Aktor-Netze.* Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Diplomarbeit, Juli 2007

[103] Sun Microsystems, Inc. (Veranst.): *The Jini Technology Surrogate Architecture Overview.* v1.0.1_002. 2001. – URL https://surrogate.dev.java.net/doc/overview.pdf

[104] Technology, Crossbow: MICAz WirelessM Measurement System / Crossbow Technology. URL http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf (Document Part Number: 6020-0060-04 Rev A). – Datenblatt

[105] Tilborg, Henk C. A. van (Hrsg.): *Encyclopedia of Cryptography and security*. Springer, 2005. – ISBN 0-387-23473-X

[106] Trabelsi, Slim ; Pazzaglia, Jean-Christophe R. ; Roudier, Yves: Enabling Secure Discovery in a Pervasive Environment. In: Clark, John A. (Hrsg.) ; Paige, Richard F. (Hrsg.) ; Polack, Fiona (Hrsg.) ; Brooke, Phillip J. (Hrsg.): *Third International Conference on Security in Pervasive Computing* Bd. 3934. York, UK : Springer, April 2006, S. 18–31. – ISBN 3-540-33376-2

[107] Uniband Electronics Corporation: *UZ2400 - Brief Data Sheet Rev. 0.2*, Oktober 2005. – URL http://www.dacomwest.de/pdf/uz2400_short.pdf. – Document Number BF-2400-01

[108] Ververidis, C.N. ; Polyzos, G.C.: Routing layer support for service discovery in mobile ad hoc networks. In: *Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*. Kauai Island, Hawaii, USA : IEEE Computer Society, März 2005, S. 258–262. – ISBN 0-7695-2300-5

[109] Wander, Arvinderpal S. ; Gura, Nils ; Eberle, Hans ; Gupta, Vipul ; Shantz, Sheueling C.: Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In: *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA : IEEE Computer Society, 2005, S. 324–328. – ISBN 0-7695-2299-8

[110] Wang, Haodong ; Sheng, Bo ; Li, Qun: Elliptic Curve Cryptography Based Access Control in Sensor Networks. In: *International Journal of Security and Networks* 1 (2006), Dezember, Nr. 3/4, S. 127–137. – ISSN 1747-8405

[111] Watro, Ronald ; Kong, Derrick ; Cuti, Sue fen ; Gardiner, Charles ; Lynn, Charles ; Kruus, Peter: TinyPK: securing sensor networks with public key technology. In: *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. Washington, DC, USA : ACM Press New York, 2004, S. 59–64. – ISBN 1-58113-972-1

[112] Weiser, M.: The Computer for the Twenty-First Century. In: *Scientific American* 265 (1991), Nr. 3, S. 94–104

[113] Wenyuan, Xu ; Ke, Ma ; Trappe, W. ; Yanyong, Zhang: Jamming sensor networks: attack and defense strategies. In: *IEEE Network* 20 (2006), Mai-Juni, Nr. 3, S. 41–47. – ISSN 0890-8044

[114] Wurm, Matthias: *Entwurf und Evaluierung einer clusterbasierten Sicherheitsarchitektur für drahtlose Sensornetze.* Karlsruhe, Deutschland, Institut für Telematik, Universität Karlsruhe (TH), Diplomarbeit, März 2006

[115] Yuan, Y. ; Agrawala, A.: A Secure Service Discovery Protocol for MANET / University of Maryland, Computer Science Department. Maryland, USA, 2003 (CS-TR-4498). – Forschungsbericht

[116] Zeng, X. ; Bagrodia, R. ; Gerla, M.: GloMoSim: A Library for Parallel Simulation of Large Scale Wireless Networks. In: *Proceedings of 12th Workshop on Parallel and Distributed Simulations (PADS) '98.* Banff, Alberta, Kanada : IEEE Computer Society, 1998, S. 154–161

[117] Zhu, Feng ; Mutka, M. ; Ni, L.: PrudentExposure: a private and user-centric service discovery protocol. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2004).* Orlando, Florida, USA : IEEE Computer Society, März 2004, S. 329–338. – ISBN 0-7695-2090-1

[118] Zhu, Feng ; Mutka, Matt ; Ni, Lionel: Splendor: A Secure, Private, and Location-Aware Service Discovery Protocol Supporting Mobile Services. In: *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications.* Fort Worth, Texas, USA : IEEE Computer Society, März 2003, S. 235–242. – ISBN 0-7695-1893-1

[119] Zhu, Feng ; Mutka, Matt W. ; Ni, Lionel M.: Service Discovery in Pervasive Computing Environments. In: *Pervasive Computing* 4 (2005), Oktober-Dezember, Nr. 4, S. 81–90. – ISSN 1536-1268

[120] Zimmermann, H.: OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. In: *IEEE Transactions on Communications* 28 (1980), April, Nr. 4, S. 425–432. – ISSN 0096-2244

Index