Erik-Oliver Blaß

Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen

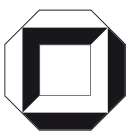
universitätsverlag karlsruhe

Erik-Oliver Blaß

Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen

Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen

von
Erik-Oliver Blaß



universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH)
Fakultät für Informatik, 2007

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2007
Print on Demand

ISBN: 978-3-86644-142-2

Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen

zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Dipl.-Inform. Erik-Oliver Blaß

aus Wuppertal

Tag der mündlichen Prüfung: 20. April 2007

Erster Gutachter: Prof. Dr. Martina Zitterbart
Universität Karlsruhe (TH)

Zweiter Gutachter: Prof. Dr. Felix Freiling
Universität Mannheim

—TEMPUS FUGIT!
Für Papa

Danksagung

Frau Professor Dr. Martina Zitterbart danke ich für die Leitung dieser Arbeit und für zahlreiche Anregungen und Ratschläge im Verlaufe ihrer Durchführung. Herrn Prof. Dr. Felix Freiling danke ich für die Übernahme des Korreferats und für wertvolle Hinweise auch auf formale Aspekte dieser Arbeit.

Danke den Wissenschaftlern Dr. Roland Bless, Dr. Curt Cramer, Dr. Artur Hecker (MCF) und Dipl.-Inform. Bernhard Hurler für zahlreiche Hilfestellungen und Diskussionen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

1. Einleitung

Computer und Computernetze sowie der Zugang des Menschen dazu befinden sich seit einiger Zeit im Wandel: Nicht nur am Arbeitsplatz kann der Mensch auf einen Computer und ein Netz zugreifen, sondern inzwischen begleiten den Menschen Computernetze überall. Zu Hause haben sich Computer, Fernseher und Stereoanlage zu einer *Home-Entertainment-Plattform* zusammengeschlossen, die dem Menschen über einen transparenten Internet-Zugang bei Bedarf den Genuß aktueller Musik, Filme und Fußballübertragungen ermöglicht. Zukünftig werden immer mehr Computer in immer mehr Bereichen des alltäglichen Lebens präsent sein.

Mark Weiser nennt diese Vision in seinen Arbeiten *Ubiquitous Computing* [238, 239] – zu Deutsch allgegenwärtige Computersysteme, die das komplette Leben und alle Lebensbereiche des Menschen durchdringen. Kleine Computer, deutlich kleiner als heutige Laptops oder PDAs, lassen sich in viele alltägliche Gegenstände integrieren und dienen dem Menschen. Beispielsweise informieren sich die Kaffeemaschine und die Kaffeetasse über drahtlose Kommunikationsschnittstellen darüber, wann neuer Kaffee zubereitet werden muß. Die Jalousie läßt sich gleichzeitig mit einer Zimmerlampe über den PDA oder eine andere drahtlose Fernbedienung ansteuern. Oder sogar weiter: Ein kleiner Computer, integriert in die Armbanduhr des Menschen, teilt beim Betreten des Raums seine Ankunft diesen Geräten automatisch mit, genauso wie beim Verlassen des Hauses elektrische Geräte automatisch ausgeschaltet werden. Die Computer sind derart in den Alltag integriert, daß der Mensch sie gar nicht mehr realisiert. Das Verhältnis zwischen Computern und Menschen verschiebt sich so. Aus der ursprünglichen Idee der Mainframe, „ein großer Computer dient mehreren Menschen“, über die aktuelle Situation „ein Desktop oder Laptop dient einem Menschen“, wird „viele kleine Computer dienen einem Menschen“.

Einen besonderen Aspekt des Ubiquitous Computing stellen die sogenannten drahtlosen Sensor-/Aktornetze, oder im folgenden einfach *Sensornetze*, dar. Sensornetze sind Netzwerke bestehend aus einer großen Anzahl, vielleicht Tausende, von Knoten, den sogenannten Sensoren. Bei ihnen handelt es sich um Kleinst-Computer, physikalisch kleine Rechner von wenigen Zentimetern und zukünftig gar Nanometern Größe. Ihnen steht nur wenig Energie in Form von Batterien zur Verfügung, und sie sind ausgestattet nur mit

einem leistungsschwachen Mikrocontroller als CPU. Diese Sensoren verfügen weiterhin über Funkschnittstellen und können damit drahtlos Daten untereinander austauschen. Die Aufgabe solcher Sensoren besteht zunächst ganz allgemein einmal darin, ihre Umgebung zu beobachten und zu messen beziehungsweise Ereignisse wahrzunehmen. In vielen Szenarien werden die gemessenen Informationen schließlich durch das (Sensor-)Netz an eine oder mehrere *Senken* oder *Basisstationen* gemeldet, beziehungsweise von Aktoren ausgewertet, die dann entsprechend der Informationen aktiv die Umwelt manipulieren können. Sensornetze eignen sich besonders zur Beobachtung räumlich *verteilter* und *heterogener* Phänomene: Einzelne Sensoren lassen sich an verschiedenen Positionen plazieren und messen je nach Sensortyp unterschiedliche Aspekte eines Phänomens.

Ein typisches Beispiel für den möglichen Einsatz eines Sensornetzes beschreibt die Wochenzeitung *Die Zeit* in ihrer Ausgabe 6/2006 [37]. Die Vitalfunktionen eines Patienten werden zu Hause durch ein Sensornetz überwacht. Kleine Sensoren, integriert in die Kleidung, messen an mehreren Stellen permanent seine Körpertemperatur, seine Herzfrequenz und seinen Blutdruck sowie seine Bewegungen. Die Sensoren tauschen die gemessenen Daten aus und versenden sie an eine Basisstation. Auf diese Weise wird kontinuierlich das Krankheitsbild des Patienten überwacht und aufgezeichnet. Kritische Veränderungen können sofort registriert und mögliche Gegenmaßnahmen eingeleitet werden. Das Sensornetz beobachtet das verteilte Phänomen „Gesundheitszustand“ des Patienten an mehreren Positionen des Körpers und mißt verschiedene Aspekte wie Temperatur oder Herzfrequenz.

Aus dem Artikel geht ein besonders wichtiger Aspekt von Sensornetzen hervor: Kommunikation und Zusammenarbeit. Sensoren kooperieren miteinander, um einen bestimmten Dienst, wie die Patientenüberwachung, zu leisten. Erst wenn der aus mehreren einzelnen Werten gemittelte Temperaturwert und der Blutdruck eine Schwelle überschreiten und der Bewegungssensor atypische Bewegungen registriert, wird die Basisstation alarmiert. Auf Netzwerkebene redet man von der sogenannten *Aggregation* der Daten, Daten-*Fusion* oder von *In-Network-Processing* [101, 113, 114, 179, 183, 212]: Da es häufig nicht notwendig ist, die detaillierten Meßwerte oder Beobachtungen der einzelnen Sensoren bis zur Datensenke zu transportieren, sondern die Senke meistens nur davon abstrahierte Informationen benötigt, werden Daten auf ihrem Transport zur Senke bereits im Netz zusammengeführt, kombiniert oder vorverarbeitet. Solch ein *Datentransport* heißt aggregierend. Einige der Sensorknoten, die *Aggregationsknoten*, sammeln die Meßwerte anderer Sensoren und verarbeiten diese vor. Das Sensornetz übernimmt dadurch einen Teil der Meßdatenauswertung selbst. Solch ein aggregierender Datentransport hat gegenüber dem einfachen Ansammeln aller Daten an der Senke und ihrer Auswertung dort einen enormen Energievorteil, wichtig in batteriebetriebenen Sensornetzen: Theoretisch kann eine Menge kostspieliger Funkübertragen durch Aggregation eingespart werden [179, 212], in Krishanamachari et al. [133] spricht man je nach Situation von bis zu 80%. Aggregation stellt demnach eine wichtige Eigenschaft in Sensornetzen dar.

1.1 Problemstellung

Diese neue Form des aggregierenden Datentransports unterscheidet sich von Transportschemata in bisherigen Netzen wie beispielsweise klassischem Festnetz, Internet, P2P-Netzen usw., und bringt neue Herausforderungen mit sich.

Eine der zentralen Herausforderungen betrifft die *Sicherheit* des Datentransports. Anhand des zuvor beschriebenen Beispiels läßt sich die zentrale Problemstellung motivieren, der

sich diese Arbeit widmet und welche *Die Zeit* in ihrem Ausblick so zusammenfaßt: „Ungeklärt ist noch, wie die Sensordaten am besten vor neugierigen Lauschern geschützt werden. Die intimen Datenströme sind noch nicht abhörsicher.“ [37]

Die in diesem und in vielen anderen Beispielen im Netz zwischen den Sensoren auszutauschenden Daten sind äußerst sensibel und schützenswert. Am Beispiel: Medizinische Informationen dürfen nicht unbefugt abgehört oder modifiziert werden. Gelangen Details über das Befinden eines Patienten in falsche Hände, so kann dies vielfältige negative Konsequenzen (zum Beispiel Rufschädigung, finanzielle Nachteile) nach sich führen. Werden Daten von einem *Angreifer* darüberhinaus mutwillig verändert, so führt dies beispielsweise zu falschen, lebensbedrohlichen Diagnosen. Allgemein: Der für das Erbringen von Dienstleistungen unabdingbare Transport der Daten im Sensornetz, Kommunikation angefangen von den messenden (Quell-)Sensoren bis hin zur Datensenke, muß gegen bestimmte Gefährdungen eines Angreifers abgesichert sein. Typische Gefahren sind unter anderem Abhören von Daten, Einfügen, Löschen, Verändern oder Wiedereinspielen von Daten oder die Maskerade eines Angreifers, bei der er eine falsche Identität vortäuscht [41]. Die Daten im Sensornetz müssen folglich vertraulich, unverändert und authentisch übertragen werden.

Eine besondere Herausforderung im Hinblick auf Sicherheit stellt dabei die Aggregation dar. Ein von einem Aggregationsknoten aus den einzelnen Meßwerten der Sensoren gebildetes *Aggregat* läßt in vielen Fällen keine Rückschlüsse auf die Originalwerte zu. Offen ist dann, wie ohne Kenntnis der Originalwerte Authentizität, Datenechtheit oder Originalität eines Aggregats überprüft beziehungsweise garantiert werden können. Die Gefahr besteht, daß ein Angreifer Aggregationen fälscht. Im Laufe dieser Arbeit wird gezeigt, daß Authentizität und Aggregation im Widerspruch zueinander stehen. Klassische Lösungen, wie zum Beispiel digitale Signaturen [156], eignen sich begrenzt für diese Problemstellung, weil einzelne Sensoren nur ihre eigenen Meßwerte und nicht das Aggregat signieren können und der Empfänger eines Aggregates ohne Kenntnis der einzelnen Signaturen nicht in der Lage ist, die Echtheit des Aggregats zu überprüfen.

Erschwerend kommt hinzu, daß Sensornetze aufgrund ihrer Eigenschaften besondere Anforderungen an jedwedes Sicherheitsprotokoll stellen: Sensornetze verzichten auf feste Infrastrukturkomponenten, müssen sich weitestgehend selbständig und ohne ständige Benutzerinteraktion organisieren, verhalten sich betreffend Netzein- und -austritt von Knoten sehr dynamisch usw. Weiterhin verfügen Sensoren nur über simple, schwache Hardware und können damit keine anspruchsvolleren Algorithmen ausführen. Durch ihren Batteriebetrieb gilt als oberstes Ziel jedes Sicherheitsprotokolls schließlich die Energieeffizienz [100]. Ein Angreifer verfügt in einem Sensornetz wahrscheinlich über wesentlich mehr Einfluß als in klassischen Netzen, da der physikalische Zugang zu Sensoren wesentlich einfacher und ihre primitive Hardware gegen Übernahmeversuche kaum zu schützen ist. Er wird daher in der Lage sein, einzelne Knoten zu korrumpieren [16]. All dies sind Faktoren, die den Entwurf von Sicherheitsprotokollen für Sensornetze verkomplizieren.

Es kristallisieren sich zwei große Problemfelder heraus:

1. *Schlüsselaustausch*
   Bevor die Sensoren innerhalb des Netzes überhaupt miteinander Daten austauschen beziehungsweise in Richtung Datensenke transportieren können, sind gemeinsame *Vertrauensanker* zwingend notwendig. Es können niemals Sicherheitsbeziehungen zwischen Knoten ohne solche gemeinsamen Geheimnisse, üblicherweise kryptographische Schlüssel, existieren. Ein an die Aggregation angepaßter, geeigneter

Austausch solcher Schlüssel im Sensornetz muß daher als Grundlage jedes weiteren komplexeren Sicherheitsprotokolls durchgeführt werden. Der Austausch muß, auch wenn ein Angreifer bereits Teile des Netzes erfolgreich korrumpiert hat, sicher funktionieren.

Sind schließlich gemeinsame Schlüssel im Netz ausgetauscht, kann man von einer Art *Basissicherheit* sprechen. Basissicherheit zwischen jeweils zwei Knoten bedeutet, daß zwischen beiden Knoten ein vertraulicher, unveränderter, authentischer usw. Datentransport möglich ist.

2. *Authentische Aggregation*
   Schlüsselaustausch und die daraus resultierende Basissicherheit löst allerdings noch nicht das Problem der Authentizität von Aggregation. Durch Aggregation wird es denjenigen Knoten, die Aggregate empfangen und weiterverarbeiten, so wie zum Beispiel der Senke, unmöglich, deren Echtheit, Korrektheit oder Authentizität zu überprüfen. Ein effizienter Mechanismus ist notwendig, der die Aggregation derart erweitert oder überprüfbar macht, daß die Senke und andere Knoten, Authentizität von Aggregaten feststellen können. Auch hier gilt es zu beachten, daß der Angreifer unter Umständen Teile des Netzes unter seine Kontrolle gebracht hat. Die oben erwähnte Basissicherheit zwischen jeweils zwei Knoten im Sensornetz ist hierfür eine notwendige Voraussetzung, allerdings noch nicht ausreichend.

Bisher gibt es kaum Forschungsarbeiten, die sich diesen Themen intensiv widmen. So sind beispielsweise alleine die Auswirkungen von Aggregation auf die Sicherheit im Sensornetz noch gar nicht ausreichend untersucht worden. Die Forschung konzentriert sich entweder auf grundsätzliche Sicherheisprobleme in Sensornetzen, wie beispielsweise sicheres Routing, sichere Informationsverteilung im Netz oder auch allgemeine Schlüsselverteilung, ohne sich jedoch speziell mit den Konsequenzen von Aggregation auseinanderzusetzen. Auch die verglichen mit klassischen Netzen deutlich größeren Möglichkeiten von Angreifern sind bisher weitestgehend unbeachtet.

1.2 Ziel der Arbeit

Die genannten Probleme zeigen, daß ein abgesicherter Datentransport die fundamentale Voraussetzung für jedes Sensornetz darstellt, in dem sensible und kritische Daten verarbeitet werden. Potentielle Betreiber von Sensornetzen werden diese neue Technologie nicht einsetzen können, solange deren Daten ungeschützt sind. In vielen Situationen zwingt der Gesetzgeber sogar die Betreiber zum Beispiel zur Geheimhaltung der verarbeiteten Daten, siehe StGb § 203 [217].

Diese Arbeit analysiert zunächst die Sicherheitsgefahren, mit denen ein Datentransport in drahtlosen Sensornetzen konfrontiert wird. Insbesondere stehen die Auswirkungen von *Aggregation* auf Sicherheit im Fokus. Daraus werden Anforderungen für geeignete Lösungen zum Absichern solchen Datentransports abgeleitet. Ausgehend von diesen Ergebnissen werden schließlich Protokolle, Verfahren und Mechanismen für einen sicheren, aggregierenden Datentransport in Sensornetzen entworfen. Deren Einsatz- und Leistungsfähigkeit wird schließlich anhand von Simulationen verifiziert. Im Detail ist das Ziel dieser Arbeit die Untersuchung der folgenden Fragestellungen:

- Welche neuen *Probleme* entstehen genau durch die besonderen Merkmale von Sensornetzen für die Sicherheit eines aggregierenden Datentransports? Welchen Einfluß nimmt Aggregation beispielsweise auf den Austausch von Schlüsseln? Inwiefern beeinflußt Aggregation die Authentizität der transportierten Daten? Wieso reichen bisherige Lösungsansätze, zum Beispiel aus klassischen Netzen, nicht aus?

- Wie kann ein realistisches *Angreifermodell* in aggregierenden Sensornetzen beschrieben werden? Genauer: Über welche Angriffsmöglichkeiten verfügt ein Angreifer in Sensornetzen, welche Form der Beeinflussung oder Gefährdung des Sensornetzes ist vorstellbar? Welche Unterschiede zu Angreifern in klassischen Netzen existieren?

- Wie können im Sensornetz *Schlüssel* ausgetauscht werden, so daß zwischen Knoten Basissicherheit existiert? Insbesondere: Wie können die Schlüssel dabei an die Aggregation angepaßt und unter Annahme des Angreifermodells ausgetauscht werden?

- Auf welche Weise kann die Aggregation von Meßwerten so abgesichert werden, daß der Empfänger des Aggregats sicher ist, daß sich das Aggregat ursprünglich aus Meßwerten von Sensoren zusammensetzt, die dafür legitimiert sind? Wie kann im Sensornetz *authentisch aggregiert* werden?

- Welche *Kosten* in Bezug auf kritische Ressourcen wie Energie oder Speicher entstehen durch den Austausch von Schlüsseln und das Absichern von Aggregation?

1.3 Ergebnisse und Gliederung der Arbeit

Kapitel 2 präsentiert die für das weitere Verständnis der Arbeit notwendigen Grundlagen über Sensornetze. Anhand zweier typischer Beispielanwendungen werden die besonderen Eigenschaften von Sensornetzen erklärt und die in dieser Arbeit getroffenen Annahmen spezifiziert. Daran anschließend motiviert die Definition eines realistischen Angreifermodells die Notwendigkeit für einen sicheren Datentransport und teilt diese Aufgabenstellung in zwei aufeinander aufbauende Teilprobleme auf: den Austausch von Schlüsseln sowie authentischen Datentransport. Der übrige Teil von Kapitel 2 stellt kurz die Hintergründe zu kryptographischen Schlüsseln und Authentizität vor.

In Kapitel 3 wird ein effizientes, den spezifizierten Sensornetzen und dem Angreifer genügendes Schlüsselaustauschprotokoll präsentiert: SKEY. Dazu analysiert Kapitel 3 zunächst neue Herausforderungen, vor denen ein Schlüsselaustausch in Sensornetzen steht. Damit lassen sich Entwurfsziele angeben, die ein ideales Schlüsselaustauschprotokoll in Sensornetzen erfüllen muß. Nach einem Überblick über den Stand der Forschung wird SKEY vorgestellt, das sämtliche Entwurfsziele erfüllt. Schließlich untersucht Kapitel 3 die prinzipielle Einsatzfähigkeit von SKEY in einer Evaluierung.

Das zweite Problem, das des authentischen Datentransports, bearbeitet Kapitel 4. Auch hier wird zunächst untersucht, welche neuen Herausforderungen durch die Eigenschaften von Sensornetzen und das Angreifermodell auf die Authentizität von im Netz gemessenen Daten entstehen. Nach Spezifizierung der Entwurfsziele eines idealen authentischen Transportprotokolls zeigt Kapitel 4, daß noch keine der bisherigen Forschungsarbeiten diesen genügen. Aus diesem Grund wird im Anschluß das Protokoll ESAWN präsentiert,

welches die Entwurfsziele mit Hilfe eines Sicherheit–Energie-Kompromisses erfüllt. Kapitel 4 schließt mit einer Evaluierung von ESAWN.

Eine kurze Zusammenfassung dieser Arbeit und der erzielten Ergebnisse findet sich neben einem Ausblick auf mögliche weitere Fragestellungen, die sich im Rahmen dieser Dissertation ergeben haben, in Kapitel 5.

Lesefluß durch diese Arbeit

Die Kapitel 3 und 4 hängen inhaltlich nur sehr grob zusammen und können deshalb unabhängig voneinander gelesen werden. Die einzige Voraussetzung für Kapitel 4 besteht darin, daß Sensorknoten über paarweise Schlüssel mit anderen Sensoren auf ihrem sogenannten Aggregationspfad besitzen. Für beide Kapitel ist allerdings Kapitel 2 über Grundlagen zwingende Voraussetzung. Hier werden die Grundlagen, Definitionen und Annahmen über Sensornetze, Aggregation und Sicherheit präsentiert, ohne die sich die meisten Aussagen in den Kapiteln 3 und 4 nicht verstehen lassen.

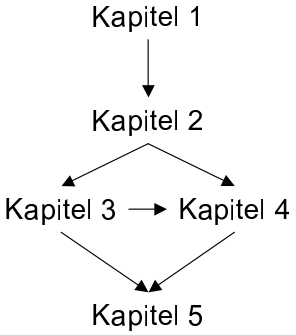


Abbildung 1.1 Leseflüsse durch die Kapitel

Zu erwähnen sei an dieser Stelle noch Anhang E, der auf Seite 197 eine Übersicht über alle kapitelübergreifenden, wichtigen mathematischen Symbole und ihre Bedeutung enthält.

2. Grundlagen und Annahmen

Ziel des Kapitels: Dieses Kapitel beschreibt Grundlagen über Sensornetze, die zum weiteren Verständnis der Arbeit notwendig sind. Zunächst werden anhand zweier detaillierter Szenarien Beispiele für Sensornetze vorgestellt. Bei den Beispielen handelt es sich um typische Anwendungen für Sensornetze, wie sie in naher Zukunft vorstellbar sind. Die Details dieser Beispiele dienen dazu, die charakteristischen Eigenschaften und Merkmale von Sensornetzen abzuleiten und daraus die Problemstellung und Herausforderung dieser Arbeit zu motivieren.

Dabei ist zu bedenken, daß Sensornetze sich je nach Szenario und Phantasie in *allen* ihren Konfigurationsaspekten unterscheiden können. Dies fassen beispielsweise Römer und Mattern [191] in einem sehr breiten „Design Space“, Merkmalsraum von Sensornetzen, zusammen. Die im folgenden beschriebenen Merkmale sind daher eher auch als Annahmen zu verstehen, die für die in dieser Arbeit betrachtete Kategorie von Sensornetzen gelten sollen. Es mag durchaus Sensornetze geben, die sich in dem einen oder anderen Aspekt von den hier angenommenen unterscheiden. Die sich eventuell daraus resultierenden Auswirkungen auf die Problemstellung werden dann allerdings auch an entsprechender Stelle im Text genannt.

Aufbau des Kapitels: Anknüpfend an die Beispielszenarien in Abschnitt 2.1 sowie die Merkmale und in dieser Arbeit getroffenen Annahmen in 2.2 folgt mit Abschnitt 2.4 ein Abschnitt über die fundamentalen Sicherheitprobleme, vor denen alle Sicherheitsprotokolle in Sensornetzen stehen. Im speziellen wird ein für Sensornetze realistisches Angreifermodell entworfen, daß sich von den Modellen in klassischen Netzwerken deutlich unterscheidet. Schließlich lassen sich daraus in Abschnitt 2.5 die Teilprobleme beziehungsweise Unterprobleme herleiten, die im Rahmen dieser Arbeit bearbeitet werden. Kapitel 2 schließt mit der Beschreibung einiger für diese Arbeit besonders wichtiger Sicherheitsgrundlagen.

2.1 Beispielszenarien

Wahrscheinlich wird ein wichtiges Aufgabengebiet künftiger Sensornetze die Unterstützung des Menschen beim Meistern alltäglicher Situationen oder die Unterstützung älterer

oder behinderter Menschen sein. Das erste der beiden Beispielszenarien geht daher auch detailliert auf diese Thematik ein.

2.1.1 Betreutes Wohnen

Menschen, die aufgrund ihres hohen Alters oder einer Behinderung nicht mehr in der Lage sind, ihren Alltag selbständig zu meistern, könnten durch den Einsatz eines Sensornetzes geeignet unterstützt werden. In einer Wohnung oder im Krankenhaus oder Seniorenheim könnte ein solches sich selbst organisierendes Sensornetz beispielsweise eigenständig in Abhängigkeit der Zimmertemperatur eine Klimaanlage ansteuern, bei Dunkelheit für ausreichende Beleuchtung sorgen oder automatisch notwendige Einkäufe planen. Als noch wichtiger muss die medizinische Überwachung einer Personen betrachtet werden: Sensoren sind in der Lage, permanent Blutdruck oder Herzfrequenz von Patienten zu überwachen, die Medikamentierung zu kontrollieren und können im Alarmfall schnell den Notdienst rufen. So könnte beispielsweise ein Sensor im Medikamentenschrank das Herausnehmen einer bestimmten Arznei feststellen und diese Information an einen weiteren Sensor übergeben. Dieser Sensor wiederum kennt die Krankengeschichte seines Patienten und kann die empfangenen Informationen weiterverarbeiten. Beispielsweise analysiert (aggregiert) er Herzfrequenz, Blutdruck, eingenommene Medikamente und die bisherige Krankengeschichte und reagiert entsprechend. Er könnte einen Aktor, vielleicht einen kleinen Lautsprecher oder ein kleines Display, ansteuern, um den Patienten vor einer Überdosierung zu warnen, ihm automatisch Tipps und Ratschläge zur Einnahme geben, Wechselwirkungen mit anderer Medizin aufzeigen und im Notfall Hilfe verständigen. Solche medizinischen Informationen sind extrem sensibel, da sie die intimste Privatsphäre des Menschen betreffen. Ein Mißbrauch kann ähnlich dem Vertauschen von Arzneimitteln fatale Folgen mit sich bringen – der Schutz der in einem solchen Sensornetz ausgetauschten Daten ist unabdingbar.

Die Anzahl der Sensoren in einem solchen Szenario ist sicherlich variabel: Sie reicht von einigen Dutzenden Sensoren in der Wohnung eines Patienten bis hin zu vielleicht einigen tausend Sensoren im Seniorenheim oder Krankenhaus mit vielen Patienten. Die Sensoren sind spezialisiert und abgestimmt auf ihre jeweiligen Aufgaben, Behinderungen oder Unzulänglichkeiten der betreuten Person. Es mag Sensoren zur Überwachung von Medikamenten geben, genauso wie in allen Zimmern der Patienten Sensoren und Aktoren zur Kontrolle von Licht, Zimmertemperatur oder ähnlichem vorhanden sind. Schließlich existieren noch Sensoren am Körper des Patienten zur Kontrolle der Vitalfunktionen. Ihre Energie beziehen die Geräte aus kleinen Batterien und kommunizieren drahtlos miteinander. Sie verfügen aus Gründen der Flexibilität über keinerlei festen, drahtgebundenen Zugang zu einer Art Kommunikationsinfrastruktur.

Da die Sensoren in diesem Szenario sehr klein und batteriebetrieben sind sowie größtenteils drahtlos miteinander kommunizieren können, kann es durchaus häufiger vorkommen, daß einzelne Knoten oder gar ganze Teile des Netzes vorübergehend oder permanent nicht erreichbar sind: Knoten „verlassen“ das Netz wegen aufgebrauchter Batteriereserve oder werden, wie alte Medikamente, einfach weggeworfen. Genauso versuchen Knoten, den größten Teil des Tages ihre Batterie durch Schlafzustand zu schonen. Der Sensor an der Medikamentenpackung kann bis zu dem Moment, an dem ein Medikament entnommen wird, schlafen. Wieder andere Knoten, befestigt am Körper der betreuten Person, sind aufgrund von Mobilität zeitweise nicht mehr in Funkreichweite usw. Eine permanente Kommunikationserreichbarkeit aller Knoten kann im Sensornetz nicht garantiert werden.

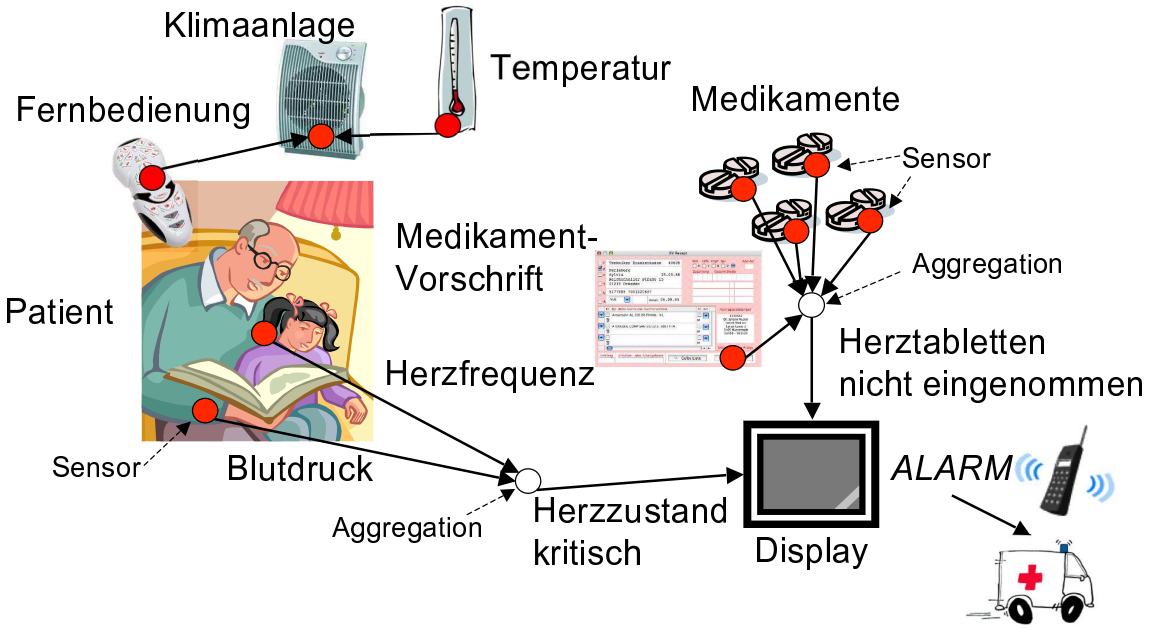


Abbildung 2.1 Beispiel für ein einfaches Sensornetz, alle roten Punkte sind Sensoren

Abbildung 2.1 zeigt ein einfaches Beispiel für solch ein Szenario: Im Haus einer betreuten Person sei ein Display, die *Datensenke* im Netz. Diese Datensenke ist an den konkreten Meßwerten der einzelnen (Quell-)Sensoren im Detail gar nicht interessiert. Der aktuell am Körper der Person gemessene Puls oder Blutdruck spielen zunächst nur eine untergeordnete Rolle. Erst wenn die Herzfunktionen einen kritischen Zustand einnehmen *und* noch keine Tabletten nach Medikament-Vorschrift dafür eingenommen worden sind, soll das Display entsprechend warnen oder den Notdienst alarmieren. Die Datensenke benötigt aggregierte, vorverarbeitete Daten: „Herzzustand kritisch“ *und* „Herztabletten nicht eingenommen“. Das sind abstraktere Informationen als nur der tatsächlich gemessene Puls. Die Meßwerte über Blutdruck und Herzfrequenz werden von einem Aggregationsknoten verarbeitet und eine Information über den Herzzustand, kritisch oder nicht, generiert. Die einzelnen Medikamente zusammen mit der Medikament-Vorschrift ergeben aggregiert die Information, daß benötigte Medikamente nicht eingenommen wurden. Daraus zusammen entscheidet das Display über einen Notfall. Die Aggregate, die an die Senke geschickt werden, berechnen sich dabei aufgrund komplexer Regeln, das heißt nicht nur durch einfache Mittel- oder Maximalwertbildung, sondern auch durch Zusammenhänge der unterschiedlichen Datenkategorien wie Blut*druck* und Herz*frequenz*. Gleichzeitiges Überschreiten der Herzfrequenz und des Blutdrucks bilden das Aggregat „Herzzustand kritisch“. Die Überwachung des Gesundheitszustandes ist nur ein Teil der Aufgaben des Sensornetzes. Wie in Abbildung 2.1 zu sehen, bietet das Netz noch die Möglichkeit, automatisch oder über eine Fernbedienung nach Benutzerwunsch die Klimaanlage zu steuern. Der Benutzer kann über die Fernbedienung beispielsweise eine bestimmte Raumtemperatur vorgeben, welche die Klimaanlage mit Hilfe eines Temperatursensors einzuhalten versucht. Die Klimaanlage aggregiert Benutzerwunsch und aktuelle Temperatur zu einem abstrakteren Wert wie „kälter“ oder „wärmer“.

Die Aufgaben dieses Sensornetzes sind vielfältiger Art. Die einzelnen stattfindenden Aggregationen „passen“ nicht immer inhaltlich zusammen. Nicht alles Sensorknoten müssen

mit allen anderen zusammenarbeiten. Dies ist eine wichtige Beobachtung, die später, zum Beispiel für das Verteilen von Schlüsseln, noch von Bedeutung sein wird.

2.1.2 Überwachen von Kernwaffentests

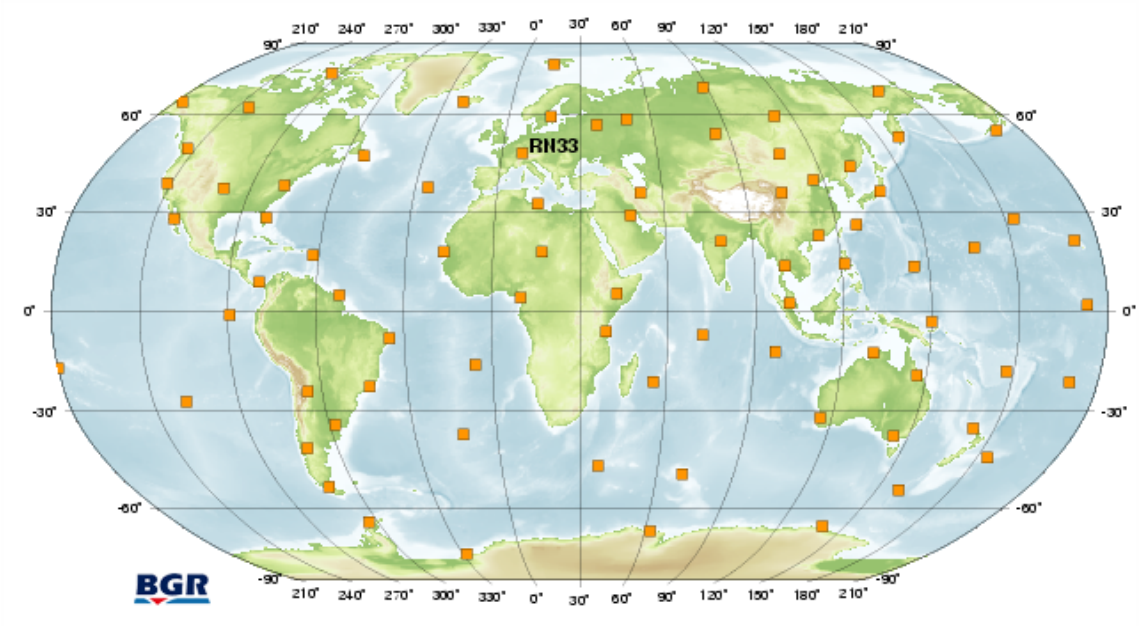


Abbildung 2.2 Weltweite Positionen von radionukliden Sensorfeldern, aus [42]. Jedes Quadrat steht für den Standort eines Feldes.

Seit Mitte der 1990er Jahre kümmert sich die *Comprehensive Nuclear Test Ban Treaty Organization* (CTBTO) [218] um die Einhaltung des Verbots atomarer Waffenforschung. Hervorgegangen aus den SALT II Verträgen zur Abrüstung und dem Comprehensive Nuclear Test Ban Treaty der 1970er Jahre, gehören dieser Organisation inzwischen 130 Staaten an. Diese verpflichten sich gegenseitig, in ihren Ländern in bestimmten, für Atomwaffentests geeigneten Regionen, zum Beispiel in Sibirien, Nevada oder Alaska, Sensoren zur Überwachung zu installieren. Teil dieser Installationen sind jeweils eine Menge seismischer Sensoren, Radionuklid-Sensoren und akustische Infraschallsensoren und Hydrophone, angeordnet in Feldern. In manchen Regionen kommen Sensornetzfelder bestehend aus verschiedenen Kategorien von Sensoren, zum Beispiel seismische und Radionuklid-Sensoren, zum Einsatz, in anderen Fällen sind ausschließlich Sensoren einer einzigen Kategorie im Gebrauch. So zeigt Abbildung 2.2 die aktuellen Positionen der weltweiten Radionukliden-Sensornetzfelder. Die Station RN33 befindet sich beispielsweise im deutschen Freiburg. Die Sensoren messen jeweils, ob bestimmte Kriterien, die auf einen neuen Kernwaffentest hinweisen, erreicht sind. Dies ist beispielsweise dann der Fall, wenn radioaktive Zerfälle übermäßig stattfinden, Schallwellen spezifischer Frequenzen meßbar sind oder wenn seismische Erschütterungen zunehmen. Alle Messungen sämtlicher Sensoren werden derzeit noch drahtlos per Satellit zu einer Zentrale in Wien übermittelt und dort ausgewertet. Auch hier wird intuitiv klar, daß die gemessenen Daten unbedingt sicher übertragen werden müssen. Neben der Geheimhaltung der Informationen, steht vor allem ihre Sicherheit gegenüber mutwilligen Modifikationen im Fokus. Führt ein Land illegal Atomwaffentests durch, so darf es die von den Sensoren gemessenen und nach Wien gemeldeten Daten nicht manipulieren können. Umgekehrt soll ein Angreifer nicht in der

Lage sein, unbemerkt Meßdaten so zu modifizieren, daß ein Land fälschlicherweise eines Atomwaffentests verdächtigt wird.

Aggregation

Die batteriebetriebenen Sensoren werden von Wartungspersonal in den „verdächtigen“ Regionen fest installiert und sind danach auf sich allein gestellt. Die einzelnen Sensoren verschicken ihre Daten, pro Tag immerhin knapp 12 MByte [208], direkt an die Zentrale. Die momentane Form des Transports, verschlüsselt zwischen jeweils einzelnen, messenden Sensoren und der „Senke“ in Wien, ist zwar sicher, allerdings nicht effizient.

Dieses Szenario zeigt das Potential, das in einem sicheren und effizienten Aggregationsprotokoll liegt: Während des Transports zur Zentrale nach Wien, werden die einzelnen Daten der Sensoren bereits vorverarbeitet und aggregiert. Anstatt der zentralen Auswertung könnten so beispielsweise die seismischen Sensoren ihre Daten untereinander derart austauschen, daß sie letztendlich selbst entscheiden können, ob genug seismische Aktivität vorliegt, um einen möglichen Atomwaffentest zu melden. Die seismischen Sensoren würden ihre Daten zu einem Aggregationsknoten schicken, genauso wie die radionukliden. Diese Aggregationsknoten entscheiden dann, ob Grenzwerte für ein einzelnes Phänomen überschritten sind und leiten diese Information, das Aggregat, weiter. Ein zentraler Aggregationsknoten pro Standort sammelt von den Aggregationsknoten pro Kategorie (seismisch, radionuklid, usw.) die erzeugten Aggregate und überprüft, ob Wechselwirkungen zwischen den Kategorien auf einen Atomwaffentest hindeuten. All dies hätte den Vorteil, enorme Energiemengen durch teure Satellitenübertragungen einzusparen. Auch die Aggregation heterogener Sensorwerte ist so möglich: Erst wenn mehrere kritische Meßgrößen überschritten werden, zum Beispiel hohe seismische Erschütterungen *und* Schallwellen bestimmter Frequenz, dann wird der Zentrale ein Alarm mitgeteilt. Sensoren verschiedener Typen könnten demnach ihre gemessenen Daten, vielleicht bereits aggregierten Daten, sammeln und dann weiterverrechnen: Von den seismischen Aktivtäten wird ein Mittelwert über mehrere Sensormessungen berechnet, bevor dieser zusammen mit dem Mittelwert der Schallwellen aggregiert wird.

2.2 Merkmale drahtloser Sensornetze

Aus den beschriebenen Beispielszenarien lassen sich nun typische Eigenschaften und Merkmale von Sensornetzen ableiten. Wie im späteren Verlauf der Arbeit noch zu sehen sein wird, haben diese einen entscheidenden Einfluß auf viele Entwurfsziele der Sicherheitsprotokolle. Im folgenden werden die entscheidenden Eigenschaften von Sensornetzen vorgestellt und ihre möglichen Implikationen auf Sicherheitsprotokolle zunächst kurz und allgemein skizziert. Auf die konkreten Auswirkungen, die bestimmte Eigenschaften für den Entwurf eines bestimmten Protokolls mit sich bringen, wird dann jeweils im entsprechenden Abschnitt über den Entwurf des Protokolls eingegangen.

Es sei an dieser Stelle nochmals erwähnt, daß Sensornetze sich in vielen Aspekten je nach Szenario und Phantasie unterscheiden können. Die folgende Aufstellung beschreibt die *Annahmen* über die Kategorie von Sensornetzen, welche in dieser Arbeit behandelt wird. Jeder Abschnitt stellt die jeweilige Problematik vor und faßt kurz die getroffenen Annahmen zusammen. Andere Sensornetze mögen sich in einigen Aspekten von den angenommenen unterscheiden. Falls dies Auswirkungen auf die Aufgabenstellung, Sicherheit, haben sollte, wird dies an entsprechender Stelle genannt.

2.2.1 Ressourcenarmut

Sensorknoten verfügen häufig über kleine physikalische Ausmaße. Damit sie beispielsweise in alltägliche Gegenstände integrierbar oder für einen Patienten am Körper oder an seiner Kleidung bequem zu tragen sind, dürfen sie nicht größer als wenige Zentimeter sein. Dementsprechend beschränkt sind die Möglichkeiten, komplexe Hardware auf diesen Knoten zu integrieren. Üblicherweise kommen von ihrer Rechenleistung her schwache *Mikrocontroller* als CPU mit wenig Speicher zum Einsatz. Sie haben neben ihrer geringen Größe zudem den Vorteil, besonders energieeffizient zu arbeiten. Da Sensoren, zum Beispiel am Körper des Patienten oder in großer Wassertiefe beim Bewachen von Testgebieten, unabhängig von externer Versorgung nur über eigene Stromquellen[1] verfügen, ist Energieeffizienz ein kritisches Kriterium bei der Auswahl der Sensor-Hardware. Üblicherweise kommen Batterien als Energiequelle für die Sensoren zum Einsatz – ist ihre Kapazität erschöpft, stirbt der Sensor und kann seinen Dienst nicht mehr erbringen. Zwar sind neben Batterien durchaus andere Formen der Energieversorgung für Sensornetze denkbar, diese eignen sich jedoch noch nicht allein zur ausreichenden Energieversorgung, sondern nur als Unterstützung von Batterien [193]. Beispiele für solche alternativen Energiequellen sind unter anderem Solarzellen, Mikro-Brennstoffzellen, Piezo-Elemente oder Systeme, die den Seebeck-Effekt (Spannungserzeugung durch Temperatur- oder Druckunterschiede) ausnutzen.

Die konkrete Wahl einer Hardware-Plattform, die physikalischen Ausmaße und die Konfiguration eines Sensors hängen von seiner konkreten Aufgabe innerhalb eines bestimmten Szenarios ab. Möglicherweise besteht ein Sensornetz aus einer heterogenen Hardware-Landschaft. Es gibt Knoten mit einer besseren Hardware-Ausstattung als andere, manche Knoten sind nicht nur batteriebetrieben, sondern verfügen über Stromanschlüsse usw. Einen de facto Standard für Sensorknoten, wie man ihn seit Jahren aus der Intel x86/PC-Welt kennt, gibt es nicht. In verschiedenen Sensornetzen können verschiedene Sensoren mit verschiedener Hardware zum Einsatz kommen. Sogar innerhalb eines Sensornetzes kommen verschiedene Sensoren aufgrund verschiedener Aufgaben zum Einsatz: Die Sensoren zur Bestimmung des Blutdrucks werden völlig verschieden von den Sensoren in den Medikamentenpackungen sein. Es mag Sensoren mit mehr oder mit weniger Speicher geben, mit mehr zur Verfügung stehender Rechenleistung oder Energie oder mit weniger. Die Heterogenität von Sensoren kann den Entwurf neuer (Sicherheits-)Protokolle erschweren, falls schwächere Sensorknoten nicht in der Lage sind, komplexere Protokollschritte auszuführen. Die Protokolle müssen dann so entworfen sein, daß stärkere Sensoren schwächere unterstützen. Aus Sicherheitssicht liegt jedoch die weitaus größere Herausforderung in einem *homogenen* Netz aus sehr einfachen, schwachen Knoten: Wenn ein Protokoll sicher in einem homogenen Netz aus *schwachen* Knoten arbeitet, so arbeitet es auch sicher in einem heterogenen Netz mit teilweise *stärkeren* Knoten, da die stärkeren Knoten die gleichen Protokollschritte ausführen müssen wie die schwächeren. Analog dazu ist die Annahme, daß ein Netz zwar aus heterogenen Knoten mit unterschiedlicher Hardware besteht, die Knoten allerdings allesamt bestimmte Mindestvoraussetzungen an Leistung erfüllen. Eignet sich ein Protokoll für die Mindestvoraussetzung an Hardware, so läuft es auch auf den stärkeren Knoten.

Nicht zuletzt auf Grund von viel Werbung und starker Präsenz auf wissenschaftlichen Konferenzen hat sich seit einiger Zeit die Firma Crossbow mit ihrer MICA-MOTE-Sensorfamilie in der Forschungsgemeinde durchgesetzt. Hervorgegangen aus Projekten der

[1] Diese Arbeit betrachtet *Strom* und *Energie* als Synonym.

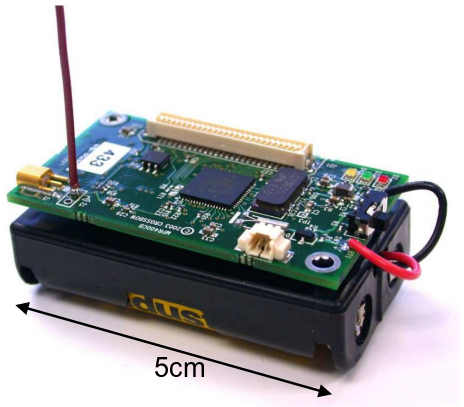


Abbildung 2.3 MICA2-Mote [66], zwei 1,5V AAA Batterien auf der Unterseite

Universität Berkeley stellt sich inzwischen deren MICA2-Plattform [66] als eine sehr populäre und häufig zitierte Basis heraus. Viele konkurrierende Plattformen sind den MICA2-Motes ähnlich und verwenden teilweise identische Komponenten wie beispielsweise den Mikrocontroller. Zu den MICA-ähnlichen Plattformen gehören unter anderem die sogenannten BTNodes der ETH Zürich [185] oder die am Institut für Telematik entwickelte Karlsruher SensorNEtz-Plattform KaSNEP [27]. Abbildung 2.3 zeigt einen MICA2-Knoten. Den größten Teil des Sensors nimmt dabei seine Energieversorgung, zwei handelsübliche 1,5V AAA Batterien, ein. Die Batterien sind auf der Unterseite des Motes in Abbildung 2.3 zu sehen. Alle anderen Komponenten wie CPU oder Funkschnittstelle sind auf dem sogenannten „Board“ oberhalb der Batterien integriert. Bestandteil dieses Sensorknotens sind ein 8 Bit AVR Mikrocontroller sowie eine drahtlose Funkschnittstelle, wahlweise Bluetooth, ZigBee oder eine proprietäre Technik im 433 MHz Band. Beim Mikrocontroller handelt es sich um den ATMega128L der Firma Atmel, getaktet mit bis zu 8 MHz leistet er ca. 8 MIPS, seine Harvard-Architektur verfügt über 4 KByte RAM sowie 128 KByte Flash-Programmspeicher. Diese CPU benötigt im normalen Betrieb zum Ausführen von Instruktionen bei 2,7 V ungefähr 5 mA Energie [11]. Kommunizieren kann ein solcher Knoten beispielsweise mit Hilfe eines ChipCon Single IC Transceivers. Der Datendurchsatz eines solchen Transceivers beträgt 38400 Bit/s bei einer Funkreichweite von ca. 10 m. Der Transceiver benötigt dabei zum Versenden von Daten zusätzlich zum Mikrocontroller weitere 16 mA Energie [69]. (Solche Angaben über die Leistungseigenschaften der MICA2-Hardware sind in der Literatur häufig unterschiedlich und widersprechen sich teilweise. Dies diskutiert Anhang A, S. 173.) An dieses Hardware-Grundgerüst eines MICA2 kann nun verschiedenste Sensorik angeschlossen werden, wie zum Beispiel Temperaturfühler, Bewegungs- und Beschleunigungsmelder usw.

Die Hardware-Konfiguration eines solchen Sensorknotens läßt bereits erahnen, vor welchen Problemen jedes Sicherheitprotokoll stehen muß: Der mit 4 KByte äußerst knapp bemessene Speicher, der zwischen Applikation, Netzwerk- oder sonstiger Middleware und Sicherheitsprotokollen geteilt wird, läßt keinen Raum für große Mengen an Schlüsseln, Zertifikatsketten und ähnlichem. Man darf nicht vergessen, daß die 4 KByte in erster Linie dazu dienen, Meßwerte zu speichern und zu verarbeiten. Mit nur 8 Bit Wortbreite und 8 Millionen Instruktionen pro Sekunde sind außerdem komplexe (kryptographische) Operationen teuer und müssen, wenn überhaupt, selten und wohlbedacht eingesetzt werden. Große, zum Beispiel minutenlange Verzögerungen durch komplexe Arithmetik sind in den sicherheitsrelevanten Szenarien unangebracht. Die Tatsache, daß sich der Herzzustand gerade dramatisch verschlechtert hat, darf nicht durch minutenlanges Berechnen von

kryptographischen Operationen verzögert werden. Es kommt hinzu, daß solch zeitraubende Operationen durch den Prozessor letztendlich auch einen Energieverbrauch darstellen.

Zusammenfassung: Annahmen zur Knotenkonfiguration in dieser Arbeit

Diese Arbeit nimmt ein homogenes Netz aus schwachen Knoten an, beziehungsweise jeder Knoten soll eine minimale Hardware-Voraussetzung erfüllen entsprechend der Ausstattung der verbreiteten MICA2-Knoten [66]. Jeder Knoten verfügt wie beschrieben über einen ATMEL 8 Bit Mikrocontroller als Basis, eine endliche Batterie als Energiequelle und die simple, proprietäre Funkschnittstelle im 433 MHz Band zur Kommunikation. Die in dieser Arbeit entworfenen Protokolle eignen sich damit auch für andere, unter Umständen heterogene, Szenarien, in denen alle Knoten über mindestens diese Hardware-Voraussetzungen verfügen.

2.2.2 Fehlende Infrastrukturen

Sämtliche Aufgaben und Dienste müssen im Sensornetz unter Verzicht von dedizierten Infrastrukturkomponenten erledigt werden. Zwar existieren im Sensornetz solche Komponenten, wie zum Beispiel die Datensenke oder die Basisstation in Form des Displays in der Wohnung eines Patienten oder der Atom-Zentrale in Wien, jedoch kann im Sensornetz nicht in jeder Situation die Erreichbarkeit solcher zentraler Komponenten zugesichert werden: Es mag sein, daß beim Kauf eines neuen Medikaments, an dessen Verpackung ein Sensor verbaut ist, ein Schlüsselaustausch mit irgendeinem anderen Knoten im Netz notwendig wird, sich die Basisstation jedoch gerade nicht in Funkreichweite befindet. Knoten in ihrem direkten Umfeld könnten zeitweise ausfallen und so temporär keine Nachrichten mehr an sie weiterleiten. Genauso wie Basisstationen sollen in dieser Arbeit auch keine anderen Infrastrukturkomponenten im Netz verfügbar sein: Beispiele dafür aus „klassischen“ Netzen sind Schlüssel-Server wie Key-Distribution-Center [158], Certification Authorities (CAs) [34] oder andere Server.

Aus Sicherheitssicht vereinfacht die Verwendung einer erreichbaren, zentralen Infrastrukturkomponente viele Schritte in einem Sicherheitsprotokoll, da diese Zentralen den Sensorknoten in vielen Dingen assistieren können. Als Beispiel sei hier wieder die Verwendung eines Schlüssel-Servers genannt, der jeweils zwei Sensoren bei einem Schlüsselaustausch hilft. Allerdings präsentieren sich Zentralen im Sensornetz für einen Angreifer als ideale *Single Point of Failure* (SPF): Ein Angriff könnte sich darauf konzentrieren, Nachrichten auf dem Weg zu oder von dieser Zentrale zu sabotieren oder gar die Zentrale selbst zu korrumpieren. Auch eine mögliche Replikation der Senke stellt zumindest in Sensornetzen keine befriedigende Lösung dar. Der mit Replikation verbundene Aufwand zur Synchronisierung der Replikate kostet unnötig Energie.

Zusammenfassung: Annahmen über fehlende Infrastrukturen in dieser Arbeit

Sensornetze in dieser Arbeit müssen auf die Verwendung von ständig erreichbaren Infrastrukturkomponenten verzichten, da die Erreichbarkeit solcher Zentralen nicht permanent gewährleistet werden kann und sie sich einem Angreifer als ideales Ziel präsentieren würden. Die in dieser Arbeit entworfenen Sicherheitsprotokolle ohne Infrastrukturkomponenten arbeiten allerdings auch sicher in Szenarien mit Infrastrukturkomponenten: Sie verwenden die Infrastrukturkomponenten einfach nicht.

2.2.3 Selbstorganisation und Spontaneität

Unter dem *Deployment* eines Sensorknotens oder des gesamtes Sensornetzwerks versteht man die Installation der Knoten an ihrem Zielort. Die Knoten werden vom Benutzer des Netzes, das kann zum Beispiel der Patient im Seniorenheim sein oder das Wartungspersonal der Atomenergiebehörde, ausgebracht und verteilt. In dem Moment, wo ein Medikament gekauft und in den Medizinschrank gelegt oder der seismische Sensor in der Erde vergraben wird, ist der Sensor fertig installiert.

Danach gibt es aus einfachen Durchführbarkeitsgründen keine oder höchstens sehr eingeschränkte Möglichkeiten für den Benutzer, die Sensorknoten zu warten: Der typische Patient hat ohne Spezialwerkzeug und Spezialkenntnisse gar nicht die Möglichkeit, Sensoren neu zu konfigurieren. Zudem fehlt den einfachen Sensoren oftmals die dafür notwendige Benutzerschnittstelle. Im anderen Szenario ist es einfach zu teuer, häufig Wartungsarbeiten in entlegenen Regionen durchzuführen. Der Benutzer kann auch nicht die aktuelle Konfiguration des Netzes kennen. Knoten können auf Grund von leerer Batterie oder Defekten einfach ausfallen. All diese letzten Punkte sollen nur verdeutlichen, daß der Benutzer nicht beliebig mit den Sensoren im Netz interagieren kann, sondern nur mit Hilfe von Software über besondere Zugangspunkte zum Netz: ein oder mehrere Senken.

Aber auch die Software-seitige Organisation muß nach dem Deployment völlig autonom ablaufen: Soll der Sensor seinen Dienst erbringen oder am Ablauf eines Sicherheitsprotokolls teilnehmen, so soll dies ohne irgendwelche nachträgliche *Benutzerinteraktionen* möglich sein. In dieser Arbeit erlaubte Benutzerinteraktionen beschränken sich nur auf die normale Inanspruchnahme von Diensten, die das Netz erbringt, die Abfrage des Netzes nach Werten usw. Andere Formen der Interaktion, zum Beispiel spezielle, vom Benutzer extra angestoßene Softwarebefehle zur Konfiguration des Netzes oder physische Interaktion sollen nicht möglich sein. Der Sensor muß nach seinem Deployment alleine und selbständig arbeiten beziehungsweise dazu ausschließlich die Hilfe der anderen, bereits im Netz befindlichen Knoten verwenden. Grundsätzlich ist gerade die Idee hinter Sensornetzen, daß sich der Benutzer keinerlei Gedanken über die Netzkonfiguration oder Konfigurationssänderungen machen soll – das Sensornetz muß sich diesbezüglich selbst organisieren. Jegliche Form von Interaktion, Knotenkonfiguration oder Unterstützung von Knoten bei ihrer Arbeit, die über eine normale Inanspruchnahme von Diensten hinausgeht, sollte sich auf die Phase des Knoten-Deployments beschränken.

Die für den Benutzer einzige Möglichkeit, Dienste des Sensornetzes in Anspruch zu nehmen, läuft über die Senken. Nur über die Senken stellt der Benutzer seine Anfragen nach Diensten an das Netz und erhält schließlich von den Senken Antworten auf die Anfragen. Die Senke stellt dem Benutzer dazu eine passende physische Schnittstelle, wie ein Display, eine Tastatur oder ein Laptop, und eine Art „Anfragesprache“ (query language), in der ein Benutzer seine Anfragen an das Netz stellen kann, zur Verfügung. Wie das Sensornetz durch den Benutzer gestellte Anfragen, die mit Hilfe der Senke formuliert werden, bearbeitet, ist nicht Teil dieser Arbeit. Siehe dazu zum Beispiel Heinzelman et al. [101], Intanagonwiwat et al. [113, 114], Madden et al. [149], Ramachandran et al. [183].

Analog zur Selbstorganisation verhält es sich mit der *Spontaneität* in Sensornetzen. Einzelne Knoten werden dem Netz sehr spontan und ohne große Planung vorab hinzugefügt: Beispielweise nicht zwangsläufig in bestimmten Perioden, sondern „On-Demand“, bei Bedarf. Das Netz muß sich dann in so einer Situation spontan für den neuen Knoten selbst rekonfigurieren. Das bedeutet, häufig muß das Routing der Knoten untereinander angepaßt werden, Schlüssel für den neuen Knoten müssen ausgetauscht werden usw.

Selbstorganisation und Spontaneität haben auf Sicherheitsprotokolle gewichtige Auswirkungen, insbesondere beim Hinzufügen eines neuen Knotens zum Netz. Ein neuer Knoten *kann* dem Netz nicht völlig spontan, selbstorganisierend und ohne Benutzerinteraktion zugefügt werden, ohne daß die Netzwerksicherheit automatisch darunter leidet. Wäre dies möglich, so könnte ein Angreifer auch unbemerkt einen korrumpierten Knoten, ein Knoten, der ein „böses“ Ziel verfolgt, dem Netz hinzufügen.

Zusammenfassung: Annahmen über Selbstorganisation und Spontaneität

Für die Sensornetze in dieser Arbeit wird angenommen, daß nach dem Deployment kein weiterer Kontakt zu den einzelnen Knoten möglich ist. Nur bis unmittelbar vor dem Ausbringen eines Knotens kann der Benutzer diesen speziell für das Netz vorbereiten und konfigurieren. Sämtliche Dienste und Aufgaben, die einzelne Knoten ab dann durchzuführen haben, zum Beispiel der Austausch von Schlüsseln oder das sichere Aggregieren von Daten, geschieht komplett selbstorganisierend und nur unter Zuhilfenahme anderer bereits im Netz befindlicher Sensoren. Der Benutzer kennt zu keinem Zeitpunkt genaue Details über die Netzwerkkonfiguration, zum Beispiel welcher Knoten gerade mit welchem Schlüssel austauschen muß, zusammen aggregiert usw.

2.2.4 Kommunikationsfluß: Aggregation

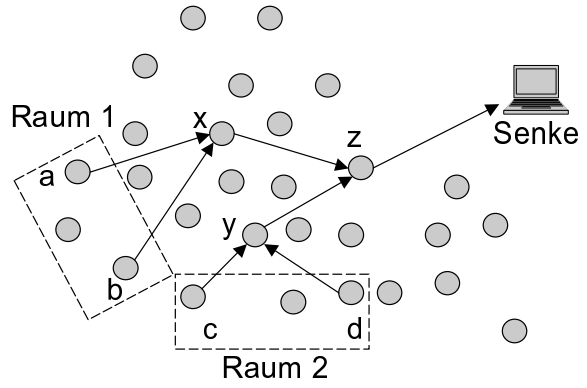
Aggregation ist eine für den Datentransport typische Eigenschaft im Sensornetz. Das Sensornetz soll Daten messen und diese Daten der Senke und damit dem Benutzer des Netzes mitteilen. Dabei ist dem Benutzer häufig ein konkreter Meßwert eines bestimmten Sensors völlig unwichtig. Im Gegenteil, der Benutzer wünscht sich eine eher abstraktere Information, die sich aus den einzelnen Teilinformationen der Sensoren zusammensetzt. Nun besteht die Möglichkeit, daß sämtliche Sensoren ihre einzelnen Meßwerte durch das komplette Netz jeweils an die Datensenke transportieren und die Datensenke aus den einzelnen Werten ihre gewünschte Information berechnet. Der Nachteil dieses intuitiven Ansatzes liegt im damit verbundenen Energieverbrauch: Die Menge der einzelnen Meßwerte kann sehr groß werden und die Meßwerte durch das komplette Netz zu transportieren, kostet aufgrund der dafür notwendigen Funkübertragungen wertvolle Energie.

Daher bietet sich in Sensornetzen ein anderer Kommunikationsfluß an: *Aggregation*. Weil die vom Benutzer gewünschte Information sich aus einzelnen Meßwerten der Sensoren zusammensetzt, kann das Zusammenfügen der Meßwerte auch innerhalb des Netzes geschehen. Sensoren messen ihre Werte und schicken sie an einen sogenannten *Aggregationsknoten*. Dieser Knoten sammelt Meßwerte einer Gruppe von Sensoren und berechnet daraus einen zusammengesetzten Wert, das *Aggregat*. Dieses Aggregat wird weiter in Richtung Senke transportiert. Formaler läßt sich Aggregation wie folgt beschreiben:

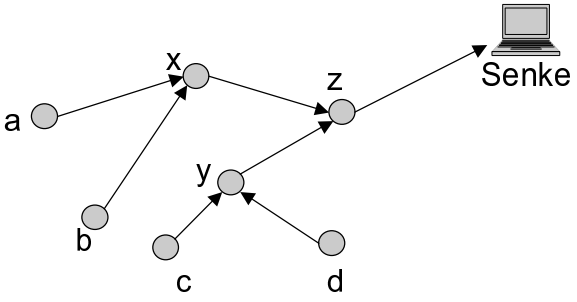
Definitionen zur Aggregation

Zu einer *Aggregation* gehören:

- ein Sensorknoten x, der sogenannte *Aggregationsknoten*
- δ verschiedene Sensorknoten $x_1, x_2, \ldots, x_\delta$, die x ihre Daten $X_1, X_2 \ldots, X_\delta$ irgendwelcher Wertebereiche $\mathbb{W}_1, \ldots, \mathbb{W}_\delta$ liefern. Die x_i heißen auch *Quellsensoren*.



(a) Beispiel für ein aggregierendes Sensornetz



(b) Aggregationsbaum

Abbildung 2.4 Aggregation als baumartiger Kommunikationsfluß

- eine Rechenvorschrift, die multivariate *Aggregationsfunktion* $f_x : \mathbb{W}_1 \times \mathbb{W}_2 \times \ldots \times \mathbb{W}_\delta \rightarrow \mathbb{W}$
- das *Aggregat* $\mathrm{agg}_x = f_x(X_1, X_2, \ldots, X_\delta)$, das x aus $X_1, X_2 \ldots, X_\delta$ und f_x berechnet. Man sagt, x *aggregiert* $X_1, X_2 \ldots, X_\delta$ zu agg_x.

An dieser Stelle gilt es, drei wichtige Beobachtungen festzuhalten:

1. Aggregation kann kaskadieren.

Von Aggregationsknoten bereits gebildete Daten können durchaus auf einer logisch höheren oder abstrakteren Ebene *nochmals* aggregiert werden. Als Beispiel dafür zeigt Abbildung 2.4 das Modell eines Seniorenheims. In diesem Seniorenheim soll die Klimasteuerung des gesamten Gebäudes über die gemittelten Temperaturen aus einzelnen Räumen gesteuert werden. In den einzelnen Räumen sind jeweils 2 Temperatursensoren angebracht, siehe Abbildung 2.4(a). Die Sensorknoten bezeichnet mit a und b messen die Temperatur in Raum 1, zum Beispiel an der Decke und am Boden, während c und d die in Raum 2 messen. Da die Senke, hier dargestellt durch einen Laptop, hingegen nur an der Durchschnittstemperatur des gesamten Gebäudes interessiert ist, aggregieren Knoten x beziehungsweise y die Temperaturen der einzelnen Räume jeweils zu Mittelwerten und leiten diese an z weiter. Schicken die Sensoren a und b ihre gemessenen Temperaturen A und B an x, dann berechnet x mit seiner Aggregationsfunktion $f_x(A, B)$ den Mittelwert der Temperatur in Raum 1 $\mathrm{agg}_x = f_x(A, B) = \frac{A+B}{2}$. Knoten y berechnet $\mathrm{agg}_y = f_x(C, D) = \frac{C+D}{2}$. Es findet nun allerdings eine weitere Aggregation statt und zwar auf den bereits „vorverarbeiteten“ Daten, den Aggregaten agg_x und agg_y: Knoten z berechnet die Durchschnittstemperatur des Gebäudes durch $\mathrm{agg}_z = f_z(\mathrm{agg}_x, \mathrm{agg}_y) = \frac{\mathrm{agg}_x + \mathrm{agg}_y}{2}$. Die Aggregation *kaskadiert*.

Die Aggregationsknoten verringern jeweils durch ihre Aggregation das Gesamtvolumen der im Netz zu transportierten Daten. In diesem Beispiel müßten nicht alle 4 an den Quellsensoren gemessenen Daten an die Senke transportiert werden, sondern die 4 Meßwerte reduzieren sich schrittweise auf nur ein Datum. Dies reduziert die insgesamt notwendige Energie für aufzuwendende Funkübertragungen. Da dadurch außerdem weniger Medienzugriffe auf das Funkmedium notwendig sind, verringert diese Technik weiterhin die Möglichkeit, daß *Kollisionen* beim Zugriff auftreten können und ermöglicht damit einen im Sensornetz insgesamt höheren Datendurchsatz.

2. Die Aggregationsfunktionen sind beliebige, berechenbare Funktionen

Die bisher wenigen Forschungsarbeiten, die sich mit der Problematik sicherer Aggregation beschäftigen, siehe deren Zusammenfassung in Abschnitt 4, S. 109, gehen der Einfachheit halber immer von sehr beschränkten Aggregationsfunktionen aus. Bei den Aggregationsfunktionen handelt es sich häufig um sehr primitive arithmetische Operationen wie beispielsweise Mittelwertbildung, Maximum-, Minimumfindung usw. Aggregation kann allerdings deutlich komplexer sein. Aus Blutdruck und Herzfrequenz, zwei völlig unterschiedlichen Wertedomänen und Bedeutungen, kann ein komplexes Aggregat über den Herzzustand berechnet werden. Die dafür notwendige Aggregatfunktion läßt sich durch eine mathematische Rechenvorschrift darstellen, ist von ihrer Form her jedoch weitaus komplizierter als einfache Mittelwertbildung. Aggregationsfunktionen können eine beliebige (mathematische) Form haben, solange diese sich durch ein Computerprogramm *berechnen*[2], läßt. Weiterhin sind die Aggregationsfunktionen durch die auf Sensorknoten zur Verfügung stehende geringe Rechenleistung beschränkt. Aggregationsfunktionen können also beliebige, effizient berechenbare Funktionen sein.

Auf der anderen Seite ist auch die „einfachste“ Form der Aggregation, durchaus denkbar: Die *Identitätsfunktion* als Aggregationsfunktion f_x eines Aggregationsknotens x. Bei dieser Aggregationsfunktion sammelt Aggregationsknoten x eingehende Daten nur und fügt sie dann zur Weiterleitung in eine einzelne Nachricht zusammen: $f_x(X_1, \ldots, X_\delta) = (X_1, \ldots, X_\delta)$. Da die Daten X_i nicht einzeln in Nachrichten verschickt werden, sondern mehrere zusammen in einer Nachricht, spart auch diese Form von Aggregation in vielen Situationen schon Energie – zum Beispiel genau dann, wenn die Mindestlänge einer Nachricht größer als die einzelnen Daten sind.

3. Aggregation bildet einen Baum.

Betrachtet man den *Fluß* der transportierten Daten in Abbildung 2.4(b), von den messenden Sensoren bis hin zur Datensenke, so fällt auf, daß dieser eine Hierarchie, einen Baum bildet. Die Quellsensoren schicken ihre Meßwerte an Aggregationsknoten, diese schicken ihre Aggregate an logisch nächst höhere Aggregationsknoten usw. Solch ein Aggregationsbaum läßt sich formaler wie folgt beschreiben:

Definitionen zum Aggregationsbaum

Ein *Aggregationsbaum* $G = (V, E)$ ist ein zusammenhängender, gerichteter, zyklenfreier Graph, der aus einer Menge V von Knoten sowie einer Menge E von gerichteten Kanten besteht. Es gilt:

[2] Zur Definition des Wortes *berechnen* siehe Schöning [203].

- die Knoten des Graphen repräsentieren die Sensoren, die Wurzel des Graphen w steht für die Senke.
- $n = |V|$ heißt die *Gesamtanzahl* aller Sensoren
- der *Aggregationsgrad* δ_x eines Knotens $x \in V$ ist die Menge der eingehenden Kanten bei x
- die Menge der Knoten $\{x_i | x_i \in V, \delta_{x_i} = 0\}$ bezeichnet die *Blätter* von G.
- alle anderen Knoten, für die gilt $\Lambda = \{x_i | x_i \in V, \delta_{x_i} > 0\}$, sind die *Aggregationsknoten* Λ von G. Aggregationsknoten sind die inneren Knoten in G.
- der *(durchschnittliche) Aggregationsgrad* δ von G ist das arithmetische Mittel der Aggregationsgrade sämtlicher Aggregationsknoten:

$$\delta = \frac{\sum_{x_i \in \Lambda} \delta_{x_i}}{|\Lambda|}.$$

- existiert zwischen zwei Knoten x_1 und x_2 eine gerichtete Kante $\overrightarrow{x_1 x_2}$, dann haben x_1 und x_2 eine *Aggregationsbeziehung* miteinander. Die Kanten von G heißen demnach die *Aggregationsbeziehungen* von G.

  Existieren Aggregationsbeziehungen $\overrightarrow{x_1 x}, \overrightarrow{x_2 x}, \ldots, \overrightarrow{x_{\delta_a} x}$ in G, dann *aggregiert* x die Daten der Quellsensoren $x_1, x_2, \ldots, x_{\delta_x}$; siehe dazu die Definitionen auf Seite 16.

- existiert ein gerichteter Weg oder auch *Pfad* $\mathcal{P}_{x_1 x_l} = \overrightarrow{x_1 x_2 \ldots x_l}$ über Aggregationsbeziehungen zwischen zwei Knoten x_1 und x_l, dann heißt x_l *Vorgänger(knoten)* von x_1 in G. Analog heißt x_1 ein *Nachfolger(knoten)* von x_l in G. Besteht eine Aggregationsbeziehung $\overrightarrow{x_1 x_2}$, dann heißt x_2 *direkter* Vorgänger oder *Vater* von x_1 in G. Analog heißt x_1 *direkter* Nachfolger oder *Sohn* von x_2.
- der *Aggregationspfad* $\mathbb{P}_x$ eines Knotens x ist die Menge aller Vorgänger- und Nachfolgerknoten von x in G. Wenn $x_j \in \mathbb{P}_x$, dann *liegt* x_j *auf* dem Aggregationspfad von Knoten x.
- Knoten $\{x_1, x_2, \ldots, x_l\} \in \mathbb{P}_x$ liegen *hintereinander* auf einem Aggregationspfad $\mathbb{P}_x$, wenn gilt $\forall i \in \{1, \ldots, l-1\} : \overrightarrow{x_i, x_{i+1}} \in E$. Das bedeutet, jeweils zwei Knoten x_i, x_{i+1} sind direkte Vorgänger beziehungsweise Nachfolger zueinander in G.
- für alle Pfade $\mathcal{P}_{x_1, x_l} = \overrightarrow{x_1, x_2, \ldots, x_l}$ in G gilt, $l = |\overrightarrow{x_1, x_2, \ldots, x_l}|$ bezeichnet die Länge von $\mathcal{P}_{x_1, x_l}$
- in G gibt es einen *längsten* Pfad $\mathcal{P}_{x', w}$ mit $|\mathcal{P}_{x', w}| \geq |\mathcal{P}_{x_i, w}|, \forall x_i \in V, \delta_{x_i} = 0$. Dieser Pfad ist von einem Blatt zur Wurzel w. Die Länge dieses Pfades heißt die *Höhe* h von G.
- ein *(Aggregations-)Teilbaum* $G' = (V', E')$ von G ist ein zusammenhängender, gerichteter, zyklenfreier Graph mit $V' \subseteq V$, $E' \subseteq E$ und Wurzel $w' \in V'$. Damit besteht jeder Aggregationsbaum $G = (V, E)$ aus der Vereinigung von Aggregationsteilbäumen: $G'_1 = (V'_1, E'_1), G_2 = (V'_2, E'_2)', \ldots, G'_i = (V'_i, E'_i)$, $E = \{E'_1 \cup E'_2 \cup \ldots \cup E'_i\}$, $V = \{V'_1 \cup V'_2 \cup \ldots V'_i\}$, $i \geq 1$. Diese Aggregationsteilbäume sind dabei nicht zwangsläufig knoten- oder kantendiskunkt.

Die Autoren von *Directed Diffusion* fordern in ihren Arbeiten Intanagonwiwat et al. [113, 114], Krishanamachari et al. [133], daß dieser Aggregationsbaum den energieeffizientesten Pfaden der einzelnen Knoten zur Senke entspricht: Aggregation wird dem Routing zur Senke gleichgesetzt. Das heißt, die einzelnen Knoten, die aus Routing-Gründen auf dem Multi-Hop-Pfad zwischen (Quell-)Sensoren und der Senke liegen, sind automatisch Aggregationsknoten. Im Beispiel aus Abbildung 2.4 sind die Knoten x und y deshalb die Aggregationsknoten von a und b, da sie laut Routing jeweils den nächsten Hop auf dem Weg zur Datensenke darstellen. Diese Annahme schränkt den Aufbau des Aggregationsbaums sehr ein und wird deshalb in dieser Arbeit in Analogie zu Heinzelman et al. [101] wie folgt erweitert. Der Aggregationsknoten x einer Menge von (Quell-)Sensoren $x_1, \ldots, x_m$ kann ein beliebiger Knoten im Sensornetz sein, der sich in Funkreichweite von $x_1, \ldots, x_m$ befindet. Es muß nicht unbedingt der laut Routing „nächste“ Knoten auf dem Weg zur Senke, sondern kann auch ein besonders *geeigneter* Knoten sein, der über besonderes Wissen oder besondere Möglichkeiten zur Aggregation verfügt. An dieser Stelle sei darauf hingewiesen, daß ein Aggregationsknoten x auch als einfacher Weiterleiter (*Forwarder*) fungieren kann, der Meßwerte seiner Quellsensoren $x_1, \ldots, x_m$ einfach nur in Richtung der Datensenke weiterleitet, ohne die Meßwerte inhaltlich zu aggregieren. Ein Beispiel für einen Forwarder wäre ein Aggregationsknoten, der die oben erwähnte Identitätsfunktion als Aggregationsfunktion einsetzt.

Es findet Kommunikation *ausschließlich* zwischen den Knoten des Aggregationsbaums statt, genauer nur zwischen Knoten und ihren Vorgängern oder Nachfolgern im Baum. Aus Sicht der Aggregation gibt es keinerlei Bedarf für Kommunikation zwischen zwei Sensoren auf gleicher Höhe im Aggregationsbaum. So haben – aus Sicht der Aggregation – die Knoten a und b, sie seien zum Beispiel Temperatursensoren an verschiedenen Stellen in Raum 1 eines Patienten, in Abbildung 2.4 kein gegenseitiges Interesse an ihren Daten. Selbiges gilt für Knoten unterschiedlicher Kategorien: Knoten b, Temperatursensor aus Raum 1, und c, Temperatursensor aus Raum 2, aus Abbildung 2.4 müssen ihre Meßwerte nicht austauschen. Gleichfalls benötigen Temperatursensoren zum Beispiel keine Daten von Sensoren anderen Typs wie Luftfeuchtigkeitssensoren. Radionuklid-Sensoren haben beispielsweise kein Interesse an seismischen Informationen usw. Ausschließlich die Aggregationsbeziehungen, die Anordnung der Knoten im Aggregationsbaum, gibt den Kommunikationsfluß vor.

Die Anordnung der einzelnen Knoten im Baum, das heißt die Konfiguration des Baumes, welcher Knoten mit welchem Aggregationsknoten kommunizieren soll, legt eine entsprechende Software-Komponente fest und soll nicht Inhalt dieser Arbeit sein. Dies lösen Protokolle wie Directed Diffusion [113, 114] oder Tiny Aggregation, siehe Madden et al. [148]. Auch die Frage, welche Knoten im Netz nun genau Aggregationsknoten werden und ob gewöhnliche, nicht-dedizierte Knoten die Aufgabe eines Aggregationsknoten übernehmen, wird in dieser Arbeit nicht behandelt. Die hier entworfenen Protokolle zum sicheren, aggregierenden Datentransport gehen davon aus, daß ein Aggregationsbaum mit Aggregationsbeziehungen existiert, die einen Kommunikationsfluß vorgeben. Dieser Transport der Daten anhand von Aggregationsbeziehungen muß abgesichert werden.

In Abbildung 2.4(b) wird eine sehr einfache Aggregation dargestellt, die vom gesamten Netz nur 7 Knoten und die Senke beinhaltet. Dadurch wird der Eindruck erweckt, alle anderen Knoten im Sensornetz nähmen an dieser Aggregation nicht teil. Das ist so zunächst korrekt, allerdings mißverständlich: Alle anderen Knoten im Netz gehören zwar nicht zur

Aggregation „Temperatur und Feuchtigkeit im Gebäude“, sind jedoch Teil anderer Aggregationen, die ihre Daten zur selben Datensenke transportieren, allerdings mit Hilfe eines anderen Aggregationsteilbaumes. Dies verdeutlicht Abbildung 2.5. Während der

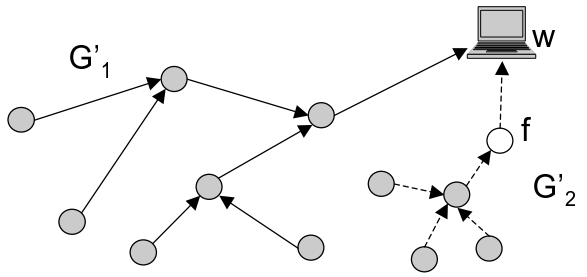


Abbildung 2.5 Ein Aggregationsbaum G aus zwei Aggregationsteilbäumen G'_1, G'_2

linke Teilbaum G'_1 wie bisher Klima im Gebäude (Temperatur und Luftfeuchtigkeit) aggregiert, aggregiert der rechte, G'_2 mit gestrichelten Aggregationsbeziehungen, beispielsweise Puls und Blutdruck am Körper des Patienten, der sich in einem Raum im Gebäude aufhält. In diesem Beispiel bilden G'_1 und G'_2 zusammen mit ihrer gemeinsamen Senke $w'_1 = w'_2 = w$ den Aggregationsbaum G mit Senke w. Alle Sensoren eines Sensornetzes sind immer Teil eines Aggregationsbaumes, wenn auch möglicherweise in unterschiedlichen Aggregationsteilbäumen. In Abbildung 2.5 enthält Teilbaum G'_2 als Beispiel noch einen Forwarder-Knoten f. Dieser Knoten leitet empfangene Daten einfach nur weiter zur Senke.

Grundsätzlich sind auch degenerierte Teilbäume möglich, in denen gar keine inhaltliche Aggregation stattfindet oder in denen ein Quellsensor direkt mit der Datensenke verbunden ist. Außerdem kann es Knoten geben, die für einen längeren Zeitraum *schlafen*. Das heißt, sie nehmen eine Weile nicht mehr am Netzwerkverkehr teil, erbringen keine Dienste etc, beispielsweise um Energie zu sparen. Auch auf diese Aspekte der Aggregation geht diese Arbeit nicht im Detail ein. Es sei nur bemerkt, daß Aggregation dynamisch ist: Aggregationsbeziehungen zwischen Knoten können sich von Zeit zu Zeit ändern.

Letztlich können im Netz auch mehrere Datensenken, das heißt mehrere Wurzeln und mehrere Aggregationsbäume existieren, die inhaltlich nicht zusammenhängen. Der Einfachheit halber sei im folgenden aber angenommen, daß die Knoten im Sensornetz ihre Daten über verschiedene Teilbäume zu *einer* Wurzel/Datensenke senden, das heißt: In *einem* Sensornetz soll *eine* Senke als Wurzel aller Aggregationsteilbäume und damit *eines* Aggregationsbaumes dargestellt werden. Dies muß, wie gesagt, nicht immer der Fall sein, hat aber auf den weiteren Verlauf der Arbeit auch keine Auswirkungen. Ein Sensornetz mit mehreren Senken kann wie eine Menge von Aggregationsbäumen betrachtet werden. Jede Senke entspricht dabei der Wurzel eines Aggregationsbaumes.

Ein Aktor, wie zum Beispiel ein Display oder ein Alarmgeber müssen auch nicht zwangläufig in die Datensenke integriert sein. Auch ein Knoten mitten im Sensornetz kann ein Aktorknoten sein – die Vorverarbeitung und Aggregation der Daten begünstig das. So könnte ein kleiner Piepser am Körper eines Patienten nicht nur Herfrequenz und Blutdruck aggregieren und weiterleiten, sondern auch gleich einen Warnton mit ausgeben.

Zusammenfassung: Annahmen über Aggregation in dieser Arbeit

In dieser Arbeit wird angenommen, daß Aggregation *einen* Baum mit *einer* Senke bildet. Innerhalb dieses Baumes werden Daten mehrfach kaskadierend aggregiert. Die dabei an-

gewendeten Aggregationsfunktionen können eine im Rahmen der Ressourcen der Sensorknoten effizient berechenbare Funktion sein. Weiterhin ist die Aggregation im Netz, die Aggregationsbeziehungen der Knoten untereinander, der komplette Aggregationsbaum, unabhängig von dieser Arbeit *gegeben* und bereits selbstständig konfiguriert. Die Wahl der Aggregationsknoten kann dabei in Erweiterung zu Intanagonwiwat et al. [113, 114], Krishanamachari et al. [133] beliebige Knoten in Funkreichweite treffen. Auch dynamisches Ändern des Aggregationsbaums sei möglich und muß von den zu entwerfenden Sicherheitsprotokollen berücksichtigt werden. Aufgrund welcher Bedingungen und wie sich der Aggregationsbaum verändert, soll allerdings nicht Teil dieser Arbeit sein.

Weiterhin wird angenommen, daß Aggregationsbäume im allgemeinen *nicht degeneriert* sind, das heißt die meisten Knoten typischerweise mehr als nur einen Nachfolger besitzen – nur dann macht Aggregation, zumindest aus energietechnischer Sicht, überhaupt Sinn. Im Beispiel des linken Aggregationsbaums aus Abbildung 2.5 besitzt jeder Aggregationsknoten genau zwei Nachfolger. Aus diesem Grund soll ein *durchschnittlicher* Aggregationsgrad $\delta > 1$ im Aggregationsbaum angenommen werden. Diese Arbeit nimmt *normalverteilte*, durchschnittliche Aggregationsgrade $d = \{2, 3, 4 \dots\}$ an. Es soll damit sehr wahrscheinlich sein, daß im Durchschnitt eine Häufung von Aggregationsgraden um einen bestimmten *erwarteten* Grad, zum Beispiel $\delta = 2, 3, 4, \dots$ existiert.

Es gibt demnach viele Knoten mit dem Aggregationsgrad δ im Aggregationsbaum sowie mit abnehmender Wahrscheinlichkeit auch seltener Knoten mit mehr oder mit weniger Nachfolgern als δ. Die Annahme normalverteilter Aggregationsgrade erscheint durchaus realistisch: Im Sensornetz werden viele Aggregationen, zum Beispiel von Temperaturwerten, Herzfunktionen usw., von einer bestimmten Größe δ sein. Es mögen durchaus Aggregationen mit weniger oder mehr Sensoren im Netz existieren, diese sind jedoch unwahrscheinlicher.

Die Höhe h eines Aggregationsbaums bezeichnet die Länge des längsten Pfades von den Blättern bis zu der Wurzel. In einem Aggregationsbaum der Gesamthöhe h mit Aggregationsgrad δ befinden sich auf jeder Ebene h' unterhalb der Wurzel im Baum im Durchschnitt $\delta^{h'}$ Knoten. Für die durchschnittliche Gesamtanzahl der Knoten n im Aggregationsbaum gilt damit die Gleichung

$$n \approx \sum_{i=0}^{h} \delta^i = \frac{\delta^{h+1} - 1}{\delta - 1}.$$

Auflösen nach h:

$$\begin{aligned} n &\approx \frac{\delta^{h+1} - 1}{\delta - 1} \\ \delta^{h+1} &\approx (\delta - 1)n + 1 \\ h + 1 &\approx \log_\delta((\delta - 1)n + 1) \\ h &\approx \log_\delta(1 + (\delta - 1)n) - 1. \end{aligned}$$

Die durchschnittliche Höhe h eines Aggregationsbaums steigt demnach logarithmisch mit der Gesamtanzahl der Knoten: $h \in O(\log n)$.

2.2.5 Dynamisches Netzverhalten

Viele Konfigurationsaspekte im Sensornetz verhalten sich dynamischer als zum Beispiel im Festnetz und ähneln anderen Netzen wie Ad Hoc- oder P2P-Netzen. Neben dem spon-

tanen Beitritt neuer Knoten kann es im Sensornetz vorkommen, daß ein nicht unwesentlicher Anteil Knoten das Netz auch unplanmäßig wieder verläßt, etwa durch Defekte, leere Batterien oder auch ausgedehnte Schlafphasen der Sensoren. Es kommt hinzu, daß einige, wenn auch wenige Sensoren, doch von Zeit zu Zeit durch neue Sensoren ersetzt werden. Die Gesamtanzahl der Knoten im Sensornetz ist, zum Beispiel im Gegensatz zum Festnetz, a priori nur schätzbar. Sensornetze können im Laufe ihres Lebens, je nach Szenario, stärker wachsen, zeitweise schrumpfen oder aber, zum Beispiel durch Ersetzen von Knoten, in etwa gleichgroß bleiben.

Diese Dynamik wird auch in den eingangs geschilderten Szenarien deutlich: Neben dem Wachsen der Netze ist es durchaus vorstellbar, daß ein Netz wieder schrumpft, zum Beispiel wenn manche Medikamente nicht mehr benötigt werden oder in bestimmten Regionen keine Atomwaffentests mehr möglich sind und sich deren Überwachung dort nicht mehr lohnt.

Aber auch ein anderer Teil der Netzkonfiguration verhält sich in Sensornetzen sehr dynamisch: Die Aufgaben, die einzelne Knoten im Netz übernehmen, können sich erweitern. Wird ein neuer Radionuklid-Sensor dem Netz hinzugefügt, muß der dazu passende Aggregationsknoten diesen zukünftig in seine Aggregatbildung mit einbeziehen. Fällt ein Aggregationsknoten, aus welchem Grund auch immer, aus, so wird er automatisch durch einen anderen Knoten ersetzt. Der Aggregationsbaum und damit die Kommunikationsbeziehungen der Knoten untereinander ändern sich.

Sicherheitsprotokolle müssen mit dieser Form der Dynamik umgehen können, und Daten müssen auch nach Konfigurationsänderungen noch sicher und wie geplant transportiert werden.

Zusammenfassung: Annahmen über Dynamik in dieser Arbeit

In Sensornetzen kann ein Anteil der Knoten spontan ausfallen und neue Knoten können spontan hinzugefügt werden. Die Kommunikation der Knoten untereinander, der Aggregationsbaum und damit die Aggregationsbeziehungen können sich ändern.

2.3 Weitere Annahmen

In den vorausgegangenen Abschnitten sind Merkmale von Sensornetzen und Annahmen erarbeitet worden, die sich für einen sicheren Datentransport noch als von besonderer Bedeutung herausstellen werden. Der Vollständigkeit halber seien jetzt noch einige weitere, für diese Arbeit *wichtige* Annahmen über Eigenschaften von Sensornetzen aufgezählt.

Das Sensornetz ermöglicht über andere Software-Komponenten bereits die Kommunikation zwischen den Knoten im Netz. Ein unsicherer Datentransport unter den Knoten ist möglich. Dazu gehört insbesondere ein funktionierendes Routing wie zum Beispiel das aus Intanagonwiwat et al. [114]. Die einzelnen Knoten können unter Zuhilfenahme dieses Routing-Protokolls beliebige andere Knoten im Netz erreichen. Häufig kann man aus Gründen der Energieeffizienz davon ausgehen, daß der optimale Routing-Pfad eines Knotens zur Senke mit den Vorgängerknoten auf seinem Aggregationspfad übereinstimmt. Treten dem Netz neue Knoten bei oder fallen alte Knoten aus beziehungsweise werden in einen längerfristigen Schlafmodus versetzt, so paßt sich das Routing dementsprechend an.

Analog sollen alle Knoten in dieser Arbeit für eine funktionierende Netzwerkkommunikation ein passendes Adressierungsschema kennen. Knoten besitzen jeweils eindeutige Adressen beziehungsweise ID's und können sich darüber gegenseitig addressieren das heißt ansprechen und Daten austauschen. Adressierung impliziert *kein* neues Sicherheitsproblem, da in dieser Arbeit paarweise Schlüssel zwischen Knoten ausgetauscht werden und die Schlüssel an die Identitäten der Knoten gebunden sind. Kommunikation zwischen Knoten und der Transport von Meßwerten in Richtung Senke ist demnach bereits möglich – diese Arbeit sichert ihn ab.

2.4 Angreifermodell

Vor der Herleitung irgendwelcher (Sicherheits-)Probleme des Datentransports in Sensornetzen muß zunächst einmal klar werden, warum und wie diese Probleme in Computernetzwerken grundsätzlich entstehen. Die Abstraktion dieser Probleme, das erwartete Fehlverhalten, wird in einem Angreifermodell festgehalten, welches Voraussetzung und Grundlage einer Sicherheitsanalyse des Systems ist.

Definition Security Policy, Angreifer und Angriff

Ein sogenannter *Angreifer* ist eine Entität, die mit Hilfe eines *Angriffs* auf ein Computernetzwerk versucht, ein *unbefugtes* Ziel zu erreichen. Unbefugt bedeutet hierbei, daß ein Computernetzwerk, genauer dessen Protokolle, laut Spezifikation dem Angreifer das Ziel eigentlich verwehrt [106]. Häufig spezifiziert eine sogenannte *Security Policy*[3], was im Netz erlaubt ist und was nicht. Der Angreifer versucht, gegen diese Policy zu verstoßen.

Ein Beispiel eines Angriffs auf den Datentransport im Sensornetz: Die Security Policy beim Szenario des betreuten Wohnens besage, daß die sensiblen Patientendaten für Außenstehende geheim bleiben sollen. Ein Angreifer will gegen diese Policy verstoßen und versucht, die im Netz per Funk transportierten Patientendaten abzuhören – ein Angriff. Zum Ausführen dieses Angriffs stehen dem Angreifer dabei eine Menge Werkzeuge zur Verfügung, unter anderem Abhörgeräte, eigene Computersysteme, Programme, physischer Zugang usw., siehe auch hierzu Howard und Longstaff [106].

Aufgabe eines Sicherheitsprotokolls

Die Aufgabe oder das Problem eines Sicherheitsprotokolls ist nun, Angriffe zu verhindern. Dabei ist es wesentlich, vor dem Entwurf eines Sicherheitsprotokolls genau festzulegen, über welche Möglichkeiten, über welche Arten von Angriffen ein Angreifer voraussichtlich verfügt und damit welche Bedrohungen für das Sensornetz existieren. Das Sicherheitsprotokoll schützt dann das Computernetzwerk gegen den so spezifizierten Angreifer.

2.4.1 Ziele des Angreifers: Angriffe und Bedrohungen

Dieser Abschnitt beschreibt die von einem möglichen Angreifer ausgehenden Bedrohungen. Das sind die Ziele, die der Angreifer im Rahmen seiner Möglichkeiten mit einem Angriff auf den Transport der Daten im Sensornetz zu erreichen versucht. Daraus wird dann im Anschluß klar, welche Probleme ein Protokoll zum sicheren Transport in Sensornetzen genau lösen muß.

Im allgemeinen lassen sich die folgenden Bedrohungen für Netzwerke erkennen, vergleiche auch Bless et al. [34], SOG-IS [211], Voydock und Kent [228]:

[3] im Deutschen: Sicherheitsstrategie

- **Abhören von Daten**
  Das Ziel des Angreifers ist es, unberechtigt an Daten zu gelangen, die im Netz ausgetauscht werden. Da das Sensornetz je nach Szenario häufig sensible und wertvolle Daten zur Senke transportiert, will der Angreifer irgendwie durch Abhören an diese Daten gelangen, um sie dann zum Beispiel zu verkaufen.

  Sämtliche Daten müssen demnach *vertraulich* durch das Sensornetz transportiert werden. Dabei ist bei dem im nächsten Abschnitt definierten, starken Angreifermodell klar, daß Daten, die ein vom Angreifer *korrumpierter* Knoten im Rahmen des normalen Datentransports laut Protokollspezifikation erhält, nicht gegen Abhören geschützt werden können. Der korrumpierte Knoten erhält diese Daten ja auf rechtmäßige, protokollkonforme Weise. Da unbekannt ist, welche Knoten korrumpiert sind und welche nicht, kann man nicht verhindern, daß korrumpierte Knoten im Rahmen des normalen Protokollablaufs an für sie bestimmte Daten gelangen. Lediglich diejenigen Daten, die den Angreifer und seine korrumpierten Knoten nicht auf protokollkonforme Weise erreichen, müssen vertraulich bleiben.

- **Modifizieren von Daten**
  Ein weiteres Ziel für den Angreifer ist das Modifizieren von Nachrichten. Gelingt es dem Angreifer, Daten in Nachrichten auf ihrem Funkweg zwischen jeweils zwei Knoten zu verändern, dann kann er so erheblich Einfluß auf die weitere Aggregation dieser Daten nehmen und die Entscheidungen, die im Netz oder an der Datensenke aufgrund dieser Daten getroffen werden, beeinflussen. Überträgt ein legitimer Knoten a einen kritischen Herzfrequenz-Wert oder einen kritischen Wert für radioaktiven Zerfall an den legitimen Knoten b, dann versucht der Angreifer während der Übertragung diesen Wert so zu modifizieren, daß bei b ein nicht-kritischer Wert ankommt.

  Im Kontext von Aggregation existiert allerdings noch eine weitere Bedrohung: Der Angreifer aggregiert falsch. Da der Angreifer auch Aggregationsknoten korrumpiert, besteht die Gefahr, daß er empfangene Daten auf diesen Knoten falsch aggregiert und ein gefälschtes Aggregat weiter in Richtung Senke leitet. Wenn ein vom Angreifer korrumpierter Knoten beispielsweise von zwei legitimen Knoten sowohl einen kritischen Blutdruck als auch eine kritische Herzfrequenz empfängt, dann kann er böswillig ein Aggregat erzeugen und im Aggregationsbaum weiterleiten, daß nicht auf den kritischen Gesundheitszustand des Patienten hinweist. Daher soll an dieser Stelle gefordert werden, daß Aggregationsknoten auch *korrekt* aggregieren müssen. Korrekt bedeutet, daß ein Aggregationsknoten ein Aggregat genau so erzeugt, wie es ihm aufgrund der von ihm empfangenen Daten und seiner Aggregationsbeziehungen auch obliegt.

  Teil der *Integrität* des Datentransportes in einem aggregierenden Netz ist nicht nur der integere[4] Transport von Daten, sondern auch die Korrektheit der erzeugten Aggregate.

- **Maskerade und Erzeugen von Daten**
  Der Angreifer probiert, sich als ein legitimer Knoten im Netz auszugeben und am normalen Protokollablauf teilzunehmen. Die Maskerade ist dabei nur die Vorbereitung, um zum Beispiel unrechtmäßig eine Nachricht mit einem Datum an einen Aggregationsknoten zu versenden und dadurch die Aggregatbildung zu fälschen.

[4] aus dem Lateinischen: integer – heil, ganz, unversehrt

Trotz des hohen Blutdrucks eines Patienten versucht der Angreifer beispielsweise sich bei einem Aggregationsknoten als Blutdrucksensor auszugeben und einen niedrigen Blutdruckwert zu melden. In einem anderen Beispiel versucht der Angreifer sich bei der Atom-Behörde in Wien als Aggregationsknoten eines seismologischen Sensornetzes auszugeben und meldet fälschlicherweise einen Atomwaffentest. Der Angreifer fälscht also seine Identität, um Nachrichten im Namen anderer Knoten zu erzeugen und zu verschicken.

Die Kommunikation muß daher im Sensornetz *authentisch* geschehen: Empfangen legitime Knoten im Netz einen Meßwert oder ein Aggregat, dann müssen sie entscheiden können, von wem es kommt. Auch hier gilt intuitiv, daß der Angreifer auf jeden Fall Nachrichten erzeugen kann, die (anscheinend) von Knoten stammen, welche er korrumpiert hat. Er darf allerdings nicht in der Lage sein, sich erfolgreich als ein Knoten auszugeben, den er zuvor nicht korrumpiert hat.

2.4.2 Charakterisierung des Angreifers

Bei Sicherheitsprotokollen für klassische Netze wie zum Beispiel die weit verbreiteten TLS [75] oder IPSec [127] beschreiben die Autoren die Möglichkeiten eines Angreifers meistens gar nicht explizit, sondern gehen implizit von einem bestimmten Angreifer aus, der ursprünglich auf einen Vorschlag aus Dolev und Yao [79] zurückgeht und häufig *Man-in-the-Middle*-Modell genannt wird: Der Angreifer kann sämtliche stattfindende Kommunikation im Netz abhören. Dabei gelingt es ihm allerdings nicht, verschlüsselte Daten zu entschlüsseln, wenn er nicht den dafür passenden Schlüssel besitzt. Genauso kann der Angreifer selbst mit anderen Knoten im Netz kommunizieren, jedoch gelingt es ihm auch hier nicht, Daten ohne passenden Schlüssel zu verschlüsseln. Bildlich gesehen stellt in diesem Modell der Angreifer bei jeder Kommunikation zwischen zwei Knoten einen „Mann in der Mitte“ dar, der alle Nachrichten, welche die beiden Knoten austauschen, zunächst abfängt, modifiziert, vielleicht durch eine eigene ersetzt und an den Empfänger weiterleitet.

Im Bereich der Sensornetze bestehen jedoch zusätzliche realistische Angriffsmöglichkeiten, die im folgenden erläutert werden.

1. Hardware-Manipulation

Ein Web-Server, File-Server oder sogar der eigene private PC eines Benutzers sind häufig in speziell „geschützten“ Umgebungen untergebracht: Das mag ein gegen unbefugten Zutritt gesichertes Rechenzentrum sein, ein abgeschlossener Server-Schrank oder die eigene, abgeschlossene Wohnung. Ein Angreifer *kann* dementsprechend häufig nur Kommunikation abhören oder selbst Nachrichten an Protokollteilnehmer verschicken – er hat allerdings keinen physischen Zugang zu Hardware.

Im Gegensatz zu klassischen Servern sind Sensoren meist ungeschützt in der Öffentlichkeit beziehungsweise an für jeden einfach zugänglichen Orten installiert. Aus Kostengründen ist die Hardware der Sensoren meistens ungeschützt und nicht *tamper-proof*[5]. Damit befindet sich ein Angreifer in der Lage, *physisch* auf Sensoren zuzugreifen. Durch den physischen Zugriff kann der Angreifer versuchen, die Hardware selbst „anzugreifen“, das heißt, mit einfachen technischen Hilfsmitteln den gesamten Speicher eines Sensors auszulesen. Der Angreifer kennt dadurch sämtliche Geheimnisse, die auf einem Sensor gespeichert sind, das komplette Programm, welches der Sensor ausführt usw. Weiterhin kann der

[5] im Deutschen: sicher gegen Manipulation

Angreifer den Sensorknoten re-programmieren: Er ersetzt dazu das auf dem Sensor bisher vorhandene Programm durch sein eigenes. Der Sensor führt dann dementsprechend die Befehle des Angreifers aus, er ist in der Gewalt des Angreifers [175]. Kurz: Dieser Sensorknoten ist *korrumpiert*. Sobald ein Angreifer einen oder mehrere Knoten korrumpiert, können diese zusammenarbeiten und gegenseitig Informationen austauschen, um weitere Ziele des Angreifers zu verfolgen.

Das Korrumpieren eines Sensorknotens ist aufgrund seiner simplen, Mikrocontroller-basierten Hardware relativ einfach, wie in Becher et al. [16] bereits gezeigt. Die Autoren geben beim Einsatz eines handelsüblichen *JTAG*-Kabels sowie frei verfügbarer Software eine Zeit von unter 5min für das Korrumpieren eines MICA-Knotens an. Das Dolev-Yao Modell muß also für den Bereich Sensornetze erweitert werden.

2. Korrumpieren durch Würmer

Der Vollständigkeit halber sei noch auf eine besonders häufig im Internet vorkommende Form der Korrumpierung eingegangen, obwohl sie für Sensornetze, wie der folgende Abschnitt zeigt, nicht besonders realistisch scheint:

Laut Statistik des BSI ist die im Internet am häufigsten vorkommende Form der Knoten-Korrumpierung die durch sog. „Würmer“ [40]. Bei einem Wurm handelt es sich um ein vom Angreifer geschriebenes Programm, das eine fehlerhafte Implementierung zum Beispiel im Netzwerk-Stack eines (Opfer-)Computersystems ausnutzt, um sich auf diesem Computersystem über das Netzwerk zu installieren und sich dann weiter auf andere Computersysteme zu verbreiten [186]. Würmer basieren häufig darauf, daß sich durch einen sogenannten *Buffer-Overflow* der Programmcode des Wurms in den Speicher (zum Beispiel Stack [5]) des Computersystems installiert und dort ausgeführt wird. Eine andere Möglichkeit liegt darin, Variablen im Hauptspeicher so zu überschreiben, daß sich der normale Programmablauf des Computers verändert.

Solche Angriffe durch Würmer lassen sich auf Sensornetze aus zwei Gründen schwer übertragen: Die Hardware-Architektur typischer Sensorknoten unterscheidet sich fundamental von der üblichen im Internet. Die Mikrocontroller-basierten Sensorknoten verfügen über eine Harvard-Architektur, die Datenspeicher (RAM) und Programmspeicher (Flash-ROM) voneinander trennt. Das Installieren von schadhaftem Code im Flash des Sensors nur mit Hilfe eines Buffer-Overflows gestaltet sich dadurch als extrem schwierig. Der Sensor müßte dazu gezwungen werden, Teile des Hauptspeichers in den Flash-Speicher zu kopieren und dann auszuführen. Außerdem sind Applikationen und auch Betriebssysteme für Sensoren wie TinyOS besser bzgl. möglicher Buffer-Overflows zu *verifizieren*: Im Gegensatz zu Applikationen und Betriebssystemen auf PCs ist Quellcode für Sensorprogramme deutlich überschaubarer und damit in sicherheitskritischen Szenarien einfacher zu verifizieren. Moderne Compiler unterstützen den Entwickler außerdem automatisch mit Schutzmaßnahmen gegen Buffer-Overflows.

Obwohl das Korrumpieren von Sensorknoten durch Buffer-Overflows unrealistisch erscheint, mag es einem Angreifer durchaus gelingen, dadurch einen Teil der Knoten im Netz unter seine Kontrolle zu bringen. Derartige Angriffe sind jedoch bis heute noch nicht bekannt geworden.

An dieser Stelle soll noch erwähnt sein, daß Forschungsarbeiten existieren, die rein Software-seitig versuchen, die Programmausführung auf Prozessoren resistent gegen das Korrumpieren eines Angreifers zu machen, vergleiche zum Beispiel Abadi und Feigenbaum

[1], Sander und Tschudin [199]. Die Arbeiten basieren dabei auf der Idee, die Ausführung eines Programmes selbst zu chiffrieren, so daß der Angreifer keinen gezielten Eingriff auf die Programmausführung durchführen kann. Allerdings schlagen die Forschungsarbeiten noch keine konkreten Algorithmen zur Implementierung dieser Idee vor, und bisher eignen sich diese Ansätze nur für Programme, die boolesche Ausdrücke oder Polynome evaluieren. Weiterhin basieren die Ideen auf der für Sensor-Hardware teuren Exponentiation in endlichen Körpern.

Erweiterung des Dolev-Yao Modells

Diese Arbeit nimmt, egal ob durch physischen Zugang, Würmer oder ähnlichem, an, daß der Angreifer durchaus in der Lage ist, Sensorknoten zu korrumpieren. Man spricht dann davon, daß korrumpierte Knoten sich *byzantinisch* verhalten, siehe Lamport et al. [136]. Um die Fähigkeiten des Angreifers in diesem Fall genauer zu spezifizieren, folgt nun eine Modellierung beziehungsweise Charakterisierung der Eigenschaften des Angreifers nach Cramer und Damgård [65]. Die einzelnen Unterpunkte stellen dabei wieder Annahmen dar, die in dieser Arbeit der Plausibilität halber getroffen worden sind. Es mag – wie immer – andere Szenarien von Sensornetzen geben, die über andere Annahmen verfügen. So nehmen beispielsweise die bisher einzigen beiden Arbeiten, in denen ein Angreifer in Sensornetzen ähnlich formal spezifiziert wird, Ács et al. [4], Benenson et al. [22], auf Grund einer anderen Aufgabenstellung deutlich schwächere, *lokal begrenzte* Angreifer an.

- Die Angreifer beziehungsweise seine korrumpierten Sensoren verhalten sich *aktiv*. Sie können nicht nur Nachrichten, die für sie verschlüsselt worden sind, entschlüsseln und lesen, sondern sie versenden auch neue Nachrichten. Die korrumpierten Knoten können sich dabei zunächst eine Weile lang wie *legitime*, das heißt nichtkorrumpierte Knoten, protokollkonform verhalten, bevor sie dann (zwischendurch) tatsächlich etwas Bösartiges machen wie zum Beispiel ein Aggregat falsch berechnen. Daher wird an dieser Stelle auch angenommen, daß es den legitimen Knoten nicht ohne weiteres möglich sei zu erkennen, ob ein bestimmter Knoten korrumpiert worden ist oder nicht.

- Die korrumpierten Knoten arbeiten zusammen, um das Ziel des Angreifers zu erreichen. Dabei können die korrumpierten Knoten untereinander Daten austauschen. Die dafür notwendige Kommunikation zwischen den korrumpierten Knoten muß nicht zwangsläufig über protokollkonforme Nachrichten und die „normale“ Funkschnittstelle der Sensoren erfolgen, sondern *irgendwie* – beispielsweise über Out-of-Band-Mechanismen. Die Kommunikation der korrumpierten Knoten untereinander ist instantan: Sie benötigt keine Zeit. Sinn und Zweck dieser beiden Annahmen ist nur, daß der Angreifer und alle seine Knoten über einen gemeinsamen Wissensstand verfügen. Kennt ein korrumpierter Knoten ein bestimmtes Geheimnis, etwa einen Schlüssel, dann kennen ihn gleichzeitig auch alle anderen korrumpierten Knoten im Netz.

  Diese Annahme ist durchaus realistisch: Der Angreifer kann zum Beispiel mit Hilfe besonderer Funktechnik, die wesentlich stärker, schneller und effizienter als die der Sensorknoten ist, seine Knoten untereinander synchronisieren.

- Der Angreifer ist nicht auf einen Ort im Netz lokal beschränkt, sondern *global* im Netz vertreten. Er kann sämtliche Nachrichten, die zwischen Sensoren ausgetauscht

werden, abhören, allerdings nur dann entschlüsseln, wenn einer seiner korrumpierten Knoten den dafür notwendigen Schlüssel kennt. Diese Annahme ist deshalb plausibel, weil die *drahtlose* Funkkommunikation zwischen jeweils zwei Sensoren mit einfachen technischen Mitteln abgehört werden kann.

Genauso kann der Angreifer beliebig im Netz neue Nachrichten injizieren, allerdings erzeugt er nur Chiffrate mit Hilfe von Schlüsseln, die seine korrumpierten Knoten kennen.

Weiterhin kann der Angreifer keine Nachrichten, zu denen er nicht den Schlüssel kennt, modifizieren. Modifiziert soll an dieser Stelle bedeuten, daß er nicht in der Lage ist, durch Modifikation des Chiffrats bewußt den sich dahinter verbergenden Klartext zu verändern. Grundsätzlich kann der Angreifer in jedem Fall eine Nachricht verändern, beispielsweise durch gezieltes Stören eines Teils der Nachrichtenübertragung einzelne Bits der Nachricht und damit einzelne Bits eines Chiffrats manipulieren. Aufgrund des Avalanche-Effekts (Lawinen-Effekt) [235] kann der Angreifer damit allerdings *nicht gezielt* Bits des Klartextes manipulieren. Genauer: Für jeweils ein Bit, daß der Angreifer im Chiffrat kippt, kippt jedes Bit im Klartext mit der Wahrscheinlichkeit $\frac{1}{2}$. Außerdem kann der Empfänger eines Chiffrates mit bestimmten Techniken überprüfen, ob ein Angreifer das Chiffrat modifiziert hat – dazu später in Abschnitt 2.6.2.2 mehr.

Im allgemeinen nennt man das Angreifermodell, in dem ein Angreifer keine Daten lesen, generieren oder verändern kann, zu denen er keinen Schlüssel hat, *Kryptographisches Modell* [65].

Der Angreifer ist bzgl. der ihm zur Verfügung stehenden Rechenleistung *polynomial gebunden*. Er mag zwar über sehr viel Rechenleistung verfügen, deutlich mehr Rechenleistung im Vergleich mit den Sensoren, allerdings kann er damit keine sichere Chiffre in weniger als exponentieller Zeit in Abhängigkeit der Schlüssellänge „brechen“ oder eine Hash-Funktion in weniger als exponentieller Zeit „invertieren“.

Die letzten beiden Annahme sind eigentlich selbstverständlich für mögliche, in der Realität vorkommende Angreifer. Ein Angreifer mag über viel Rechenleistung verfügen, er kann allerdings keine Wunder vollbringen: Um ein Chiffrat zu entschlüsseln, zu erzeugen oder zu modifizieren, braucht er den Schlüssel. Ohne Schlüssel ist der Rechenaufwand zum Entschlüsseln zu hoch, siehe unter anderem Menezes et al. [156].

- Der angenommene Angreifer verhält sich bzgl. der Korrumpierung von Knoten nicht adaptiv, sondern *statisch*: Er korrumpiert eine bestimmte Anzahl $\mathcal{B}$ von Knoten im Netz auf einmal und *vor* dem Beginn der normalen Arbeit im Sensornetz (Daten messen, verschicken, aggregieren, etc.). Der Angreifer korrumpiert solange und so viele Knoten, bis seine Ressourcen zum Korrumpieren erschöpft sind, siehe dazu den nächsten Abschnitt. Man könnte sich auch vorstellen, daß dem Angreifer zum Korrumpieren der Knoten im Netz nur eine begrenzte Zeit zur Verfügung steht, in der er seinem Treiben *unentdeckt* nachgehen kann. Irgendwann ist der Angreifer fertig mit dem Korrumpieren. Danach kann er oder will er keine Knoten mehr korrumpieren.

  Das bedeutet, der Angreifer ist insbesondere nicht in der Lage, während der Ausführung eines Protokolls *neue* Knoten zu korrumpieren. Diese Annahme wird üblicherweise auch deshalb getroffen, damit der Angreifer aus Erkenntnissen über den

bisherigen Protokollverlauf, zum Beispiel den bisherigen Nachrichtenfluß zwischen verschiedenen Knoten, etwas über die Wichtigkeit bestimmter Knoten im Netz lernt und daraufhin gezielt die „wichtigen“ Knoten korrumpiert. Ein Angreifer, der beispielsweise mittels Analyse von Datenflüssen im Netz die Aufgaben verschiedener Knoten im Netz analysieren und daraufhin gezielt wichtige Knoten korrumpieren könnte, wäre äußerst unfair: Er würde genau die Knoten korrumpieren, die für das Erreichen seiner Ziele notwendig wären. Gegen solch einen Angreifer kann *kein* Sicherheitsprotokoll sinnvoll schützen.

Diese Arbeit nimmt an, daß der Angreifer nicht weiß, welche Knoten im Sensornetz zu einem späteren Zeitpunkt wichtig sein werden und welche nicht. Der Angreifer bewegt sich völlig *zufällig* und *wahllos* durch das Sensornetz und korrumpiert dabei ebenso zufällig Knoten. Da alle Knoten gleich leicht vom Angreifer zu erreichen sind und kein Knoten über besondere Schutzmaßnahmen verfügt, korrumpiert der Angreifer Knoten *gleichverteilt*. Er macht dies solange, bis seine Ressourcen erschöpft sind. Wenn der Angreifer fertig mit dem Korrumpieren ist, hat er insgesamt $\mathcal{B}$ Knoten korrumpiert.

Die Annahme, daß alle Knoten völlig ungeschützt sind, ist wieder eine starke Annahme: In der Realität mögen in einigen Szenarien ein paar Knoten durchaus tamper-proof sein. Dennoch wird in dieser Arbeit die Annahme getroffen, daß kein Sensor speziell geschützt ist. Ein Protokoll, daß bei dieser Annahme sicher ist, ist erst recht in Netzen mit tamper-proof Hardware sicher.

In den später folgenden Untersuchungen zur Sicherheit der Protokolle zum Schlüsselaustausch und zur authentischen Datenaggregation geht diese Arbeit demnach davon aus, daß der Angreifer von allen n Knoten insgesamt $\mathcal{B}$ zufällig korrumpiert hat – und *dann* wird untersucht, welche Implikationen dadurch auf das Hinzufügen eines neuen Knotens zum Netz (Schlüsselaustausch) beziehungsweise die Durchführung der Aggregationen im Netz (authentische Datenaggregation) auf die Sicherheit entstehen.

- Die Kommunikation im Sensornetz findet (auf Applikationsebene) *synchron* statt. Synchron bedeutet hier, daß der Angreifer nicht in der Lage ist, Nachrichten zwischen zwei Knoten unendlich lange zu „blockieren“. Der Einfachheit halber soll angenommen werden, daß Knoten, die auf den Empfang einer Nachricht warten, irgendwann über einen Timer darüber informiert werden, daß die Nachricht nicht angekommen ist. Umgekehrt merken Knoten, die eine Nachricht versendet haben über einen ablaufenden Timer, daß eine erwartete Quittierung der Nachricht ausbleibt. Die Nachricht kann dann neu geschickt oder ein „Alarm“ ausgelöst werden. Eine Nachricht, die Knoten a an Knoten b schickt, kommt – wenn auch zeitverzögert – immer bei b an.

  Der Angreifer kann allerdings das Senden und Empfangen einer Nachricht voneinander entkoppeln und Nachrichten „abfangen“: Mit Hilfe geeigneter Funktechnik ist er in der Lage, die von a versendete Nachricht zu empfangen, gleichzeitig aber mittels eines Störsenders (sogenanntes *Jamming*) b am Empfang zu hindern. Bevor er nun die abgefangene Nachricht von a an b ausliefert, kann er noch schnell eine eigene Nachricht an b oder einen anderen Knoten verschicken. Ein Angreifer mit solchen Möglichkeiten heißt *rushing*.

- Eine ähnliche Annahme betrifft die Broadcast-Eigenschaft der Funkkommunikation. Wenn die Knoten a, b und c sich alle in gegenseitiger Funkreichweite befinden, dann erlaubt normalerweise das drahtlose Medium, daß eine Nachricht, die a an b verschickt, auch gleichzeitig von c mitgehört werden kann. In dieser Arbeit soll allerdings angenommen werden, daß der Angreifer, wiederum durch den Einsatz besonderer Funktechnologie, in der Lage ist, die Nachricht von a so zu stören oder abzuschirmen, daß b die Nachricht empfängt, Knoten c allerdings nicht. Weiterhin soll es dem Angreifer möglich sein, eine Nachricht eines seiner korrumpierten Knoten a' so an einen legitimen Knoten b zu verschicken, daß ein anderer legitimer Knoten c davon nichts mitbekommt – obwohl sich auch c in Funkreichweite von a' befindet.

2.4.3 Anzahl korrumpierter Knoten $\mathcal{B}$

Wenn man zuläßt, daß der Angreifer eine bestimmte Anzahl $\mathcal{B}$ von Knoten im Netz korrumpiert, muß man diskutieren, wie groß $\mathcal{B}$ sein kann. Zunächst ist klar, daß $\mathcal{B} < n$ gelten muß, also nicht alle Sensorknoten im Netz korrumpiert sein dürfen. In einer Situation mit $\mathcal{B} = n$ kann es keine sinnvolle Sicherheit geben.

Die tatsächliche Anzahl korrumpierter Knoten $\mathcal{B}$ ist durch die Fähigkeiten des Angreifers, Knoten zu korrumpieren, beschränkt. Obwohl zum Beispiel Becher et al. [16] zeigt, daß es durchaus in relativ kurzer Zeit und mit einfach Mitteln gelingt, Sensoren zu korrumpieren, *kostet* das Korrumpieren von Knoten den Angreifer bestimmte Ressourcen: beispielsweise Zeit und Geld. Der Angreifer braucht Zeit und Ausrüstung, um Knoten zu korrumpieren, er benötigt Zeit, um einen Buffer-Overflow zu entwickeln usw. Über derartige Ressourcen verfügt der Angreifer allerdings nur beschränkt und kann dementsprechend nicht beliebig viele Knoten korrumpieren. Weiterhin muß der Aufwand, den der Angreifer betreibt, um Knoten zu korrumpieren, im Verhältnis mit dem zu erreichenden Ziel stehen: Wenn der Angreifer aufgrund der Durchführung seines Angriffs zum Beispiel einen bestimmten Geldbetrag M erhalten kann, dann wird er sicherlich nicht mehr Zeit und Geld in den Angriff investieren als M – die Rationalität des Angreifers vorausgesetzt.

Weiterhin muß die Gesamtanzahl der Knoten im Netz n nicht zwangsläufig einen Einfluß auf die Zahl $\mathcal{B}$ haben. In einem Netz mit $n = 10000$ Knoten ist es zum Beispiel nicht doppelt so „einfach“, Knoten zu korrumpieren, wie in einem Netz mit $n = 5000$ Knoten: Das große Netz mit 10000 Knoten ist beispielsweise über eine viel größere geographische Distanz verteilt als das kleinere, so daß der Aufwand zum Korrumpieren mehrerer Knoten für den Angreifer nicht unbedingt geringer wird. Analog: Im Netz mit 10000 Knoten existieren unter Umständen 1000 Sensoren mit Firm- und Software Version 1, 5000 mit Version 2 und die „neuesten“ 3000 Sensoren verfügen bereits über die neueste Firmware 3. Um „mehr“ Sensorknoten zu korrumpieren, muß der Angreifer auch „mehr“ Ressourcen einsetzen.

Die Anzahl $\mathcal{B}$ der korrumpierten Knoten im Netz bestimmt in dieser Arbeit folglich alleine der Angreifer aufgrund seiner zur Verfügung stehenden Ressourcen.

Im weiteren Verlauf der Arbeit bezeichnet $\beta = \frac{\mathcal{B}}{n}$ den Anteil oder Prozentsatz der korrumpierter Sensorknoten im gesamten Netz. Wenn sich der Angreifer aus den oben beschriebenen Gründen wahllos durch das Netz bewegt und damit zufällig und gleichverteilt Knoten korrumpiert, dann ist jeder Knoten im Netz mit $\beta\%$ Wahrscheinlichkeit korrumpiert. Bei der angenommenen Gleichverteilung der korrumpierten Knoten gilt weiterhin:

In jeder Teilmenge von $n' < n$ Knoten aus dem Netz sind im Durchschnitt $k = \beta \cdot n'$ Knoten korrumpiert und $(1 - \beta) \cdot n'$ legitim.

2.4.4 Annahmen über Blätter und Senke

Diese Arbeit nimmt weiter an, daß der Angreifer die Senke beziehungsweise Basisstation des Sensornetzes nicht korrumpieren kann. Falls es dem Angreifer gelingen sollte, die Senke zu korrumpieren, könnte er sämtliche die Senke erreichenden Aggregate dem Benutzer gegenüber fälschen. Alle Informationen, die der Benutzer aus dem Sensornetz über die Senke bezieht, wären falsch. Genauso: Im Beispiel des betreuten Wohnens würde das Display als Datensenke ständig falschen Alarm schlagen können, völlig unabhängig von dem, was genau im Sensornetz passiert. Selbst wenn der Angreifer nur die Senke korrumpiert, macht die gesamte Sicherheit im Sensornetz in einer solchen Situation keinen Sinn mehr. Die Senke im Sensornetz wird daher in dieser Arbeit als nicht korrumpiert angenommen.

Blätter und Meßwerte

Etwas ähnliches gilt für die Blätter im Aggregationsbaum. Ein Sensor, der ein aus Protokollsicht transzendentes, externes Phänomen oder Ereignis mißt, *kann* diesbzgl. alleine mit Mechanismen aus dem Protokoll heraus nicht überprüft werden. Dies ginge nur dann, wenn mehrere Sensoren das gleiche Phänomen beobachten und sich dadurch, beispielsweise über einen Mehrheitsentscheid, gegenseitig überprüfbar machen. Läßt man allerdings allgemein zu, daß ein einzelner Sensor alleine ein Phänomen beobachtet, dann kann *kein* Sicherheitsprotokoll die Meßwerte dieses Sensors überprüfen. Der Angreifer muß den Sensor dazu noch nicht einmal korrumpieren, es ist vielfach einfacher für den Angreifer, die Meßwerte selbst zu fälschen: Beispielsweise kann der Angreifer den Meßwert eines Temperatursensors fälschen, in dem er ein Feuerzeug neben den Sensor hält. Kein Sicherheitsprotokoll kann gegen einen solchen Angriff schützen. Es können nur sich daraus ergebende Aggregationen überprüft werden.

Alle *Blätter* im Aggregationsbaum, das sind die Sensoren, die Meßwerte von „außerhalb“ des Protokolls messen, dürfen in dieser Arbeit vom Angreifer entweder gar nicht oder nur *eingeschränkt* korrumpiert werden, das heißt: Im Kontext sicherer Aggregation dürfen sie in Bezug auf ihre gemessenen Werte nicht *lügen*, weil dies kein Protokoll überprüfen kann. Mißt ein Blatt x einen Wert X, dann muß x auch genau diesen Wert X an seinen Aggregationsknoten schicken. Es soll damit dem Angreifer nicht möglich sein, „protokollexterne“ Daten zu fälschen. Ansonsten verhalten sich aber auch korrumpierte Blätter wie alle anderen korrumpierten Knoten, versuchen fremde Nachrichten abzuhören, zu modifizieren usw. Nur das Lügen bezüglich ihrer Meßwerte ist verboten. Hierbei mag es sich um eine unnötig starke Annahme handeln, die jedoch aus obigen Gründen notwendig wird. Ließe man lügende Blätter oder durch den Angreifer gefälschte Meßwerte zu, dann können die in dieser Arbeit entwickelten Lösungen – und auch alle anderen – nur diese gefälschten Meßwerte sicher aggregierend durch das Sensornetz transportieren. In dieser Arbeit sind Blätter nicht korrumpiert, bzw. sie betrügen nicht in Hinsicht auf die Authentizität ihrer Meßwerte.

Die Problematik mit Blättern und falschen, protokollexternen Meßwerten ist nur im Kontext der sicheren Aggregation in Kapitel 4 ab Seite 109, von Bedeutung. Für die Schlüsselverteilung in Kapitel 3 wird angenommen, daß korrumpierte Blätter sich exakt genauso wie andere korrumpierte Knoten verhalten.

2.4.5 Denial-of-Service

Unter einer „Denial-of-Service“-Attacke versteht man einen Angriff auf die Verfügbarkeit eines Dienstes oder einer Diensteigenschaft (zum Beispiel Verarbeitungsdauer eines Dienstes), die ein (Sensor-)Netzwerk erbringen kann. Das Ziel einer solchen Attacke ist, daß ein legitimer Knoten im Netzwerk einen Dienst nicht mehr in Anspruch nehmen kann. Dabei muß der angegriffene Knoten nicht korrumpiert sein, sondern es gelingt dem Angreifer einfach, dessen normale Diensterbringung zu unterbinden. Ein Beispiel für einen erfolgreichen DoS-Angriff auf einen Aggregationsknoten wäre, wenn es dem Angreifer gelingen würde, daß der Aggregationsknoten keine Daten mehr entgegennimmt, daraus kein Aggregat mehr bildet und dies auch nicht weiterschickt.

Grundsätzlich kann man dabei zwischen zwei verschiedenen DoS-Kategorien unterscheiden, siehe zum Beispiel van Tilborg [219]: dem Ausnutzen von Implementierungsschwächen sowie dem Verbrauchen von Ressourcen. Beim ersteren nutzt der Angreifer einen Fehler in der Implementierung eines Protokolls aus, um ein System an seiner Diensterbringung zu hindern. Ein sehr prominentes Beispiel hierfür war der *Ping-of-Death* von 1996 [45], bei dem aufgrund fehlerhafter Netzwerk-Stacks einige Betriebssysteme mit Hilfe spezielle präparierter ICMP-Echo Nachrichten zum Absturz gebracht werden konnten. Ähnlich wie bei Buffer-Overflows lassen sich solche Protokollschwächen allerdings durch sorgfältigen Entwurf und Implementierung vermeiden.

Eine wesentlich größere Gefahr für Sensornetze geht von der zweiten Kategorie von DoS-Angriffen aus: dem Verbrauch von Ressourcen. Hier versucht der Angreifer, bestimmte, endliche Ressourcen so zu verbrauchen, daß diese für legitime Diensterbringung nicht mehr zur Verfügung stehen. Als einfaches Beispiel läßt sich hier das *Jammen*, Blockieren, des Kommunikationsmediums nennen. Mit relativ einfachem Aufwand kann der Angreifer die Ressource „Funk“ komplett belegen, so daß legitime Knoten nicht mehr miteinander kommunizieren können. Ein anderer, einfacher Angriff, gerade in Sensornetzen, ist die sogenannte *Sleep-Deprivation Torture* aus Stajano und Anderson [216]: Der Angreifer versucht ständig, Dienste eines Sensors in Anspruch zu nehmen. Auf diese Weise wird verhindert, daß dieser Sensor sich in einen Energie-Schlafmodus versetzt und dadurch seine Batteriereserven schnell aufbraucht. Damit steht er schließlich für legitime Diensterbringung nicht mehr zur Verfügung.

Gegen diese Art von DoS-Angriffen, insbesondere Jamming-Angriffe, kann sich ein Sensornetz nur schwer „verteidigen“. Die Arbeiten Xu et al. [247, 248] haben Jamming-Attacken auf MICA2 Knoten untersucht und kommen zu dem Ergebnis, daß Jamming-Attacken nicht nur einfach durchzuführen und schwer festzustellen sind, sondern auch keine Verteidigungsmöglichkeiten dagegen existieren. Wood und Stankovic [245] geben eine allgemeine Übersicht, gegen welche bekannten DoS-Attacken Sensornetze besonders anfällig sind. In Deng et al. [72] analysieren die Autoren spezielle Angriffe, sogenannte Path-based-DoS-Attacken, durch die Sensornetze ebenso besonders gefährdet sind.

Da die Sensorknoten weiterhin häufig an öffentlichen und leicht zugänglichen Plätzen montiert sind, liegt ein weiterer realistischer DoS-Angriff auf einen Sensorknoten in der physischen Zerstörung des Sensors: Der Angreifer zerstört den kleinen Sensorknoten ganz einfach. Mit Sicherheit existieren noch viele weitere DoS-Angriffe dieser Art.

Zusammenfassung

Man kann demnach festhalten, daß sich gerade Sensornetze als anfällig gegenüber DoS-Angriffen erweisen und keine Möglichkeiten zur Verteidigung existieren.

Weiterhin gilt, daß sich jede *ausgeklügeltere* DoS-Attacke als Jamming, zum Beispiel ein selektiver Angriff auf bestimmte Dienste oder Teilaspekte eines Systems, durch das einfache Jammen der Funk-Kommunikation im Netz „emulieren“ läßt: Jedes Sensornetz und jedes Sicherheitsprotokoll ist immer anfällig gegen DoS-Attacken, mindestens durch Jamming.

Diese Arbeit schließt daher DoS-Attacken, Angriffe auf die Verfügbarkeit, als Ziel des Angreifers grundsätzlich aus.

2.5 Herleitung der Unterprobleme

Betrachtet man nun die in dieser Arbeit angenommenen Sensornetze zusammen mit dem vorgestellten Angreifermodell, mit den Möglichkeiten und den Zielen des Angreifers, dann läßt sich sofort die Problemstellung motivieren, der sich diese Arbeit widmet: Das Sensornetz transportiert sensible Meßwerte von den Blättern im Aggregationsbaum zur Senke und aggregiert die Meßwerte dabei mehrfach. Meßwerte und daraus entstehende Aggregate sollen in Bezug auf Vertraulichkeit, Integrität und Authentizität gegenüber einem Angreifer gesichert werden.

Diese Aufgabe läßt sich in zwei Teilprobleme unterteilen:

1. Den *Schlüsselaustausch* zwischen Knoten. Damit Knoten überhaupt „sicher“ miteinander kommunizieren können, benötigen sie kryptographische Schlüssel. Diese Schlüssel müssen die Knoten vor ihrer Kommunikation zunächst sicher untereinander austauschen.

2. Ein *authentischer Datentransport* stellt sich als zweites Problem dar, weil alleine mit Hilfe ausgetauschter Schlüssel noch keine Authentizität in einem aggregierenden Sensornetzwerk garantiert werden kann. Durch die aggregierende Form des Datentransports sind neben dem Schlüsselaustausch noch weitere Schritte für Authentizität notwendig.

2.5.1 Schlüsselaustausch

Die fundamentale Voraussetzung für jedwede sichere Kommunikation zwischen mehreren Kommunikationspartnern ist eine zwischen ihnen vorab etablierte „Sicherheitsbeziehung“, ein gemeinsamer Vertrauensanker. Ohne eine solche Sicherheitsbeziehung kann keine vertrauliche, integere oder authentische Kommunikation möglich sein, siehe hierzu die Diskussionen in Alkassar et al. [6], Anderson [8], Barak et al. [14], Beth und Desmedt [24].

Üblicherweise kommen kryptographische Schlüssel als solche gemeinsamen Vertrauensanker zum Einsatz. Kryptographische Schlüssel müssen allerdings zunächst an die einzelnen Sensoren verteilt werden beziehungsweise die Knoten müssen Schlüssel untereinander austauschen – man spricht von Schlüsselverteilung oder Schlüsselaustausch. Da der Begriff Schlüssel*verteilung* häufig die Verteilung von Schlüsseln durch *zentrale Infrastrukturkomponenten* impliziert, eignen sich in den hier angenommenen Sensornetzen nur Schlüssel*austausch*-Protokolle zum Etablieren von Schlüsseln.

Beim Austausch von Schlüsseln als Voraussetzung weiterer Sicherheit ist zu beachten, daß auch der Vorgang des Austausches selbst, das Schlüsselaustauschprotokoll, sicher

in Gegenwart des spezifizierten Angreifermodells ablaufen muß. Den kritischen Punkt stellt hier diejenige Situation dar, in der ein neuer Knoten i dem Sensornetz beitritt, in dem der Angreifer jedoch $\mathcal{B}$ Knoten korrumpiert hat. Will i nun mit anderen Knoten im Netz Schlüssel austauschen, so muß das Schlüsselaustauschprotokoll sicherstellen, daß der Angreifer mit Hilfe seiner korrumpierten Knoten den Austausch neuer Schlüssel weder abhört noch speziell beeinflußt: Der Angreifer darf keinen der von i auszutauschenden Schlüssel kennen. Es versteht sich bei dem angenommenen Angreifer jedoch von selbst, daß, falls i (protokollkonform) Schlüssel mit bereits korrumpierten Knoten austauchen möchte, der Angreifer auch in deren Kenntnis gelangt.

Schlüsselaustausch ergibt Basissicherheit

Nachdem Sensoren Schlüssel untereinander ausgetauscht haben, können sie mit deren Hilfe sicher kommunizieren: Es existiert eine Art *Basissicherheit* zwischen jeweils zwei Knoten. Basissicherheit bedeutet, daß jeweils zwei Knoten *vertraulich*, *integer* und *authentisch* miteinander kommunizieren können [43].

Kapitel 3 widmet sich im Detail den speziellen Problemen von Schlüsselaustausch, insbesondere im Kontext von Sensornetzen und entwirft das Protokoll SKEY, das einen effizienten Schlüsselaustausch in aggregierenden Sensornetzen ermöglicht.

2.5.2 Authentischer Datentransport

Aggregation hat im Sensornetz Auswirkungen auf die Sicherheit des Datentransports: Intuitiv ist klar, daß die Senke bei dem von Knoten z in Abbildung 2.4 empfangenen Aggregat, der Durchschnittstemperatur des Gebäudes, ohne weitere Informationen keinerlei Möglichkeiten zu dessen Überprüfung besitzt. Weder ist es ihr möglich zu verifizieren, ob die Temperatur von x, y und z jeweils nicht nur korrekt aggregiert worden ist, sondern auch, ob tatsächlich die Sensoren a, b, c und d dafür ursprünglich verantwortlich waren. Die Senke kann nicht überprüfen, ob das von z empfangene Aggregat *authentisch* ist.

Zwischen der Aggregation von Daten und deren Authentizität liegt ein *Widerspruch*: Alleine durch den Empfang des Aggregats von z kann die Senke dessen Korrektheit nicht überprüfen, dazu braucht sie weitere Informationen. Diese zusätzlichen Informationen müssen aber von x und y an die Senke geschickt werden. Durch das Versenden der zusätzlichen Informationen an die Senke wird aber die Idee der Aggregation umgangen, die ja gerade genau das vermeiden soll. Würden auch x und y ihre Daten direkt an die Senke schicken, dann wäre die Aggregation „aufgehoben“.

Dies wird insbesondere dann deutlich, wenn die Aggregationsfunktion f_z eines Knotens z einer Hash-Funktion, einem perfektem „Random Oracle“ [19], entspricht, das aus der Eingabe ein Aggregat bestehend nur aus einem einzelnen Bit berechnet: $\mathrm{agg}_z = f_z(X, Y, \dots) = H(X, Y, \dots) \in \{0, 1\}$. Weiterhin gelte, daß $X, Y, \dots \in \{0, 1\}^*$ die ursprünglichen Werte der zur Aggregation beitragenden Knoten x, y genau wie in Abbildung 2.4 seien. Knoten z schickt nun agg_z an die Senke des Aggregationsbaums, die agg_z überprüfen will. Damit die Senke aber das einzelne Bit agg_z auf Korrektheit überprüfen kann, benötigt sie *alle* (beiden) Werte X, Y von x und y. Um weiterhin zu überprüfen, ob tatsächlich x und y die Werte X und Y an z geliefert haben, müßten x und y dies der Senke bestätigen. Dazu müßten x und y ihre Meßwerte direkt an die Senke schicken und damit die Aggregation aufheben. Kurz: Daten können im Sensornetz *entweder* authentisch *oder* aber aggregiert zur Senke transportiert werden.

Die Überprüfung der Korrektheit der Aggregation und des Datenursprungs ist die zweite Kernaufgabe dieser Arbeit. Kapitel 4 widmet sich im Detail den speziellen Problem von Authentizität bei aggregierendem Datentransport in Sensornetzen und präsentiert das Protokoll ESAWN – einen Sicherheit-Energie Kompromiß zwischen Authentizität und Aggregation.

Geheimhaltung

Die Geheimhaltung der Daten über Aggregationsfunktionen hinweg macht in diesem Kontext *keinen* Sinn: Wie in den Annahmen über Aggregation beschrieben ist es durchaus denkbar, daß Aggregationsknoten nicht nur mit Hilfe beliebiger Funktionen Daten aggregieren, sondern auch basierend auf den empfangenen Daten als Aktoren im Netz dienen. So kann beispielsweise der Piepser am Körper des Patienten nicht nur Daten aggregieren und weitergeben, sondern gleich mit einem Warnton auf eine kritische Situation hinweisen.

Diese Arbeit nimmt an, daß ein Aggregationsknoten die Meßwerte oder Aggregate, die er empfängt, auch „lesen“ oder „verstehen“ können muß. Die Daten, die an einen Aggregationsknoten zur Aggregatbildung verschickt werden, *dürfen* demnach nicht vor ihm geheim gehalten werden.

2.6 Grundlagen zur Sicherheit

Diese Arbeit hat nicht den Anspruch, ein Lehrbuch über Sicherheit im allgemeinen oder über Sicherheit in Sensornetzen im speziellen zu sein. Dementsprechend werden die meisten Zusammenhänge über Sicherheit entweder als bereits bekannt vorausgesetzt oder aber, falls sie von *besonderer Bedeutung* für den sicheren Datentransport in Sensornetzen sein sollten, kurz und nur im Überblick vorgestellt.

2.6.1 Kryptographische Schlüssel

Im Verlauf der bisherigen Arbeit sind schon mehrfach *Schlüssel* erwähnt worden, mit denen Nachrichten oder beliebige Daten ganz allgemein *verschlüsselt* und *entschlüsselt* werden. Der folgende Abschnitt erklärt kurz, was überhaupt unter Schlüsseln sowie Ver- und Entschlüsseln zu verstehen und was dabei wichtig für Sensornetze ist.

2.6.1.1 Ver- und Entschlüsseln

Wenn zwei Knoten, Menschen, Protokollparteien usw. vertraulich miteinander Nachrichten austauschen wollen, chiffrieren[6] beziehungsweise verschlüsseln sie eine Nachricht, den sogenannten Klartext M, zu einem Chiffrat C, versenden C und de-chiffrieren beziehungsweise entschlüsseln C nach Empfang wieder zurück zu M. Zum Verschlüsseln von M kommt dabei eine Verschlüsselungsfunktion E zum Einsatz und zum Entschlüsseln von C eine Entschlüsselungsfunktion D. Es gilt $C = E(M)$ und

$$M = D(C) = D(E(M)).$$

Die Funktion D ist die (Links-)Inverse zu E. Falls ein Angreifer E und vor allem D nicht kennt, kann er C nicht entschlüsseln. Falls sich zwei Knoten im Sensornetz auf geheime E und D einigen können, könnten sie damit vertraulich kommunizieren.

[6] aus dem Französischen: Chiffre – das Geheimzeichen

In der Vergangenheit hat sich allerdings gezeigt, daß aus verschiedenen Gründen sowohl Chiffrierfunktionen E, D nicht lange geheimgehalten werden können und sollten und auch allgemein die Sicherheit von Ver- und Entschlüsselung nur von einem weiteren Parameter K, dem sogenannten Schlüssel, abhängen sollte. Dieses nennt man das „Kerckhoffsche Prinzip“, siehe dazu Kerckhoffs [128, 129]. Die Funktionen E und D können allgemein bekannt sein und werden für die Kommunikation zwischen verschiedenen Knoten mit verschiedenen Schlüsseln K_i parametrisiert zu $C = E_{K_i}(M)$, $M = D_{K_i}(C) = D_{K_i}(E_{K_i}(M))$. Der Schlüssel K_i selbst ist dabei nichts anderes als eine zufällige Zahl mit einer bestimmten Länge.

In dieser Arbeit werden der Einfachheit halber nur *perfekte* Funktionen E_{K_i}, D_{K_i} angenommen: $I(X)$ bezeichne eine Funktion, welche die Entropie einer Information oder eines Wertes X berechnet. Weiterhin bezeichnet $I(X|Y)$ die Entropie von X unter der Voraussetzung, daß Y bekannt ist. Die Funktionen E_{K_i}, D_{K_i} ver- beziehungsweise entschlüsseln *perfekt*, wenn gilt $I(M) = I(M|C)$ und $H(C) = H(C|M)$. Das bedeutet, die Entropie des Klartextes hängt nicht vom Chiffrat ab, und die Entropie des Chiffrats hängt nicht vom Klartext ab. Kennt ein Angreifer den Chiffretext, erhält er keinerlei Informationen über den Klartext, siehe dazu Rothe [192]. Kurz: Die „Sicherheit“ der Funktionen E_{K_i}, D_{K_i} hängt nur vom Schlüssel K_i ab. Bis auf den berühmten, in der Praxis jedoch kaum einsetzbaren *Vernam One-Time-Pad* [206], hat man noch für keine gängige Chiffrierfunktion die Eigenschaft *perfekt* beweisen können. Im Gegenteil: Häufig ist gezeigt worden, daß gängige Chiffrierfunktionen nicht perfekt sind. Dennoch nimmt diese Arbeit in einer Black-Box-Sichtweise der Einfachheit halber an, daß die zum Einsatz kommenden Chiffren perfekt sind, da die daraus resultierenden Probleme für die Aufgabenstellung dieser Arbeit unerheblich sind. Es geht nicht um absolut aus Informationstheorie sichere, unbrechbare, sondern um *praktisch* sichere Chiffren, die kein Angreifer mit „akzeptablem Aufwand“ brechen kann.

2.6.1.2 Symmetrische Verschlüsselung

Für *symmetrische* Verschlüsselungsfunktionen müssen sich zwei Kommunikationspartner, zwei Knoten im Sensornetz a, b, auf einen gemeinsamen Schlüssel $K_{a,b}$ einigen, mit dem sie wie beschrieben E und D parametrisieren. Wie oben erwähnt, hängt die Sicherheit symmetrischer Verschlüsselung nur von diesem Schlüssel ab, das bedeutet, a und b müssen ihren Schlüssel anderen gegenüber geheimhalten. Wenn ein anderer Knoten a' in Besitz von $K_{a,b}$ gelangt, kann er die Kommunikation zwischen a und b abhören, modifizieren, neue Chiffrate erzeugen usw.

Populäre symmetrische Chiffren sind zum Beispiel DES, 3DES, RC4, RC5, IDEA (Beschreibungen dazu unter anderem in Menezes et al. [156], Schneier [202]) oder AES (beschrieben in National Institute of Standards and Technology [163]). Die zum Einsatz kommenden Längen der Schlüssel $K_{a,b}$ variieren in den Verfahren zwischen 56 Bit und 256 Bit. Die sogenannte *Blockgröße* oder *Blocklänge* einer Chiffre bezeichnet dabei die Anzahl der Bits, die eine Chiffre in einem Ver- oder Entschlüsselungsvorgang gleichzeit bearbeitet. Typische Blocklängen gängiger Chiffren sind zum Beispiel 64 Bit oder 128 Bit. Falls ein Klartext M größer als eine solche Blocklänge sein sollte, muß er in mehrere Teile dieser Länge zerlegt werden. Falls ein Klartext kleiner als die Blocklänge verschlüsselt werden soll, muß er mit sogenannten *Padding*-Bits [156] auf die Blocklänge vergrößert werden.

2.6.1.3 Asymmetrische Verschlüsselung

Bei *asymmetrischen*, sogenannten *Public-Key*-Verschlüsselungsfunktionen E, D besitzt jeder Knoten a einen Schlüssel K_a bestehend aus zwei Teilen $K_a = (K_{a,e}, K_{a,d})$. Mit $K_{a,e}$ bezeichnet man den *öffentlichen* Teil des Schlüssels, mit $K_{a,d}$ den *geheimen* oder *privaten* Teil. Knoten a kann $K_{a,e}$ veröffentlichen. Das heißt, jeder andere Knoten b und auch korrumpierte Knoten dürfen $K_{a,e}$ kennen. Will nun Knoten b einen Klartext M für a verschlüsseln, so berechnet er $C = E_{K_{a,e}}(M)$ und verschickt C an a. Knoten a benutzt zum Entschlüsseln seinen geheimen Schlüssel $K_{a,d}$: $M = D_{K_{a,d}}(C) = D_{K_{a,d}}(E_{K_{a,e}}(M))$.

Typische Varianten von asymmetrischen Verschlüsselungsfunktionen sind die klassischen RSA, El-Gamal, siehe jeweils Schneier [202], einige mathematische Varianten davon, zum Beispiel El-Gamal basierend auf elliptischer Kurven Kryptographie (ECC) [155], NTRU [105] oder XTR [144]. Die hier zum Einsatz kommenden, üblichen Schlüssellängen variieren je nach Verfahren – so zum Beispiel mindestens 512 Bit bei RSA oder El-Gamal und ca. 100 Bit bei den moderneren Algorithmen.

2.6.1.4 Schlüssellängen – Vergleich symmetrisch und asymmetrisch

Ganz allgemein läßt sich festhalten, daß ein größerer Schlüssel auch eine höhere Sicherheit für die Verschlüsselung impliziert [192]. Auf der anderen Seite wiederum resultieren größere Schlüssellängen auch in mehr Speicher- und Energiebedarf bei den Verfahren, was sich gerade in Sensornetzen als äußerst kritisch darstellt.

Dementsprechend schwierig gestaltet sich die Suche nach einer „geeigneten“ Schlüssellänge. Geeignet bedeutet, daß der Schlüssel lang genug ist, um eine adäquate Sicherheit zu garantieren, so daß ein Angreifer den Schlüssel nicht einfach brechen oder durch Ausprobieren erraten kann. Gleichzeitig soll ein Schlüssel allerdings auch so kurz wie möglich sein, um nicht unnötig wertvolle Rechenzeit und Energie zu verbrauchen. Dabei lassen sich die Schlüssellängen für unterschiedliche Verfahren nicht einfach miteinander vergleichen. Zu unterschiedlich sind die möglichen Schwächen und damit Angriffspunkte auf einzelne Verfahren, deren Aufwand, der ständige Forschritt in der Forschung bzgl. der Unsicherheit der Verfahren, Entwicklungen im Hardware-Bereich wie das berühmte Moore'sche Gesetz [160] usw. Einige theoretische Hinweise zum Vergleich unterschiedlicher Schlüssellängen bei unterschiedlichen Verfahren geben Lenstra [142], Lenstra und Verheul [143], Orman und Hoffman [168]. So entsprach im Jahr 2005 die theoretische Sicherheit einer Schlüssellänge von 622 Bit beim RSA-Verfahren in etwa 56 Bit Schlüssellänge des DES und etwa 105 Bit bei ECC [34].

Welche Mindestlängen Schlüssel nach heutigen Gesichtspunkten haben sollten, ist demnach alles andere als einfach festzustellen. Auch die sogenannten *Crypto-Challenges* helfen dabei nicht viel: Die Firma RSA-Security veranstaltet einen Wettbewerb, bei dem Freiwillige Teilnehmer versuchen können, RSA-Schlüssel verschiedener Längen zu brechen [197]. Der bisher größte gebrochene RSA Schlüssel war 640 Bit lang und wurde innerhalb von 5 Monaten mit Hilfe von 30 jeweils 2.2 GHz Opteron Prozessoren gebrochen [196]. In Shamir und Tromer [205] behauptet Shamir sogar, daß man theoretisch mit spezieller, 10 Millionen US$ teurer Hardware innerhalb eines Jahres einen 1024 Bit RSA-Schlüssel brechen könnte. Genauso veranstaltet RSA-Security auch einen Wettbewerb zum Brechen von RC5 Schlüsseln [195]. Der größte bisher gebrochene symmetrische Schlüssel hatte eine Länge von 64 Bit und wurde vom Projekt *distributed.net* [78] innerhalb von 1757 Tagen (knapp 5 Jahre) gebrochen. Dem Projekt stand dabei permanent eine Rechenleistung zur Verfügung die in etwa äquivalent zu 160.000 Pentium-II

266 MHz Rechnern war. Im Rahmen eines Wettbewerbs wurde auch der bisher größte ECC-Schlüssel gebrochen: Innerhalb von 549 Tagen gelang es, einen 109 Bit Schlüssel mit Hilfe von nicht näher spezifizierten 10.000 Computern zu brechen [55].

Die obigen Zahlen sollen für die Problematik nur sensibilisieren – die Aussagekraft, ob eine bestimmte Schlüssellänge damit als unsicher gilt, ist dabei sicherlich diskutabel: Selbst wenn eine der in Schneier [202] häufig erwähnten sogenannten „Drei Buchstaben-Organisationen“ (CIA, FBI, NSA, KGB, BND, BKA, etc.) über eine höhere Rechenleistung verfügt, bleibt fraglich, ob sie den damit verbunden Aufwand in *akzeptabler* Zeit aufbringen kann und nicht 5 Jahre auf das Brechen eines Schlüssels warten muß. Festzuhalten bleibt, daß es sich heutzutage zumindest auf normalen PCs empfiehlt, zumindest keine wesentlich kürzeren Schlüssel als zum Beispiel 64 Bit RC5, 640 Bit RSA oder 109 Bit elliptische Kurven zu verwenden.

2.6.1.5 Fazit: Verschlüsselung in Sensornetzen

Auf Grund der beschränkten Hardware-Ressourcen gehören kryptographische Verschlüsselungsverfahren zu den „teuersten“ Algorithmen für Sensoren – sie benötigen viel Hauptspeicher und auf Grund ihrer Komplexität viel Rechenzeit und damit wertvolle Energie.

Zum Vergleich: Eine Addition zweier 8 Bit Zahlen geschieht auf der MICA2 Plattform in einem Takt und kostet damit ≈ 625 pAs Energie [11]. Das Versenden einer Nachricht auf MICA2/TinyOS kostet schon 245 μAs [30, 69], das entspricht ca. Faktor 400.000 zur Addition. Das Chiffrieren eines Klartextes mit zum Beispiel 64 Bit RC5 kostet $1,3$ μAs [11, 123] und ist damit zwar um Faktor 2080 teurer als eine Addition, aber noch deutlich günstiger, Faktor 190, als der Versand einer Nachricht. Eine einzelne Public-Key Chiffrieroperation mit elliptischen Kurven liegt allerdings je nach Implementierung und Schlüssellänge zwischen ungefähr 4 bis 28 mAs [29, 99] – Faktor ca. $7.000.000$ bis $45.000.000$ zur Addition. Der Unterschied im Energieverbrauch zwischen einer symmetrischen Verschlüsselung und einer asymmetrischen Verschlüsselung beträgt bis zu Faktor 22.000.

Dieses in Relation-Setzen unterschiedlicher Energieverbräuche soll folgendes aussagen: Public-Key-Kryptographie mag zwar in Sensornetzen auf MICA2-Basis durchaus einsetzbar sein, wie auch im Rahmen dieser Arbeit untersucht worden ist, siehe Blaß et al. [29], Blaß und Zitterbart [32, 33], Junker [119], jedoch kostet diese asymmetrische Kryptographie im Vergleich zur symmetrischen Kryptographie viel mehr Energie. Verschlüsselungsdauern im Sekundenbereich wie in Blaß et al. [29], Blaß und Zitterbart [32, 33], Gura et al. [99] bis hin sogar zum Minutenbereich [152] sind außerdem für einige Anwendungen nicht geeignet, wenn z.B. jede halbe Sekunde ein Blutdruckwert gemessen und weitergeleitet werden muß, vergleiche Diskussionen in Piotrowski et al. [178]. Außerdem bemerkenswert ist die Tatsache, daß eine symmetrische Verschlüsselung deutlich weniger Energie als der Versand einer Nachricht kostet.

2.6.1.6 Annahmen über Verschlüsselung in dieser Arbeit

Da asymmetrische Verschlüsselungsmechanismen so viel teurer als ihre symmetrischen Pendants und außerdem aufgrund der Ausführungszeiten im Sekundebereich in einigen Szenarien schlicht nicht einsetzbar sind, soll in dieser Arbeit gänzlich auf asymmetrische Kryptographie verzichtet werden und rein symmetrische Kryptographie zum Einsatz kommen. Dies macht die in den folgenden Kapiteln entworfenen Protokolle zudem auch

anwendbar für solche Sensornetzwerke, in denen noch Ressourcen-beschränktere Sensoren als die hier angenommenen MICA2 Knoten zum Einsatz kommen und bei denen sich unter Umständen asymmetrische Verschlüsselung erst gar nicht implementieren läßt.

Diese Arbeit nimmt an, daß zum Verschlüsseln von Daten eine RC5-Chiffre [187] verwendet wird. Zwar ist die konkrete Wahl einer symmetrischen Verschlüsselungsfunktion für den Entwurf von Protokollen relativ unwichtig, sie wird aber zu einem späteren Zeitpunkt für die Evaluierung der Protokolle bzgl. Speicher- und vor allem Energieverbrauch interessant. Die Wahl von RC5 als Chiffre in dieser Arbeit ist deshalb getroffen worden, weil für RC5 die bereits oben erwähnte, äußerst effiziente MICA2-Implementierung in Assembler existiert, für die in *TinySec* [123] Laufzeit und damit Energieverbrauch gemessen wurde. Zwar wären auch andere Algorithmen denkbar, beispielsweise sind im Rahmen dieser Arbeit einige der AES-Kandidaten wie MARS, RC6 und Rijndael für die MICA2-Plattform implementiert worden, sie benötigen aber ca. Faktor 1000 mehr an Energie als die Assembler-optimierte RC5 Implementierung aus TinySec [119].

Diese Arbeit nimmt wie oben beschrieben an, daß eine RC5 Ver- oder Entschlüsselung $1,3\ \mu$As Energie benötigt. Die Blocklänge sowie die Schlüssellänge betragen bei RC5 jeweils 64 Bit=8 Byte.

Public-Key Kryptographie in Sensornetzen und das Moore'sche Gesetz

Zum Abschluß des Abschnitts über den Einsatz von Verschlüsselung in Sensornetzen muß noch das berühmte „Moore'sche Gesetz" aus Moore [160] diskutiert werden. Dieses sogenannte Gesetz, es ist vielmehr eine Vorhersage, besagt, daß sich voraussichtlich die Anzahl der auf einem Chip integrierbaren Komponenten alle 18 bis 24 Monate verdoppeln. Daraus wird häufig abgeleitet, daß sich alle 18 bis 24 Monate die maximale Rechenleistung von Computersystemen verdoppelt.

Darauf aufbauend könnte man behaupten, die Rechenleistung aller in Sensornetzen zum Einsatz kommenden Mikrocontroller würde in wenigen Jahren derart steigen, so daß dadurch in Bälde auch extrem komplexe Protokolle mit viel Public-Key-Kryptographie auf Grund geringer werdender Laufzeiten möglich werden. Da asymmetrische Protokollösungen häufig wesentlich elegantere Lösungen verglichen mit rein symmetrischen zulassen, siehe zum Beispiel den folgenden Abschnitt über Authentizität, „lohnt" es sich nicht, Protokolle zu entwerfen, welche auf asymmetrische Primitive verzichten.

Diese 18 monatige Verdoppelung der Rechenleistung läßt sich allerdings nicht ohne weiteres auf Sensornetze übertragen: So argumentieren beispielsweise Mattern [153] und Karlof und Wagner [122], daß in Sensornetzen eher eine Art „umgedrehtes" Moore'sches Gesetz gilt, bei dem die Hersteller von Hardware nicht in Bezug auf maximale Rechenleistung optimieren, sondern eher versuchen, bei gleichbleibender Rechenleistung physisch kleinere Sensoren zu produzieren, die deutlich weniger Energie verbrauchen als die heutigen und dabei auch noch billiger werden. Man bedenke, daß heutige übliche Sensoren wie die MICA2-Plattform immernoch handtellergroß sind, AAA-Batterien benötigen und noch weit über 100 US$ [68] kosten. Demnach ist in Zukunft zu erwarten, daß die Sensoren eher kleiner, billiger und energiesparender werden als „schneller".

Schließlich bleibt noch daraufhinzuweisen, daß symmetrische Verschlüsselungsverfahren *immer* energiesparender als asymmetrische sein werden – in Sensornetzen *das* entscheidende Kriterium. Es „lohnt" sich also durchaus, Protokolle zu entwerfen, die auf asymmetrische Kryptographie verzichten.

2.6.2 Authentizität

Neben einigen grundlegenden Dingen zu kryptographischen Schlüsseln, die im Sensornetz ausgetauscht werden müssen, folgen nun auch Hintergründe über die *Authentizität* von Daten.

Man sagt, daß eine Nachricht M von einem Sender a an einen Empfänger b authentisch übertragen wird, wenn b nachprüfen kann, ob M während der Übertragung von a nach b nicht verändert oder (teilweise) ausgetauscht worden ist, also die von b empfangene Nachricht M tatsächlich so von a stammt.

Authentizität für übertragene Nachrichten kann man mit Hilfe verschiedener Techniken zusichern. Diese Techniken teilen sich auf in Techniken, die asymmetrische Kryptographie verwenden, und welche bei denen nur symmetrische zum Einsatz kommt. Die ersteren sollen aus obigen Gründen an dieser Stelle nur ganz am Rande betrachtet werden.

2.6.2.1 Authentizität mit asymmetrische Kryptographie – digitale Signaturen

Sehr häufig werden *digitale Signaturen* im Kontext von Authentizität genannt. Die Definition, was sich genau hinter einer digitalen Signatur verbirgt, unterscheidet sich leicht in verschiedenen Literaturstellen und der interessierte Leser sei zum Beispiel auf Anderson et al. [9], Diffie und Hellman [76], Fiat und Shamir [92], Pfitzmann [177] oder eine Zusammenfassung in Bless et al. [34] verwiesen.

An dieser Stelle reicht ein wesentlich oberflächlicheres Verständnis: Eine digitale Signatur einer Nachricht entspricht im groben einer handschriftlichen Signatur eines Dokumentes. Sie drückt eine Willenserklärung des Unterzeichners aus und dient nicht nur dem Nachweis der *Authentizität*, sondern auch der *Nicht-Abstreitbarkeit* [202] der Nachricht jedem gegenüber. Der Unterzeichner kann im nachhinein niemandem gegenüber abstreiten, daß er die Nachricht tatsächlich unterzeichnet hat. Mit Hilfe asymmetrischer Kryptographie lassen sich solche Signaturen erzeugen. Unterzeichner a signiert eine Nachricht M mit Hilfe seines privaten Schlüssels zur Signatur $S = D_{K_{a,d}}(M)$. Jeder, der $K_{a,e}$ und M kennt, zum Beispiel b, kann überprüfen, ob eine Signatur S von a stammt, in dem er $E_{K_{a,e}}(S) = E_{K_{a,e}}(D_{K_{a,d}}(M)) \stackrel{?}{=} M$ berechnet. Bei Übereinstimmung glaubt b dann, daß M auch tatsächlich von a signiert worden ist.

Mit Hilfe einer asymmetrischen digitalen Signatur kann nicht nur die Authentizität einer Nachricht vom Empfänger überprüft werden, sondern sie erfüllt auch *Nicht-Abstreitbarkeit*: b kann die Nachricht zusammen mit der Signatur von a jedem anderen Knoten wie zum Beispiel b' zeigen. Wenn b' den öffentlichen Schlüssel $K_{a,e}$ von a kennt, kann auch b' überprüfen, daß M tatsächlich von a stammt.

Problematisch bei digitalen Signaturen ist die dafür notwendige asymmetrischer Kryptographie. Daher eignen sich digitale Signaturen in Sensornetzen nur begrenzt.

2.6.2.2 Authentizität mit symmetrischer Kryptographie

Auch mit symmetrischer Verschlüsselung lassen sich Nachrichten grundsätzlich authentisch übertragen. In Burrows et al. [43] postulieren die Autoren die folgende Ableitung:

$$\frac{b \text{ believes } (a \overset{K_{a,b}}{\leftrightarrow} b), b \text{ sees } E_{K_{a,b}}(M)}{b \text{ believes}(a \text{ said } M)}$$

Falls b glaubt, einen gemeinsamen geheimen Schlüssel $K_{a,b}$ mit a zu besitzen und b ein Chiffrat $E_{K_{a,b}}(M)$ empfängt, dann glaubt b auch, daß a den dazugehörigen Klartext M versendet hat. Diese Ableitung ist völlig korrekt, allerdings zunächst nur in der Theorie.

In der Praxis treten jedoch die folgenden Probleme auf:

1. Wenn b ein Chiffrat $C = E_{K_{a,b}}(M)$ empfängt, dann ist dies zunächst einmal nichts anderes als eine beliebige Bitsequenz. Um diese zu entschlüsseln, muß b wissen, welchen Schlüssel er dafür zu benutzen hat. Steht für b nicht fest, von welchem Sender das Chiffrat kommt. Anders: Erwartet b möglicherweise von mehreren verschiedenen Sendern Nachrichten, so kann b dem Chiffrat C nicht „ansehen“, wer es verschickt hat und welchen Schlüssel b zum Entschlüsseln auswählen soll.

   Abhilfe läßt sich allerdings relativ einfach schaffen, indem a zusammen mit dem Chiffrat noch seine eigene Identität im Klartext an b mitschickt. Sender a schickt also $(a, E_{K_{a,b}}(M))$ an b. Analog kann b erkennen, wer die Nachricht abgeschickt hat, in dem er den *Kopf* der Nachricht untersucht, in dem, wie zum Beispiel bei TinyOS, typischerweise die Absenderadresse enthalten ist. Schließlich kann b auch aus dem Kontext eines Protokollablaufs klar sein, von wem er eine Nachricht empfangen muß.

   Sobald b weiß, von wem die Nachricht stammen soll, kann er den dazu passenden Schlüssel auswählen und C entschlüsseln.

2. Das reicht allerdings noch nicht aus. Selbst wenn a tatsächlich die Nachricht M authentisch verschicken möchte und dazu C an b sendet, kann ein Angreifer einzelne oder alle Bits von C verändern. Zwar gelingt es ihm damit nicht, gezielt einzelne Bits vom enthaltenen M zu modifizieren [235], es reicht aber, um M zu irgend einem anderen, zufälligen M' zu verändern. Entschlüsselt b dann C zu M', glaubt b, daß a auch M' gesendet hat. Analog kann der Angreifer auch den oben erwähnten Nachrichtenkopf mit der Absenderadresse fälschen, damit b den falschen Schlüssel $K'_{a,b}$ zum Entschlüsseln auswählt und auch so ein falschen M' erhält.

   Letztlich kann Empfänger b also doch nicht überprüfen, ob M' tatsächlich von a stammt. Hierbei handelt es sich um kein einfaches Problem, und der nächste Abschnitt beschäftigt sich mit seiner Lösung: *Authenticated Encryption*.

Authenticated Encryption

Mit Hilfe von symmetrischer Verschlüsselung wird zunächst nur Vertraulichkeit beim Versenden von Nachrichten garantiert – für die Integrität der übertragenen Nachrichten und damit die Authentizität bedarf es weiteren Aufwands. Die Kombination aus Vertraulichkeit und Authentizität mit Hilfe symmetrischer Chiffren nennt man in der Literatur Authenticated Encryption [18, 20, 169, 170].

Es existieren verschiedene Ansätze für Authenticated Encryption, üblicherweise kommen zwei verschiedene Methoden zum Einsatz:

1. **Spezielle Betriebsmodi symmetrischer Chiffren**
   Da die Größe des zu verschlüsselnden Klartextes M bei den meisten Anwendungen häufig wesentlich größer als die Blocklänge der verwendeten Chiffre ist, wird

M in mehrere Blöcke $M_1, M_2, \ldots, M_n$ aufgeteilt. Aus bestimmten Sicherheitsgründen verschlüsselt man die Blöcke nun nicht einfach jeweils einzeln, sondern verknüpft einen zu verschlüsselnden Klartextblöck M_i zum Beispiel mit vorhergehenden Klartextblöcken M_{i-1} oder vorhergehenden Chiffraten $E_{K_{a,b}}(M_{i-1})$. Die konkrete Vorschrift, nach der diese Verknüpfungen definiert man im sogenannten *Betriebsmodus* der Chiffre. Einige Betriebsmodi erzeugen nach der Verschlüsselung des letzten Klartextblocks M_n noch einen zusätzlichen, speziellen Authentizitätsblock, der es dem Empfänger der Chiffrate erlaubt, die Integrität und damit Authentizität aller bisher empfangenen Chiffrate zu verifizieren. Beispiele für solche Betriebsmodi, die auch Authentizität garantieren, sind CCM [243], XCBC [96], OCB [190], IAPM [120] usw.

2. **Message Authentication Codes**
   Bei einem *Message Authentication Code*, kurz *MAC*, handelt es sich um eine Funktion, die aus einem Klartext M einen Code $\mathrm{MAC}(M)$ berechnet. Wenn Sender a das Tupel $(M, \mathrm{MAC}(M))$ an einen Empfänger b verschickt, kann b damit überprüfen, ob M integer übertragen worden ist. Dazu berechnet er selbst den $\mathrm{MAC}(M)$ und überprüft ihn mit dem empfangenen MAC. Beispiele für MACs sind unter anderem Paritätsbits, Prüfsummen oder CRC-Polynome. Diese MACs sind allerdings unsicher in Gegenwart des spezifizierten Angreifermodells: Der Angreifer kann diese MACs genauso modifizieren, wie die Nachricht selbst. Von Bedeutung für die Authentizität in Gegenwart korrumpierter Knoten sind daher sogenannte *keyed Hashs* oder kurz HMACs, siehe Bellare et al. [17]. Hierbei handelt es sich um die Kombination aus einer Hash-Funktion und einem geheimen Schlüssel $K_{a,b}$ zwischen Sender a und Empfänger b, mit der a aus M derart einen $\mathrm{HMAC}_{K_{a,b}}(M)$ erzeugt, daß Empfänger b aus dem Tupel $(M, \mathrm{HMAC}_{K_{a,b}}(M))$ erkennt, daß M von a stammt. Ein typischer HMAC verknüpft dabei nur M zusammen mit $K_{a,b}$ und wendet eine Hash-Funktion darauf an [17].

   Um nun M vertraulich und authentisch an b zu schicken, berechnet und sendet a entweder $E_{K_{a,b}}(M, \mathrm{HMAC}_{K_{a,b}}(M))$ oder $(E_{K_{a,b}}(M), \mathrm{HMAC}_{K_{a,b}}(E_{K_{a,b}}(M)))$ oder $(E_{K_{a,b}}(M), \mathrm{HMAC}_{K_{a,b}}(M))$. Alle drei Varianten habe ihre Vor- und Nachteile, die hier allerdings keine Rolle spielen sollen, siehe dazu Bellare und Namprempre [18]. An dieser Stelle gilt es nur festzuhalten, daß a das Chiffrat und den HMAC an b sendet, der damit in der Lage ist, M's Authentizität zu verifizieren. Er erzeugt dazu genau wie oben bei einem einfachen MAC zunächst selbst einen eigenen HMAC und überprüft, ob dieser mit dem empfangenen HMAC übereinstimmt. Wenn beide HMACs übereinstimmten, dann weiß b, daß nur der andere Besitzer des Schlüssels $K_{a,b}$ die Nachricht so verschickt haben kann: a.

2.6.2.3 Authentizität und Authenticated Encryption in Sensornetzen

Beide oben vorgestellten Ansätze zur Authentizität eignen sich in erster Linie für Anwendungen, in denen Klartexte M wesentlich größer als die Blocklänge (zum Beispiel 64 Bit) einer Chiffre sind. In dem Fall, in dem ein Klartext viele hunderte oder tausende von Blöcken lang ist, fällt der zusätzliche Overhead durch die Erzeugung und Übertragung eines speziellen Authentizitätsblocks am Ende oder eines HMAC nicht weiter ins Gewicht.

In einem Sensornetz hingegen messen, verschlüsseln und versenden Sensoren Werte oder Phänomene, die häufig über eine deutlich geringere Entropie verfügen als die Blocklän-

ge. Die Herzfrequenz eines Menschen beträgt beispielsweise immer zwischen 0 und 200 Schlägen pro Minute. Damit liegt die Entropie $I(\mathrm{Herzfrequenz})$ bei unter 8 Bit. Selbst eine genauere Auflösung auf zehntel Herzschläge läßt sich in weniger als 11 Bit codieren. Viele Phänomene lassen sich sogar noch effizienter codieren, so wie „Tablette eingenommen“ in 1 Bit. Sobald jeweils ein einzelner Meßwert chiffriert von a den Empfänger b erreicht, muß b ihn sofort bzgl. Authentizität überprüfen können: Beispielsweise zeigt der Meßwert einen kritischen Gesundheitszustand an. Das bedeutet, b darf nicht erst nachdem a über einen langen Zeitraum hinweg eine ganze Reihe von Meßwerten verschickt hat, diese zusammen authentifizieren, sondern muß jeden Meßwert einzeln und sofort authentifizieren.

Erzeugt nun ein sendender Sensor a neben jedem Chiffrat für die Herzfrequenz auch noch jeweils einen HMAC zur Authentizität, so verdoppelt er damit die Kosten für Energie: Er muß zwei kryptographische Operationen ausführen und ca. die doppelte Datenmenge mit Hilfe von Nachrichten verschicken. Selbst bei einem hochauflösenden Phänomen mit 32 Bit Entropie „verschenkt“ der Sensor dabei immer noch die Hälfte seiner Blocklänge.

Daher schlägt zum Beispiel Yang et al. [250] vor, die nicht genutzten Bits der Blocklänge zur Authentizität zu nutzen. Die in 8 Bit codierte Herzfrequenz M wird mit 0er-Bits auf die Blocklänge l aufgefüllt (Padding) und verschlüsselt. Entschlüsselt der Empfänger b das daraus resultierende Chiffrat zu M' und liegt M' nicht zwischen 0 und 200, dann weiß b, daß M' nicht authentisch übertragen worden ist. Falls M' zwischen 0 und 200 liegt, dann ist der Klartext authentisch übertragen worden: Der Angreifer kann durch Verändern von Chiffrat-Bits nicht gezielt Bits im Klartext manipulieren. Eine ähnliche Idee kommt auch beim Internetprotokoll IPSec/ESP aus RFC 4304 [126] zum Einsatz: Der Klartext M wird mit einem bestimmten Muster auf Länge l aufgefüllt, so daß der Empfänger nach dem Entschlüsseln durch Überprüfen dieses Padding-Musters die Authentizität verifiziert.

Robustheit gegen Replay-Attacken

Eine andere Möglichkeit ist zum Beispiel, die ersten $(l - |M|)$ Bits des Klartextes als Padding für M zu verwenden, eine Sequenznummer wie die aus dem Nachrichtenkopf von TinyOS oder einen Zeitstempel als Padding zu nehmen usw. Eine Sequenznummer, ein einfacher Zähler oder Zeitstempel hätte sogar den Vorteil, daß man sich damit gegen Wiedereinspielungsangriffe, sogenannte *Replay*-Attacken [202], schützt, bei den der Angreifer versucht, alte Nachrichten nochmals zu versenden. Der Einsatz von Zeitstempeln hilft zudem in vielen Szenarien die Aktualität einer Nachricht zu überprüfen.

Mit welchen Informationen der Klartext auch immer aufgefüllt wird, der Benutzer kann daran erkennen, daß der Klartext authentisch übertragen worden ist.

Dadurch spart man sich die Übertragung zusätzlichen Datenvolumens, die der speziellen Authentizitätsblöcke oder der HMACs, sowie eine zweite kryptographische Operation, die Hash-Operation.

2.6.2.4 Annahmen über Authentizität und Authenticated Encryption in dieser Arbeit

Wie im Abschnitt über den Einsatz von Verschlüsselungsmechanismen bereits dargelegt, verzichtet diese Arbeit komplett auf Public-Key Kryptographie und damit auch auf (asymmetrische) digitale Signaturen. Stattdessen soll der Effizienz halber ausschließlich Gebrauch von symmetrischen Algorithmen, und zwar dem RC5 Algorithmus, für Zusicherung von Authentizität gemacht werden.

Diese Arbeit nimmt weiterhin wie Yang et al. [250] an, daß die zu übertragenden Klartexte, die Meßwerte der Sensoren, über eine deutlich geringere Entropie verfügen als die 64 Bit Blockgröße der RC5 Chiffre. Die Meßwerte in dieser Arbeit lassen sich effizient in ≤ 32 Bit codieren. Auf diese Weise können die mindestens 32 verbleibenden Bits wie oben beschrieben zur Authentizität jedes Chiffrats genutzt werden.

Was genau zum Padding des Klartextes zum Einsatz kommt, ob eine Sequenznummer, Teile des Klartextes, ein Zeitstempel oder Kombinationen davon, ist an dieser Stelle unerheblich. Empfänger b kann beim Empfang eines Chiffrates $C = E_{K_{a,b}}(M)$ verifizieren, ob das daraus entschlüsselte M so von a verschickt worden ist.

2.6.2.5 Non-Repudiation in dieser Arbeit

Zum Abschluß der Grundlagen über Authentizität muß noch eine weitere Sicherheitseigenschaft diskutiert werden, die eng verwandt mit der Authentizität ist: Die sogenannte Non-Repudiation [156], die sich am besten mit *Nicht-Abstreitbarkeit* oder *Verbindlichkeit* ins Deutsche übersetzen läßt.

Wenn a eine Nachricht, einen Klartext M, an b schickt, dann soll in vielen Situationen nicht nur b überprüfen können, ob a diese Nachricht geschickt hat (Authentizität), sondern später sollen auch andere, die an der Kommunikation zwischen a und b nicht beteiligt gewesen sind, diese Überprüfung vornehmen können. Vielfach möchte b einem Dritten, zum Beispiel einem Richter, beweisen, daß er M tatsächlich von a bekommen hat. Das bedeutet, a darf diesem Dritten gegenüber nicht abstreiten können, daß er M an b geschickt hat.

Digitale Signaturen leisten diese Form von Nicht-Abstreitbarkeit mit Hilfe asymmetrischer Kryptographie. Jeder, der den öffentlichen Schlüssel von a kennt, kann auch verifizieren, ob eine bestimmte Signatur von a stammt. Beim Einsatz rein symmetrischer Kryptographie gestaltet sich die Sache deutlich schwieriger. Die oben zitierte, theoretische Ableitung aus Burrows et al. [43] gilt in der Praxis nämlich auch nur dann, wenn b das Chiffrat $C = E_{K_{a,b}}(M)$ nicht selbst erzeugt hat. Da Empfänger b den Schlüssel $K_{a,b}$ besitzt, kann er damit offensichtlich auch selbst solche Chiffrate mit beliebigen Klartexten erzeugen. Falls b das Chiffrat C einem Richter vorlegen würde, könnte dieser nicht entscheiden, ob das Chiffrat von a oder von b erzeugt wurde.

Non-Repudiation mit Hilfe ausschließlich symmetrischer Techniken basiert immer auf der Verwendung von *Notaren*. Ein Notar N ist ein weiterer Protokollteilnehmer, dem alle anderen Protokollteilnehmer trauen – N ist niemals korrumpiert. Mit Hilfe eines Notars gibt es dann verschiedene Möglichkeiten, Non-Repudiation er erreichen. Eine Übersicht über symmetrische Non-Repudiation Techniken findet sich in ISO/IEC 13888-2:1998(E) [116].

Diese klassische Form von Non-Repudiation soll in dieser Arbeit nicht weiter betrachtet werden. Non-Repudiation würde keinen Mehrwert für die hier gewünschte Form von Authentizität bringen. In dieser Arbeit sollen Aggregationsknoten ihre Daten den Vorgängern auf ihrem Aggregationspfad authentifizieren. Es gibt keine beliebige weitere Dritte, denen gegenüber man Authentizität beweisen will. Es soll ausschließlich verhindert werden, daß korrumpierte Aggregationsknoten unbemerkt falsch aggregieren.

Andere Formen von Non-Repudiation

Die obige Form von Non-Repudiation nennt sich „Non-Repudiation of Origin“. Einem dritten gegenüber kann b beweisen, daß a wirklich M verschickt hat. Daneben existieren

allerdings noch eine Vielzahl andere Formen von Non-Repudiation: Non-Repudiation of Approval, Sending, Submission, Transport, Receipt, Knowledge oder Delivery, siehe McCullagh und Caelli [154]. Sie alle sollen nicht Thema dieser Arbeit sein. Es geht hier nur darum, daß korrumpierte Knoten nicht unbemerkt Aggregate anderen Aggregationsknoten oder der Senke gegenüber fälschen können.

2.6.3 Zerteilen von Geheimnissen

Simmons beschreibt in [209] eine Möglichkeit, Schlüssel, Geheimnisse oder sonstige wichtige Informationen in einzelne (Schlüssel-)Teile aufzuteilen. Dies ist insbesondere in Kapitel 3 beim Entwurf des neuen Schlüsselaustauschprotokolls von besonderer Bedeutung, soll aber der Übersichtlichkeit halber hier erklärt werden.

Ein Geheimnis oder ein Schlüssel K kann derart *perfekt* in zwei Teile (engl. *Shares*) K^1 und K^2 aufgeteilt werden, so daß der Besitz oder die Kenntnis von nur einem der beiden Teile keinerlei Informationen über K verrät. Eine Möglichkeit, K perfekt aufzuteilen, lautet wie folgt: $K = K^1 \oplus K^2$ und damit $K^1 = K \oplus K^2$. Hierbei steht $\oplus$ für das bitweise *exklusive oder*. Man wählt also eine zufällige Zahl K^2 der selben Länge wie der Schlüssel K und erhält nach XOR-Verknüpfung mit K schließlich K^1.

Dies erlaubt beispielsweise, zwei Schlüsselhälften K^1 und K^2 eines Schlüssels K sicher an zwei verschiedene Knoten zu verteilen. Solange nicht beide Knoten korrumpiert sind und in Zusammenarbeit ihre Schlüsselhälften austauschen, erlangt keiner der beiden irgendwelche Informationen über K.

2.7 Zusammenfassung

Sensornetze unterscheiden sich in vielen Kriterien von klassischen Netzen wie dem Internet, lokalen Netzen, modernen P2P-Netzen oder auch Ad Hoc-Netzen. Dieses Kapitel hat die besonderen Eigenschaften von Sensornetzen vorgestellt, die in dieser Arbeit angenommen werden: Ressourcenarmut, Fehlen von Infrastrukturen, Selbstorganisation, Aggregation und Dynamik. Zusammen mit einem in Sensornetzen realistischen Angreifermodell sind zwei Teilprobleme hergeleitet worden, welche die folgenden Kapitel genauer analysieren und bearbeiten – Schlüsselaustausch und authentischer Datentransport. Diese beiden Probleme bauen dabei aufeinander auf. Ein authentischer Datentransport benötigt als zwingende Voraussetzung Schlüssel zwischen den Sensorknoten, die später authentisch miteinander kommunizieren sollen.

3. Schlüsselaustausch

3.1 Motivation

Eine grundsätzliche Voraussetzung jedes sicheren Datentransports stellt die Existenz eines gemeinsamen Vertrauensankers zwischen den daran beteiligten Kommunikationspartnern dar. Üblicherweise kommen als gemeinsame Vertrauensanker kryptographische Schlüssel zum Einsatz. Erst wenn zwei Sensorknoten über einen gemeinsamen Schlüssel verfügen, können sie überhaupt sicher miteinander kommunizieren.

Dieses Kapitel widmet sich der Frage, wie solche Schlüssel in drahtlosen Sensornetzen *sicher* und für Sensornetze derart *geeignet* zwischen einzelnen Sensorknoten ausgetauscht werden, so daß wiederum damit Daten sicher von Blättern im Aggregationsbaum in Richtung Senke transportiert werden können. Die Bedeutung des Wortes *geeignet* für Sensornetze ist dabei noch zu untersuchen.

Ziel des Kapitels: Dieses Kapitel erklärt zunächst, welche Auswirkungen sich aus den besonderen Eigenschaften von Sensornetzen für einen Schlüsselaustausch ergeben. Daraus werden schlagwortartige Entwurfsziele formuliert, wie zum Beispiel *Dezentralität*, *Sicherheit* oder *Effizienz*, die ein Schlüsselaustauschprotokoll in Sensornetzen erfüllen sollte. Ein Überblick über den Stand der Forschung wird zeigen, daß bisher noch keine Arbeit all diesen Entwurfszielen genügt – insbesondere auch nicht die vielzitierten *Random Key Pre-Distribution*-Protokolle. Schließlich wird das im Rahmen dieser Arbeit entstandende Protokoll SKEY präsentiert und gezeigt, daß es alle Entwurfsziele erfüllt. Die Idee hinter SKEY ist, Schlüssel so in Teile zu zerlegen zu verschicken, daß es dem Angreifer nicht gelingt, in den Besitz aller Schlüsselteile zu gelangen.

Aufbau des Kapitels: Abschnitt 3.1.1 greift die im Grundlagenkapitel vorgestellten Eigenschaften von Sensornetzen auf, um deren Auswirkungen *speziell für den Schlüsselaustausch* zu erklären. Zusammen mit Abschnitt 3.1.2 werden damit Richtlinien, sogenannte Entwurfsziele, bestimmt, die ein geeignetes Protokoll erfüllen muß. Abschnitt 3.2 gibt einen Überblick über den Stand der Forschung, den Tabelle 3.1 auf Seite 62 zusammenfaßt. In Abschnitt 3.3 wird schließlich SKEY präsentiert. Eine Evaluierung in Abschnitt 3.4 schließt dieses Kapitel ab.

3.1.1 Neue Herausforderungen

Bevor zwei Sensorknoten a und b Daten sicher austauschen können, benötigen sie dafür zunächst einmal Schlüssel. Bei Verwendung von asymmetrischer Kryptographie muß Knoten a vorab den öffentlichen Schlüssel e_b von b kennen sowie b den öffentlichen Schlüssel e_a von a. Bei Einsatz rein symmetrischer Kryptographie müssen a und b einen gemeinsamen geheimen Schlüssel $K_{a,b}$ besitzen. Ein Protokoll zum Schlüsselaustausch versucht nun, einzelne Knoten im Netzwerk so mit Schlüsseln zu versorgen, daß Knoten aufgrund ihrer Mission oder ihres Einsatzzwecks mit jeweils notwendigen Partnern sicher kommunizieren können.

Im Gegensatz zu klassischen Netzen stehen Protokolle für den Schlüsselaustausch in Sensornetzen dabei allerdings vor speziellen Problemen. Diese werden im folgenden beschrieben. Motiviert durch diese Probleme wird anschließend die Aufstellung von *Entwurfszielen* für einen in Sensornetzen geeigneten Schlüsselaustausch motiviert.

3.1.1.1 Ressourcenarmut

Die in ihren Ressourcen beschränkte Sensor-Hardware und endliche Batteriekapazität zwingen auch Protokolle für sicheren Schlüsselaustausch zum extrem sparsamen und bewußten Umgang mit zur Verfügung stehenden Ressourcen wie Speicher, Rechenzeit oder Energieverbrauch. Obwohl sich herausgestellt hat, daß der Einsatz von Public-Key Kryptographie auf den beschränkten Knoten prinzipiell möglich ist, siehe zum Beispiel Gura et al. [99], Malan et al. [152] oder die im Rahmen dieser Arbeit entstandenen Blaß et al. [29], Blaß und Zitterbart [33], sollte sie dennoch nur wohlüberlegt und eher sparsam eingesetzt werden, da Ver- oder Entschlüsselungsoperationen ein Vielfaches von ihren symmetrischen Pendants „kosten“, siehe dazu auch Karloff et al. [123]. Grundsätzlich gilt, daß die Gesamtzahl sämtlicher kryptographischer Operationen, wie zum Beispiel Hash-, Chiffrier- oder Signierfunktionen, so gering wie möglich gehalten werden muß. Ähnliches gilt auch für die im Sensornetz zu übertragenen Daten zwischen einzelnen Knoten, die durch Schlüsselaustausch *zusätzlich* anfallen und über drahtlose Funkschnittstellen energetisch „teuer“ übertragen werden. Schließlich muß die Gesamtanzahl der zu speichernden notwendigen Informationen wie beispielsweise Vertrauensanker oder Schlüssel pro Knoten minimal sein. Im wertvollen Hauptspeicher sollte genügend Platz für die eigentliche Applikation und deren Nutzdaten zur Verfügung stehen. Auch die Länge der einzelnen Schlüssel ist dabei bewußt zu wählen: Ein hohes Maß an Sicherheit bedeutet automatisch hohen Speicherverbrauch.

3.1.1.2 Fehlende Infrastrukturen

Auf den Einsatz zentraler Infrastrukturkomponenten im Netz, zum Beispiel Server, soll im Sensornetz verzichtet werden. Aufgrund des besonderen Angreifermodells besteht die Gefahr, daß ein zentraler Server sehr einfach korrumpiert werden kann. Ein korrumpierter zentraler Schlüsselverteilungsserver würde die Sicherheit der gesamten, darauf aufbauenden Sensornetzkommunikation bedrohen. Ähnliches gilt für eine temporäre Nicht-Erreichbarkeit oder gar für den Ausfall einer solchen Zentrale, zum Beispiel mangels Energie. Die Nicht-Erreichbarkeit bestimmter Knoten oder gar ganzer Netzteile von anderen Knoten aus ist aber gerade in Sensornetzen wahrscheinlich.

Ein für Sensornetze geeignetes Protokoll zum Schlüsselaustausch darf daher auf keinen Fall häufigen Gebrauch von Infrastrukturkomponenten machen bzw. sich auf solche Komponenten nicht verlassen.

3.1.1.3 Selbstorganisation und Spontaneität

Diese Punkte sind eng mit dem der fehlenden Infrastruktur verwandt: Jede Benutzerinteraktion mit dem Sensornetz soll sich auf ein Minimum, zum Beispiel die „normale“ Abfrage nach Informationen oder die Inanspruchnahme von Diensten, beschränken. Im Gegensatz zu klassischen Netzwerken ist der (physische) Zugang zu einzelnen Sensorknoten häufig nicht gewünscht oder auch gar nicht möglich. Weiterhin kennt der Benutzer Details der Netzwerkkonfiguration nicht, so daß der Vorgang des Schlüsselaustausches nicht ohne weiteres vom Benutzer kontrolliert und gesteuert werden kann. Das Netz soll selbstorganisieren, welcher Knoten auf welche Weise Schlüssel austauscht.

Im Kontext des sicheren Schlüsselaustausches sind die Begriffe *Spontaneität* bzw. *Ad-Hoc*-Verhalten von großer Bedeutung. Die eigentliche Bildung eines Sensornetzes, wie zum Beispiel das Hinzufügen eines neuen Knotens, geschieht spontan also Ad-Hoc und nicht immer durch den oder die Benutzer a priori bis ins Detail (wann?, welcher Knoten?, wie? usw.) geplant. Bei „Bedarf“ wird dem Netz einfach ein neuer Knoten hinzugefügt. Andererseits *müssen* jedoch Knoten, die sicher miteinander kommunizieren sollen, vorab schon über gemeinsame Geheimnisse, Zertifikate etc. verfügen. Man nehme an, daß es möglich wäre, neue Sensorknoten, beispielsweise unmittelbar nach dem Kauf in einem Supermarkt, ohne weitere Benutzerinteraktion oder Vorbereitung dem Sensornetz „sicher“ hinzuzufügen. Wenn dies möglich wäre, so könnte allerdings auch ein Angreifer einen kompromittierten Knoten „sicher“ dem Netz hinzufügen. Kommunikation zwischen zwei oder mehreren Knoten kann daher ohne einmalige Benutzerinteraktion niemals sicher sein, genauer: Grundsätzlich benötigen zwei Knoten, auch zum sicheren Austausch von Schlüsseln, vorab eine Art gemeinsames Vertrauen, einen gemeinsamen Vertrauensanker [34], zum Beispiel ein gemeinsames Geheimnis oder ein Zertifikat, siehe Diskussionen in Alkassar et al. [6], Barak et al. [14], Beth und Desmedt [24], Weimerskirch und Westhoff [237]

Diese Tatsache mag auf der einen Seite als relativ einfach erscheinen, führt jedoch auf der anderen Seite zu gravierenden Konsequenzen: Die Idee der Spontaneität steht demnach zumindest zum heutigen Begriff der Sicherheit im Widerspruch. Daher ist zumindest initial durch den Benutzer ein einmaliges Vorbereiten oder *Paaren* eines neuen Knotens für den Einsatz im Netz zwingend notwendig. Im Rahmen des Paarens ist dann der Benutzer in der Lage, dem neuen Knoten einen notwendigen Vertrauensanker oder bestimmte Geheimnisse für sicheres Arbeiten im Netz zu übergeben [13, 26, 28, 102–104, 229–231]. Das Paaren eines neuen Knoten kann der Benutzer beispielsweise mit Hilfe eines *Trusted Dealer*, siehe Cramer und Damgård [65], van Tilborg [219], durchführen. Auch in dieser Arbeit kommt ein Trusted Dealer zum Einsatz, das sogenannte Master Device (MD) – dazu allerdings erst in Abschnitt 3.3.3.1, S. 67 mehr.

Im Gegensatz zu klassischen Netzen, bei denen Benutzerinteraktionen häufig während des gesamten Lebenszeitraums eines Netzwerks möglich sind, beschränken sich Benutzerinteraktionen in Sensornetzen nur auf das notwendige initiale Paaren. Danach, das heißt *nach* dem Ausbringen oder der Installation der Knoten, sind diese daher auf sich allein gestellt und müssen einen Schlüsselaustausch selbst und autonom organisieren.

3.1.1.4 Kommunikationsfluß: Aggregation

Wie bereits in Abschnitt 2.2.4, S. 16 erwähnt, kommunizieren Sensoren üblicherweise ausschließlich mit ihren zugehörigen Aggregationsknoten, mit denen sie in Aggregati-

onsbeziehung stehen. Rein auf Applikationsebene, der Aggregation, besteht kein Bedarf für beliebige Kommunikation.

Folglich kann man an dieser Stelle festhalten, daß dann auch nicht Schlüssel zwischen *beliebigen* Knoten ausgetauscht und gespeichert werden müssen, sondern nur zwischen Knoten, die Vorgänger und Nachfolger voneinander im Aggregationsbaum sind. Jeder Knoten benötigt Schlüssel mit den Knoten, die auf seinem *Aggregationspfad* liegen [25, 26]. Diese Beobachtung ist neu. Bisherige Forschungsarbeiten setzen immer die beliebige Kommunikation zwischen allen Knoten und damit implizit den paarweisen Bedarf von Schlüsseln zwischen allen Knoten voraus.

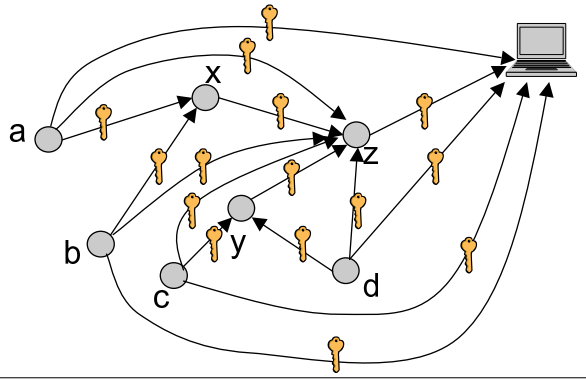


Abbildung 3.1 Schlüsselverteilung im Aggregationsbaum

Zur Verdeutlichung dient Abbildung 3.1, die den Bedarf an Schlüsseln im Aggregationsbaum zeigt. Im Beispiel der Temperatursensoren a und b in Raum 1 beziehungsweise c und d in Raum 2 wird klar, daß keinerlei Notwendigkeit besteht, zwischen den Knoten a und b oder gar a und c einen Schlüssel auszutauschen. Im Gegenteil, kommuniziert wird nur zwischen Blättern und Aggregationsknoten oder Aggregationsknoten untereinander: zum Beispiel a und x oder x und z, diese Knoten brauchen paarweise Schlüssel. Es wird nun aufgrund von Aggregationsbeziehungen miteinander kommuniziert. Um aber darüberhinaus Datenauthentizität oder Daten-*Echtheit* auch trotz Aggregation zu garantieren, zu dieser Problematik siehe Kapitel 4, sind jedoch auch paarweise Schlüssel zwischen beispielsweise a und z oder c und z erforderlich. Jede Kante im Aggregationsbaum aus Abbildung 3.1 steht für einen notwendigen Schlüssel zwischen zwei Knoten. Zusammenfassend: Jeder Knoten braucht nur mit den Knoten auf dessen Aggregationspfad $\mathbb{P}$ jeweils paarweise verschiedene Schlüssel – und nicht mit jedem beliebigen Knoten im Netz.

Die baumartige Aggregation muß zwangsläufig in einem deutlichen niedrigeren Speicherverbrauch pro Knoten für kryptographische Schlüssel resultieren: Anstatt linearem Speicherverbrauch für paarweise Schlüssel zwischen allen Knoten ist bei der Anordnung der Knoten im Aggregationsbaum nur mit logarithmischem Speicherverbrauch zu rechnen, denn der Speicherverbrauch für Schlüssel steigt mit der Höhe des Baumes. Damit einhergehend ist insgesamt weniger Aufwand für den gesamten Schlüsselaustausch im Netz zu erwarten, weniger notwendige Datenübertragungen, weniger Energieverbrauch, weil es insgesamt weniger Schlüssel auszutauschen gibt. Zur Steigerung der Effizienz im Umgang mit den zur Verfügung stehenden, begrenzten Ressourcen muß ein Protokoll zum Schlüsselaustausch diesen besonderen Kommunikationsfluß mit in Betracht ziehen.

3.1.1.5 Dynamisches Netzverhalten

Das dynamische Netzverhalten hat besondere Auswirkungen auf den Schlüsselaustausch. Durch die Dynamik, zum Beispiel zusätzliche, neue Knoten, das Netz verlassende Kno-

ten und die Umverteilung von Aufgaben innerhalb des Netzes, kann sich der Aggregationsbaum und damit der Bedarf an gemeinsamen Schlüsseln vom Netz selbstorganisiert ändern. Der Bedarf unterschiedlicher Knoten nach gemeinsamen Schlüsseln mit anderen Knoten wird auf diese Weise *dynamisch*.

Es kann kein einmaliges Austauschen von Schlüsseln zum Beispiel in einer einmaligen *Setup*-Phase geben, sondern einen kontinuierlichen Schlüsselaustausch je nach Knotendynamik. Ein Protokoll zum Schlüsselaustausch muß mit dieser Dynamik umgehen können, das heißt selbst bei einer hohen Dynamik durch eine Vielzahl spontan und beliebig hinzukommender und vor allem ausfallender Knoten noch effizient und sicher funktionieren. Es muß auch mit zunehmender Anzahl von Knoten gut skalieren.

3.1.2 Entwurfsziele und erwartete Ergebnisse

Die Entwurfsziele, die ein Protokoll zum Schlüsselaustausch in Sensornetzen erfüllen muß, lassen sich wie folgt zusammenfassen. Vorab die beiden „trivialen“ Entwurfsziele, die Schlüsselaustausch klassischerweise leisten muß:

- Entwurfsziel *Funktionalität*
  Das Protokoll muß es denjenigen Knoten, die miteinander kommunizieren wollen, ermöglichen, gegenseitig Schlüssel auszutauschen. Bei Aggregation sind das jeweils die Knoten x_i und alle Knoten auf deren Aggregationspfad $\mathbb{P}_{x_i}$.

- Entwurfsziel *Sicherheit*
  Der Austausch von Schlüssel an sich hat dabei wiederum sicher zu sein, also selbst in der Gegenwart von Angreifern, mehreren korrumpierten Knoten, muß das Protokoll noch nach den in Abschnitt 2.5.1 spezifizierten Sicherheitsanforderungen sicher Schlüssel verteilen: Der Angreifer darf ausgetauschte Schlüssel nicht abhören oder den Schlüsselaustausch gezielt beeinflussen.

Schließlich folgen Entwurfsziele, die sich aus der speziellen Problematik drahtloser Sensornetzen ergeben:

- Entwurfsziel *Dezentralität*
  Ein Verfahren zum Austausch von Schlüsseln in Sensornetzen sollte nicht auf eine jederzeit online-erreichbare Zentrale oder Infrastrukturkomponente *angewiesen* sein. Ein Schlüsselaustausch zwischen beliebigen Knoten sollte im Normalfalle unabhängig von einer Zentrale funktionieren. Zu wahrscheinlich ist, daß diese Zentrale zeitweise ausfällt oder nicht erreichbar ist.

- Entwurfsziel *Spontaneität und Selbstorganisation*
  Eine Interaktion zwischen dem Benutzer und einzelnen Sensoren ist im Allgemeinen nur *vor* ihrem Ausbringen, dem Deployment, möglich. Danach muß sich das Netz selbst organisieren.

- Entwurfsziel *Dynamik*
  Der Schlüsselaustausch muß auch bei einer hohen Anzahl von dem Netz zwischenzeitlich beitretenden oder das Netz verlassende Knoten noch sicher möglich sein. Sich im Laufe der Zeit ändernde Aufgabenstellungen und damit dynamischer Bedarf an Schlüsseln zwischen Knoten müssen unterstützt werden. Es liegt zu keinem Zeitpunkt Wissen über die gesamte Netzkonfiguration vor.

- Entwurfsziel *Effizienz*
  Neben diesen rein funktionalen Anforderungen stellt die Effizienz aller Protokolle in Sensornetzen eine weitere Voraussetzung dar: Der durch den Schlüsselaustausch entstehende Overhead an zusätzlicher Energie, zusätzlichen Nachrichten und erhöhtem Speicherverbrauch sollte möglichst gering ausfallen. Da in dieser Arbeit Aggregation als vorherrschender Kommunikationsfluß zum effizienten Datentransport angenommen wird, sollte diese in ein Protokoll mit einbezogen werden.

Das erwartete *Ergebnis* am Ende eines Schlüsselaustausches ist, daß jeweils zwei kommunikationswillige Knoten über gemeinsame Schlüssel verfügen. Denn, sobald Knoten gemeinsam mit Schlüsseln ausgestattet sind, existiert eine *Basissicherheit* zwischen ihnen. Basissicherheit bedeutet, daß, falls zwei Knoten a und b einen gemeinsamen Schlüssel besitzen, diese mit dessen Hilfe schon sicher Daten austauschen können. Besitzen die beiden beispielsweise einen gemeinsamen geheimen Schlüssel $K_{a,b}$, so können beide Knoten durch Verschlüsselung *vertraulich* und *integer* kommunizieren. Analog kann mit Hilfe von $K_{a,b}$ auch *authentisch* zwischen a und b kommuniziert werden [43]. Alleine der Besitz eines geheimen Schlüssels zwischen a und b garantiert Basissicherheit, zumindest zwischen a und b, genauer: Geheimhaltung, Integrität und Authentizität. Sicherheit über mehrere kaskadierende Aggregationen hinweg kann allerdings erst mit den Techniken aus Kapitel 4 erzielt werden.

3.2 Stand der Forschung

Da ein funktionierender Schlüsselaustausch die fundamentale Voraussetzung für jedwedes Sicherheitsprotokoll darstellt, hat sich die Forschung diesem Problem schon relativ intensiv gewidmet. Allerdings sind die bisherigen Lösungsvorschläge im Kontext von Sensornetzen alles andere als zufriedenstellend – wie im folgenden gezeigt wird.

3.2.1 Public-Key Varianten

Der aus klassischen Netzen bekannte und inzwischen sehr traditionelle Ansatz, Schlüssel zu verteilen beziehungsweise auszutauschen, ist der Einsatz einer sogenannten Public-Key-Infrastruktur [34]. Prinzipbedingt wären Public-Key Lösungen ideal, da effizient für Sensornetze: Gespeichert werden muß pro Knoten immer nur die Menge an öffentlichen Schlüsseln, die zur Kommunikation mit bestimmten anderen Knoten gerade notwendig ist.

Die Etablierung einer Public-Key-Infrastruktur (PKI) [219] funktioniert in Sensornetzen jedoch aus mehreren Gründen nicht. Der Einsatz einer dafür zwingend notwendigen zentralen Authentifizierungsstelle, der sogenannten CA – *Certification Authority* [202], widerspricht dem Entwurfsprinzip *Dezentralität*. Zudem verwalten CAs Widerrufslisten, die es notwendig machen, daß Sensoren periodisch von Zeit zu Zeit in Verbindung mit ihr treten müssen, um widerrufene Zertifikate zu überprüfen.

Ein ebenso großes Problem liegt aber in dem mit PKIs verbundenen Overhead. Zum Einsatz kommt hier asymmetrische Kryptographie, wie beispielsweise elliptische Kurven Kryptographie (ECC) [131]. Bei elliptischen Kurven handelt es sich um Punktemengen über einem endlichen Körper, für die eine Addition definiert ist, welche eine mathematische Gruppe bildet. Innerhalb dieser Gruppe gibt es ein Problem ähnlich zum Problem

des diskreten Logarithmus (DLP) in endlichen Körpern, das sogenannte ECDLP (elliptische Kurven diskreter Logarithmus Problem): Aus dem Produkt einer Skalarmultiplikation läßt sich das Skalar nicht berechnen. ECC hat gegenüber älteren Public-Key-Verfahren wie RSA [188] auf Grund kleiner Operanden einen Speicher- und Rechenzeitvorteil.

Verglichen mit rein symmetrischen Methoden befindet sich asymmetrische Kryptographie jedoch im Nachteil: Asymmetrische Schlüssel, die in Zertifikaten gespeichert werden müssen, sind größer als symmetrische. Obwohl für eine Kommunikation zwischen zwei Knoten eigentlich nur zum Beispiel 64 Bit Schlüssel erforderlich sind, benötigen Zertifikate mehr Speicher. Bei gleicher Sicherheit, die ein 64 Bit symmetrischer Schlüssel bietet, benötigt ECC 120 Bit Speicher [143] sowie die noch dringend notwendige Unterschrift der CA mit nochmals 120 Bit Speicher. Üblicherweise kommen bei Zertifikaten wie zum Beispiel X.509 [115] noch eine ganze Reihe zusätzlicher Informationen hinzu, die den Speicherbedarf pro Kommunikationsbeziehung noch weiter erhöhen. Das X.509 Zertifikat des Rechners www.tm.uka.de hat Beispielsweise eine Größe von 4397 Byte. Die Überprüfung jedes öffentlichen Schlüssels oder die Ver- und Entschlüsselung ist weiterhin mit komplexen Berechnungen verbunden, die um *Größenordnungen* mehr Energie als ihr symmetrisches Pendant kosten, siehe zum Beispiel Karloff et al. [123].

Public-Key-Verfahren erfüllen daher nicht das Entwurfsziel *Effizienz*. Beispiele für Arbeiten, die auf solcher ECC-Kryptographie in Sensornetzen basieren, sind Du et al. [84], Huand et al. [108], Huang et al. [110], Kotzanikolaou et al. [132], Kumar et al. [134], Magkos et al. [150], Malan et al. [152] und die im Rahmen dieser Arbeit entstandene Arbeit Blaß und Zitterbart [33].

Aus all diesen Gründen sind auch verteilte Authentifizierungsstellen oder ein *Web-of-Trust* [164] in Sensornetzen unrealistisch.

Eine besonders elegante Variante, Schlüssel in einem Sensornetz zu verteilen, wird unter anderem in Carman et al. [48], Lauter [137], Zhang et al. [253, 254, 255] beschrieben und geht zurück auf eine Idee aus Shamir [204]: Bei der sogenannten *identitätsbasierten Verschlüsselung* entspricht der öffentliche Schlüssel eines Kommunikationspartners beispielsweise seinem Klartextnamen oder anderen offensichtlichen Eigenschaften, anhand deren sein Gegenüber eindeutig erkennen kann, dass es sich bei einem bestimmten Public-Key tatsächlich um seinen Kommunikationspartner handelt. Der Versand und die Überprüfung von Zertifikaten ist damit nicht mehr notwendig. Sensoren könnten sich so gegenseitig, zum Beispiel alleine aus ihrer Dienstbeschreibung wie “Temperatursensor in Raum IV” oder “Lichtsensor Decke”, Schlüssel ableiten. Die Dienstbeschreibung entspräche dann dem öffentlichen Schlüssel.

Eine Umsetzung beziehungsweise Implementierung dieser Idee für Sensornetze steht allerdings noch aus, da zunächst zu klären ist, in wie weit die Rechenleistung typischer Sensor-Hardware für dieses Verfahren ausreicht. In den vorgestellten Arbeiten kommen rechenaufwendige Mechanismen wie beispielsweise elliptische Kurven und komplexe „bilineare Paarungen“ [39] zum Einsatz. Bilineare Paarungen sind Abbildungen $e : G \times G \rightarrow G$ über einer endlichen Gruppe G. Die Bilinearität von e bedeutet $e(aP, bQ) = e(P, Q)^{ab}, \forall a, b \in \mathbb{Z}, \forall P, Q \in G$. Die Berechnung solcher Paarungen ist noch rechen- und speicheraufwendiger als ECC – bisher existiert noch keine effiziente Implementierung davon für Sensor-Hardware. Identitätsbasierte Verfahren sind demnach sehr *ineffizient*.

Ähnlich dazu setzen einige Arbeiten, wie Law et al. [139], Liu und Ning [146], Yang et al. [249] auf Ideen ursprünglich aus Blundo et al. [36] über *bivariate Polynome* auf: Der Benutzer generiert ein bivariates Polynom

$$f(x,y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} x^i y^j$$

vom Grad t über dem endlichen Körper $\mathbb{F}_q$ mit der Eigenschaft $f(x,y) = f(y,x)$. Der Benutzer kann t dabei wählen. Die Koeffizienten $a_{i,j}$ hält der Benutzer geheim. Für jeden Knoten mit einer ID a berechnet der Benutzer die Koeffizienten a'_j des sogenannten *Share*-Polynoms

$$f(a,y) = \sum_{j=1}^{t} a'_j y^i.$$

Dieses Share erhält der Knoten a vor dessen Deployment und stellt seinen geheimen Schlüssel dar. Die ID eines Knotens, a, ist sein öffentlicher Schlüssel. Wollen zwei Knoten a, b miteinander einen sicheren Schlüssel austauschen, berechnet a: $f(a,b)$ und b: $f(b,a)$. Da $f(a,b) = f(b,a)$ kennen die beiden Knoten jetzt einen gemeinsamen symmetrischen Schlüssel $f(a,b) \in \mathbb{F}_q$ mit der Bit-Breite $\log q$. Jeder Knoten benötigt dabei pro Schlüssel $(t+1)\log q$ Speicher. Der Speicherbedarf dieser Ansätze steigt allerdings schnell: t muß vom Benutzer sehr hoch gewählt werden, da ansonsten t korrumpierte Knoten das geheime Polynom $f(x,y)$ rekonstruieren können. Nimmt man beispielsweise an, daß $\log q = 64$ Bit breite symmetrische Schlüssel zum Einsatz kommen, sowie 100 korrumpierte Knoten im Sensornetz zu erwarten sind, dann verbraucht ein *einzelner* Schlüssel bereits $101 \cdot 8 = 808$ Byte Hauptspeicher. Auch diese Idee des Schlüsselaustauschs ist sehr *ineffizient*.

3.2.2 Einsatz dedizierter Knoten

Einige Protokolle nehmen im Netz die Existenz besonderer Knoten, zum Beispiel besonders leistungsfähiger Knoten, an. Solche Super-Knoten können über wesentlich mehr Rechenleistung, Speicher oder Batterie verfügen als die restlichen Sensoren. Wenn sich ein solcher Super-Knoten in direkter Funkreichweite zweier Knoten x, y befindet, die miteinander kommunizieren wollen, so kann er auf Grund seiner besonderen Fähigkeiten beiden Knoten aktiv helfen, Schlüssel miteinander auszutauschen. Ein Vorteil dieser Idee ist, daß der Energieaufwand für den Schlüsselaustausch zwischen x und y durch die Hilfe des Super-Knotens in Grenzen gehalten werden kann.

Die Autoren Carman et al. [47] stellen ein zu Kerberos [158] ähnliches Verfahren vor. Es gibt mehrere dedizierte Sensoren, so genannte Schlüsselverteilungssensoren, die in der Lage sind, wie bei Kerberos für zwei angehende Kommunikationspartner auf Wunsch Sitzungsschlüssel zu verteilen. Ein Problem ist hierbei der Ausfall oder die zeitweise Nicht-Erreichbarkeit eines oder mehrerer Schlüsselverteilungssensoren, was ja aufgrund der besonderen Eigenschaften von Sensornetzen durchaus realistisch ist. Damit erfüllt Carman et al. [47] nicht das Entwurfsziel *dezentral*. Zudem muß der Benutzer vor Ausbringen der Sensoren eine bestimmte jeweils zu einem Verteilungssensor gehörende Gruppe von Sensoren speziell auf den Verteilungssensor abstimmen, eine Diskrepanz zum Entwurfsziel *dynamisch*.

Ganz ähnlich schlagen die Autoren Law et al. [138] die Existenz sogenannter *Supervisor*-Knoten vor, die *Tamper-Proof* sein sollen, das heißt einen wirksamen Schutz gegen jede

Art von physikalischer Attacke besitzen. Supervisor-Knoten agieren in diesem Vorschlag als Notar für alle anderen Sensoren in deren Sendereichweite und vermitteln paarweise Schlüssel zwischen zwei Kommunikationspartnern. In wie weit die Idee der Supervisor-Nodes in Sensornetzen realistisch ist, bleibt fraglich: Auch sie widerspricht den Entwurfszielen *Dezentralität* und *Dynamik*. Bei einem möglichen Ausfall dieser Knoten sind die Folgen für die damit verbundenen *supervised* Knoten, diejenigen Knoten, die von den Supervisor abhängen, extrem: Schlüsselaustausch ist ab dann nicht mehr möglich. Zudem müßten vorab vom Benutzer Supervisor-Knoten an strategisch möglichst günstigen Orten plaziert sein, ein Auswahlverfahren ist für deren Bestimmung erforderlich, und die Schlüssel müssen explizit zwischen Supervisor- und Supervised-Knoten verteilt werden.

Eine dazu analoge Idee stammt von Stajano [215]. Ein neuer Sensor verhält sich analog wie das *Küken* einer Ente: Er vertraut dem ersten Gegenüber (PC, PDA oder ein anderer Sensor), das er sieht wie einer *Mutter*. Ein geheimer Schlüssel, den er von diesem ersten Objekt bekommt, wird so automatisch als vertrauenswürdig und authentisch verstanden. Eine weitere Vertrauensbildung ist nicht vorgesehen. Möchte ein anderer Sensor mit dem Küken kommunizieren, muß er sich zunächst bei der Mutter authentifizieren. Diese verhandelt dann einen Sitzungsschlüssel zwischen ihrem Küken und dem neuen Sensor aus. Obwohl diese Idee aus der Biologie sehr interessant scheint, löst sie dennoch einige elementare Probleme nicht: Wie kann ein Schlüssel *dezentral* ohne die Interaktion der Mutter zwischen zwei Sensoren ausgetauscht werden? Es ist sehr wahrscheinlich, daß die Mutter dem Küken nicht permanent zur Verfügung steht. Außerdem läßt Stajano [215] offen, wie sich ein Sensor der Mutter gegenüber authentifiziert. Das initiale Bootstrapping in einem solchen System ist unklar.

Andere Protokolle verteilen vor dem Deployment der Knoten geheime symmetrische Schlüssel auf den Sensoren, so wie zum Beispiel Deng et al. [71], Khalil et al. [130] oder die μTESLA beziehungsweise SPINS genannten Protokolle aus den Arbeiten Engelbrecht und Penzhorn [87], Perrig et al. [172, 173, 174]. Sie haben die Verwendung einer Basisstation gemein, welche die Verschlüsselung zwischen zwei Knoten aktiv unterstützt. Auf diese Weise werden paarweise zwischen den Kommunikationspartnern keinerlei Schlüssel benötigt. Nachteil dieser Vorgehensweise ist die Verwendung Basisstation. Sie soll absolut vertrauenswürdig, nicht kompromittierbar von jedem anderen Sensor im Netz jederzeit erreichbar sein. Ein Konflikt zu den Entwurfszielen *Dezentralität, Dynamik* und *Spontaneität*.

Insgesamt kann man festhalten, daß der Einsatz dedizierter Knoten im allgemeinen wenig Protokollaufwand für je zwei angehende Kommunikationspartner bedeutet, jedoch aufgrund der dafür notwendigen Zentralen, mag man sie *Super*-Knoten, *Supervisor* oder *Mutter* nennen, immer zumindest dem Entwurfsziel *Dezentralität* widerspricht.

3.2.3 Annahmen über sicheres Deployment

Einen gänzlich anderen Ansatz verfolgt die Arbeit *Key Infection* aus Anderson et al. [10]. Die Autoren argumentieren, daß nicht wie bisher das initiale Verteilen von Schlüsseln oder gemeinsamen Geheimnissen in Sensornetzen das Problem ist, sondern die Robustheit der Knoten gegenüber Attacken zu einem späteren Zeitpunkt. Sie gehen von einem relativ schwachen passiven Angreifer aus, der nicht das komplette Netz überwachen kann, sondern, zumindest zum Zeitpunkt der Ausbringung der Sensoren, nur einen Teil davon. Gerade neu installierte, in gegenseitiger Reichweite befindliche Sensoren tauschen neue Schlüssel untereinander einfach im Klartext aus. Sensoren, die sich gegenseitig nicht in

Funkreichweite befinden, bekommen mit Hilfe ihrer direkten Nachbarn Schlüssel vermittelt. Mit ihren direkten Nachbarn können die Sensoren dabei jeweils verschlüsselt kommunizieren.

Fraglich ist jedoch, ob solche Annahmen über schwache, ausschließlich lokal begrenzte Angreifer realistisch sind. Die Sicherheit dieses Vorschlags basiert stark auf der Annahme, daß ein Angreifer nicht vorausschauend die Örtlichkeiten für mögliche zukünftige Sensornetze überwacht. Ein Angreifer, der die Installation eines Sensornetzes erwartet, hat mit diesem Protokoll daher leichtes Spiel. Die Annahmen dieser Arbeiten stehen also im Widerspruch zum Angreifermodell dieser Arbeit und damit zum Entwurfsziel *Sicherheit*.

Komplett unbeantwortet bleibt außerdem die Frage nach dynamischem Hinzufügen oder Entfernen von Sensorknoten – bei diesem Vorschlag können nach dem einmaligen Ausbringen von Sensoren keine *neuen* sicher hinzugefügt werden. Das widerspricht dem Entwurfsziel *Dynamik*.

Das Protokoll LEAP aus Zhu et al. [257, 258] arbeitet ähnlich. In einer initialen Discovery-Phase, in der die Abwesenheit eines Angreifers angenommen wird, tauschen physikalisch benachbarte Knoten paarweise verschiedene Schlüssel völlig ungeschützt aus. Muß nach der (sicheren) Discovery-Phase ein Knoten i einen neuen Schlüssel $K_{i,j}$ mit einem anderen Knoten j austauschen, so versendet er einen Hilfe-Broadcast an alle Knoten in Funkreichweite. Diesem Broadcast antworten alle m Nachbarn von i, die auch Nachbarn von j sind. Knoten i teilt $K_{i,j}$ in m Teile auf $K_{i,j} = K_1 \oplus \ldots K_m$ und verschickt diese Teile chiffriert an seine m Nachbarn, mit denen er und j bereits Schlüssel besitzen. Da es in vielen Fällen keine Knoten gibt, die sich in direkter Nachbarschaft sowohl zu i als auch j befinden, $m = 0$, kommt bei LEAP in diesem Fall dann das Protokoll μTESLA [172, 174] zum Einsatz. Für LEAP und auch das damit verwandte Protokoll aus Westhoff et al. [241] gelten die gleichen Kritikpunkte wie für *Key Infection*. Bei LEAP kommt jedoch noch hinzu, daß im wahrscheinlich häufig vorkommenden Fall von $m = 0$ der μTESLA-Zugriff auf eine Basistation absolut notwendig wird – Widerspruch zum Entwurfsziel *Dezentralität*.

Auch in Dutertre et al. [85] gehen die Autoren davon aus, daß der Angreifer in einer *Bootstrapping*-Phase noch keine Knoten korrumpieren kann. In dieser Phase tauschen alle Sensoren mit Hilfe eines einzigen, allen Sensoren bekannten, symmetrischen Schlüssels K_1 gesichert Sitzungsschlüssel aus. Das anschließende Hinzufügen neuer Knoten zum Sensornetz geschieht ebenfalls in Phasen, wobei immer mehrere neue Sensorknoten in eine Phase i eingeteilt werden. Alle Sensoren einer Phase i teilen sich einen gemeinsamen Schlüssel K_i, mit dem sie ihren Schlüsselaustausch untereinander sichern. Damit auch Knoten verschiedener Phasen $i, j : i < j$ Schlüssel miteinander austauschen können, erhält jeder Knoten a der Phase i vom Benutzer vor dem Deployment eine Menge Schlüssel $K_{a,i+1}, K_{a,i+2}, \ldots, K_{a,i+s}$ sowie einmalig eine Zufallszahl R_a. Die Schlüssel $K_{a,j}$ berechnet der Benutzer dabei wie folgt: $K_{a,j} = E_{K_j}(R_a)$. Auf diese Weise dient der Schlüssel $K_{a,j}$ Knoten a zum Schlüsselaustausch mit allen Knoten der j-ten Phase. Ein Knoten der j-ten Phase kann $K_{a,j}$ berechnen, da er K_j besitzt. Dieser Ansatz hat zwei Nachteile: Sobald ein Knoten einer Phase i korrumpiert wird, sind *alle* Schlüssel, die Knoten der i-ten Phase untereinander ausgetauscht haben, automatisch *unsicher*. Zweitens, der Benutzer muß a priori die Gesamtanzahl der Phasen, die im kompletten Netzwerkleben vorkommen können, sehr genau abschätzen. Genauer: Er muß wissen, wann, wie häufig und in welcher Anzahl Knoten dem Netz beitreten werden, damit er eine Obergrenze der s Schlüssel, die jeder Knoten erhält, bestimmen kann. Schätzt er s zu hoch ein, verbrau-

chen die Schlüssel *ineffizient* viel Hauptspeicher, schätzt er s zu niedrig, können Knoten verschiedener Phasen untereinander nicht miteinander kommunizieren. Dies würde die *Funktionalität* des Ansatzes sehr einschränken.

Die Autoren Liu und Cheng [147] vereinen die Ideen über ein sicheres Deployment mit Public-Key Kryptographie. Die Menge der Knoten teilt sich automatisch auf in *Service*- und *Worker*-Knoten. Die Service-Knoten versorgen sämtliche in ihrer Funkreichweite befindlichen Worker-Knoten mit Schlüsseln. Dazu schickt ein Service-Knoten den Worker-Knoten in seiner Funkreichweite während der initialen, per Annahme sicheren Phase einen öffentlichen Schlüssel n eines Rabin-Public-Key-Verfahrens [181]. Wollen zwei Knoten in Funkreichweite eines Service-Knotens gemeinsam miteinander kommunizieren, fordern sie vom Service-Knoten jeweils ein Polynom-Share nach Blundo et al. [36] an. Damit können diese beiden Knoten dann sicher einen Schlüssel austauschen. Worker-Knoten, die sich nicht direkt in Funkreichweite ein und desselben Service-Knotens befinden, müssen einen *Schlüsselpfad* über mehrere, unter Umständen korrumpierte, Zwischenknoten konstruieren. Dieser Ansatz ist auf Grund der Voraussetzung über sicheres Deployment und dem Konstruieren eines Schlüsselpfades *unsicher*: Ein Man-in-the-Middle-Angriff ist möglich. Äußerst fragwürdig und in der Arbeit nicht weiter untersucht ist zudem der Einsatz der asymmetrischen Rabin-Verschlüsselung, die in etwa die selbe Komplexität wie RSA [188] besitzt: Wenn diese Verschlüsselung sich überhaupt auf der Sensor-Hardware implementieren läßt, dann handelt es sich um eine sehr *ineffiziente* Verschlüsselung.

3.2.4 Zufallsverteilte Schlüssellisten

Eine in der Literatur bisher sehr häufig zitierte, populäre Idee zum Austausch von Schlüsseln basiert auf der zufälligen Verteilung sogenannter *Schlüssellisten* (engl. *Key Pools*).

Die Idee ist dabei, daß der Benutzer vorab eine große Menge kryptographischer Schlüssel erzeugt und in einem Key Pool P speichert. Jeder Sensorknoten x, der dem Netz beitreten soll, erhält vom Benutzer von P eine Anzahl von Schlüsseln zufällig gewählt, die er auf seinem sogenannten *Key-Ring* R_x abspeichert. Nun besitzen zwei beliebige Knoten x und y, die miteinander kommunizieren wollen, mit einer bestimmten Wahrscheinlichkeit einen gemeinsamen Schlüsseln auf ihren Key-Ringen R_x und R_y. Falls nicht, dann können x und y über Knoten, mit denen sie Schlüssel besitzen, sogenannte *Schlüsselpfade* zueinander konstruieren.

Aufgrund ihrer besonderen Bedeutung sind solche *Random Key Pre-Distribution*-Protokolle im Rahmen dieser Arbeit genauer analysiert und auf Schwächen und mögliche Probleme hin untersucht worden, um so auch den Entwurf des eigenen, in Abschnitt 3.3 beschriebenen Protokolls besser bewerten zu können. Wegen des größeren Umfangs der durchgeführten Analyse und um den Lesefluß an dieser Stelle nicht zu stören, widmet sich ihr ein eigener Abschnitt: Anhang B. Das soll allerdings nicht darüber hinwegtäuschen, daß die Analyse dieser besonders viel zitierten Protokolle für die vorliegende Arbeit von besonderer Bedeutung gewesen ist.

Aus dieser Analyse geht hervor, daß die Protokolle, die auf zufallsverteilten Schlüssellisten basieren, so wie zum Beispiel Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [88, 89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251], nicht die Entwurfsziele *Sicherheit*, *Dynamik* und *Effizienz* erfüllen:

- Abschnitt B.2.1, S. 178 zeigt, daß diese Protokolle unsicher bzgl. des angenommenen byzantinischen Angreifermodells sind, bei dem es dem Angreifer gelingt, eine Teilmenge der Knoten zu korrumpieren. Bereits eine kleine Menge korrumpierter Knoten gefährdet die Sicherheit zwischen allen Knoten im Netz.

- In Abschnitt B.2.2, S. 183 wird gezeigt, daß ein Knotenausfall große Auswirkungen auf die Verfügbarkeit sogenannter *gültiger* Schlüssel hat. Schon bei einem geringen Anteil ausfallender Knoten verringert sich die Anzahl gültiger Schlüssel sehr stark, so daß keinerlei Kommunikation im Netz mehr möglich ist.

- Die Abschnitte B.2.3, S. 185 und B.2.4, S. 186 untersuchen Speicher- und Energieverbrauch. Der Speicherverbrauch steigt schnell in Dimensionen, die den verfügbaren Speicher typischen Sensorknoten wie dem MICA2 [66] bei weitem übersteigen. Weiterhin impliziert die Notwendigkeit der Rücknahme von Schlüsseln ein stark energieverbrauchendes *Fluten* des Sensornetzes.

3.2.5 Deterministisch verteilte Schlüssellisten

Ähnlich zu Protokollen, die auf zufällig verteilten Schlüssellisten basieren, existieren auch Protokolle, bei denen *deterministisch* verteilte Schlüssellisten zum Einsatz kommt. Genau wie bei den zufälligen Verfahren erzeugt der Benutzer in Çamtepe und Yener [53, 54], Ghosh [94], Lee und Stinson [140, 141], Sanchez und Baldus [198], Simonova et al. [210] vorab einen großen Key-Pool P. Jeder neue, dem Netz beitretende Knoten a erhält eine Teilmenge von Schlüsseln aus P für seinen Key-Ring R_a. Die Schlüssel werden jedoch nicht zufällig aus P ausgewählt, sondern anhand eines deterministischen Musters. Die Key-Ringe der Knoten werden berechnet aus zueinander paarweisen *orthogonalen Lateinischen Quadraten* [53], ähnlich dem Spiel Sudoku. Basierend auf diesen deterministischen Mustern, können einige theoretische Aussagen bewiesen werden: Die Arbeiten zeigen, daß im Vergleich zu zufällig verteilten Schlüssellisten weniger Speicher und weniger Kommunikation notwendig ist, beides steigt nur logarithmisch mit der Gesamtzahl aller Knoten im Netz. Obwohl der Speicherverbrauch, genauer die Größe der Key-Ringe auf jedem Knoten, nur logarithmisch steigt, sind dennoch immer mehr Schlüssel auf allen Key-Ringen vorhanden als tatsächlich zur Kommunikation notwendig. Es wird demnach Hauptspeicher *ineffizient* für unnötige kryptographische Schlüssel genutzt – besonders bei den in Bezug auf Hauptspeicher begrenzten Sensorknoten kritisch. Weiterhin können auch die deterministischen Ansätze nicht mit *Dynamik*, umgehen und benötigen auf Netzwerkfluten basierende, *ineffiziente*, Mechanismen zur Zurücknahme von Schlüsseln. Ebenso bleibt die Problematik bestehen, daß ein Angreifer durch Kompromittieren einiger Knoten in den Besitz von Key-Ringen kommt und dadurch viele andere Kommunikationsassoziationen abhören kann. Diese Verfahren sind demnach nicht *sicher* in Bezug auf das dazu aufgestellte Entwurfsziel.

Die Arbeiten Chorzempa et al. [61, 62], Wadaa et al. [232] teilen alle Sensorknoten in mehrere Cluster ein. Innerhalb eines Clusters kommuniziert der Cluster-Head direkt mit der Basisstation. Der Cluster-Head übernimmt auch den Austausch von Schlüsseln der Sensoren innerhalb seines Clusters. Dazu benutzt er ein Verfahren aus Eltoweissy et al. [86]: *Exclusion Basis System*. Aus einer Menge von insgesamt m verschiedenen Schlüsseln erhält jeder Knoten eine Liste von $k < m$ vielen. Dabei wird sichergestellt, daß sich die Schlüssellisten von je zwei Knoten in mindestens einem Schlüssel unterscheiden. Auf diese Weise kann zwischen Cluster-Head und jedem Knoten im Cluster sicher,

das heißt authentisch, vertraulich, integer usw. kommuniziert werden, indem der Cluster-Head beziehungsweise ein Knoten seine Daten mit allen k Schlüssel hintereinander chiffriert. Es ist nur gesicherte Kommunikation zwischen Knoten eines Clusters und ihrem Cluster-Head möglich. Kommunikation zwischen Knoten innerhalb eines Clusters funktioniert nur über einen allen Knoten des Clusters bekannten Schlüssel und ist damit, sobald ein Knoten des Clusters kompromittiert wird, unsicher. Kommunikation zwischen Knoten verschiedener Cluster ist nicht möglich, das heißt, die Arbeiten Chorzempa et al. [61, 62], Wadaa et al. [232] erfüllen nicht das geforderte Entwurfsziel der *Funktionalität*. Falls ein Angreifer mehrere Knoten kompromittiert, besteht die Möglichkeit, daß er alle m Schlüssel eines Clusters rekonstruieren kann. Berechnungen in Chorzempa et al. [62] zeigen, daß *unabhängig* von der Größe des gesamten Netzes die Wahrscheinlichkeit alle m Schlüssel zu berechnen für einen Angreifer stark steigt: Bereits 14 Knoten reichen aus, um mit mehr als 60% Wahrscheinlichkeit alle m Schlüssel zu berechnen. Da ein Angreifer in vielen Fällen wesentlich mehr als 14 Knoten korrumpieren kann, sind auch diese Ansätze als *unsicher* zu betrachten.

Bei PIKE, Chan und Perrig [57], schätzt der Benutzer zunächst die voraussichtliche Gesamtgröße n des Sensornetzes ab. Dann berechnet er eine Menge $M, |M| = n(\sqrt{n} - 1)$ verschiedener symmetrische Schlüssel. Sobald nun Sensoren dem Netz beitreten, erhalten diese eine aufsteigende ID der Form (x, y). Der Benutzer vergibt IDs an die Knoten nach dem Schema

$$(0,0),(0,1),\ldots,(0,\sqrt{n}-1),(1,0),\ldots,(\sqrt{n}-1,\sqrt{n}-1).$$

Zusätzlich zur ID erhält jeder Knoten noch eine Reihe von Schlüsseln, die anhand einer Matrix der Dimension $(\sqrt{n} - 1) \times (\sqrt{n} - 1)$ bestimmt werden: Aus der Menge M erhält ein neuer Knoten (x, y) jeweils paarweise Schlüssel mit den Knoten der Matrix-Zeile (x, i) und Matrix-Spalte $(j, y), i, j \in \{0, \ldots, \sqrt{n} - 1\}$. Knoten (x, y) besitzt damit Schlüssel mit $2(\sqrt{n} - 1)$ anderen Knoten. Außerdem steht fest, daß zwei verschiedene Knoten (x, y), (x', y') entweder eine gemeinsame Zeile oder Spalte von Schlüsseln besitzen oder aber ihre verschiedenen Zeilen und Spalten sich in genau zwei Elementen der Matrix überschneiden. Diese beiden Schnittpunkte repräsentieren Knoten, für die sowohl (x, y) als auch (x', y') gemeinsame Schlüssel besitzen. Knoten (x, y) und (x', y') kennen gemeinsame Nachbarn in der Matrix. Über die gemeinsamen Nachbarn können (x, y) und (x', y') dann einen gemeinsamen Schlüssel austauschen. Falls es sich bei einem der beiden gemeinsamen Nachbarn um einen kompromittierten Knoten handelt, kann dieser den auszutauschenden Schlüssel abhören oder verändern. PIKE entspricht damit nicht dem Entwurfsziel *Sicherheit*. Der Speicheraufwand von PIKE ist mit $O(\sqrt{n})$ zwar geringer als der von beispielsweise zufällig verteilten Schlüssellisten, aber dennoch *nicht effizient*, da immer noch viel mehr Schlüssel im Speicher der Knoten gehalten werden muß, als tatsächlich zur Kommunikation notwendig. PIKE unterstützt zu dem keine *Dynamik*, da zwei Knoten dann über keine gemeinsamen Nachbarn mehr verfügen und Schlüsselaustausch unmöglich wird.

3.2.6 Weitere Arbeiten

Nun folgen noch weitere Veröffentlichungen, die von ihrer Idee eher „Mischformen“ sind und sich daher in kein bestimmtes Schema kategorisieren lassen.

Efficient Hierarchical Key Generation and Key Diffusion

In Shehab et al. [207] werden vor dem Deployment baumartig Schlüssel an die Sensoren verteilt. Der Wurzelknoten bekommt den Schlüssel k_0, seine Kinder $H(k_0)$, seine Enkel $H(H(k_0))$ usw. Vorgängerknoten im Baum können damit sämtliche Nachrichten ihrer Nachfolger lesen, allerdings nicht umgekehrt. Falls ein Knoten relativ weit "oben" im Baum kompromittiert wird, kann er nicht nur die Nachrichten all seiner Nachfolger lesen, sondern auch neue Nachrichten in ihrem Namen fälschen. HKDP ist *unsicher*. Es kommt hinzu, daß dem Netz so zur Laufzeit keine neuen Knoten beitreten können: Das Entwurfsziel *Dynamik* wird nicht erfüllt.

Kombination aus Schlüssellisten und bivariaten Polynomen

Die Arbeiten Liu und Ning [145], Moharrum und Eltoweissy [159], Zhou et al. [256] kombinieren den Einsatz von zufallsverteilten Schlüssellisten aus Abschnitt 3.2.4 mit bivariaten Polynomen aus Abschnitt 3.2.1. Der Benutzer berechnet vorab keinen Key-Pool P aus einzelnen symmetrischen Schlüsseln, sondern aus vielen bivariaten Polynomen $f_1(x,y), f_2(x,y), \ldots$. Jeder Knoten a erhält aus diesem Key-Pool zufällig eine Menge von *Shares* zum Beispiel $f_4(a,y), f_{27}(a,y), \ldots$ für seinen Key-Ring R_a. Falls zwei Knoten a, b auf ihren Key-Ringen mindestens ein gemeinsames Polynom finden, berechnen darüber ihren gemeinsamen Schlüssel $f(a,b)$. Im Vergleich zu reinen Schlüssellisten ist dieses Verfahren deutlich resistenter gegen kompromittierte Knoten: Der Angreifer muß nun pro Polynom mindestens t Knoten kompromittieren, um an die Schlüssel anderer Knoten zu gelangen. Dafür kombiniert das Verfahren jedoch auch den Aufwand der beiden zugrunde liegenden Ansätze. Pro Knoten müssen nun *mehrere* große Polynome im begrenzten RAM gespeichert werden. Dies wird bei Sensornetzen mit vielen Knoten und damit einhergehend auch vielen kompromittierten Knoten sehr *ineffizient*.

Bloms Schlüsselaustauschschema

Die Autoren Du et al. [83] modifizieren das Schlüsselverteilungsschema von Blom [35]. Der Benutzer generiert vorab zufällig zwei Matrizen G und D. Eine spezielle vandermondsche Matrix A hat die Dimension $(\lambda + 1) \times n$. Dabei gibt λ an, gegen wieviele korrumpierte Knoten der Schlüsselaustausch noch sicher funktionieren soll. Sie ist so aufgebaut, daß alle Elemente $e_{i,j}$ der i-ten Zeile Potenzen von $e_{i,1}$ sind. Die Elemente der i-te Zeile berechnen sich demnach aus $(e_{i,1}), (e_{i,1})^2, \ldots, (e_{i,1})^n$. Matrix D hat die Dimension $(\lambda+1) \times (\lambda+1)$. Parameter n stellt dabei die zu erwartende Gesamtzahl aller Sensorknoten dar. Matrix A ergibt sich aus $A = (D \times G)^T$. Jeder Knoten i bekommt nun bei seinem Deployment die i-Spalte der Matrix A sowie $e_{i,1}$ von G. Mit $e_{i,1}$ kann er die gesamte i-te Zeile von G rekonstruieren. Schlüssel lassen sich aus der Matrix $K = (A \times G)$ errechnen. Für zwei Knoten i, j gilt der gemeinsame Schlüssel $K_{i,j}$, den i und j wie folgt berechnen können: Knoten i gibt seine Spalte von A an j, und j gibt seine Spalte von A an i. Damit berechnen beide den gemeinsamen Schlüssel $K_{i,j} = K_{j,i}$ mit Hilfe ihrer jeweiligen Zeilen von G. Um dieses Schema gegen mehr als λ korrumpierte Knoten zu sichern, schlagen die Autoren vor, insgesamt τ-viele Matrizen G, D zu erzeugen und jeweils jedem Knoten $1 \leq \omega \leq \tau$ Spalten und erste Elemente initial auszuhändigen. Zwei Knoten können dann, sobald sie einen Schlüssel austauschen möchten, überprüfen, ob sie beide mindestens eine gemeinsame Matrix-Kombination aus ihren ω vielen gemein haben und darüber den Schlüssel wie gehabt berechnen. Der Nachteil dieses Verfahrens liegt im *ineffizienten*

Speicherverbrauch: Jeder Knoten muß insgesamt $\omega(\lambda + 1)$ viele symmetrische Schlüssel speichern. Die Anzahl der korrumpierten Knoten λ kann jedoch relativ groß sein. In einem Sensornetz aus 10000 Knoten sind 200 korrumpierte Knoten und mehr durchaus denkbar. Bei $\omega = 2$ muß dann schon jeder Knoten 3216 Bytes Hauptspeicher für Schlüssel verwenden. Außerdem ist der Ansatz *unsicher*: Wie bei zufälligen Schlüssellisten sind unter Umständen sogenannte *Schlüsselpfade* über mehrere Zwischenknoten notwendig. Diese könnten korrumpiert sein und den Schlüsselaustausch manipulieren.

Austausch über knotendisjunkte Schlüsselpfade

Bei den Protokollen aus Wacker et al. [229, 230, 231] bekommt ein neuer Knoten a vom Benutzer s verschiedene Schlüssel mit s zufälligen, sich bereits im Netz befindlichen Knoten. Der Benutzer muß dazu sowohl a als auch die s zufälligen alten Knoten manuell über einen externen, sicheren Kommunikationskanal, zum Beispiel physikalischer Kontakt, Berührung usw., erreichen. Der Schlüsselaustausch zwischen a und einem weiteren Knoten b, der nicht in der Liste der s anfangs zufällig ausgewählten Knoten ist, funktioniert wie folgt: Die s alten Knoten im Netz besitzen bereits Schlüssel mit anderen Knoten, die wiederum Schlüssel mit anderen Knoten besitzen usw. Knoten a versucht, über seine s Nachbarn einen knotendisjunkten (Schlüssel-)Pfad zu b zu finden. Über dieses Pfad kann a einen Schlüssel sicher mit b austauschen. Dieses Verfahren ist sicher gegen bis zu s kompromittierte Knoten. Der Kommunikationsaufwand für dieses Verfahren steigt allerdings *ineffizient* linear mit der Gesamtzahl der Knoten im Netz. Genauso steigt auch der Speicherverbrauch linear. Der initiale Schlüsselaustausch zwischen einem neuen Knoten und s alten Knoten benötigt außerdem physische Interaktion zwischen Benutzer und den alten Knoten und ist somit *nicht selbstorganisierend*.

3.2.7 Perfect Forward Secrecy

Zum Abschluß der Übersicht über den aktuellen Stand der Forschung sei hier noch *Perfect Forward Secrecy* (PFS) [97] diskutiert, eine äußerst wichtige Eigenschaft von Schlüsselaustauschprotokollen.

Langzeitschlüssel sind Schlüssel zwischen Knoten, die benutzt werden, um kurzfristige (Sitzungs-)Schlüssel auszuhandeln. Der häufige Wechsel der Sitzungssschlüssel (Re-Keying) erhöht die Sicherheit der Kommunikation. Die PFS-Eigenschaft sagt für einen Austausch von Sitzungsschlüsseln per Langezeitschlüssel aus, daß selbst wenn ein Angreifer in den Besitz des Langzeitschlüssel gelangt, davon der kurzfriste Schlüssel nicht gefährdet ist.

Re-Keying sowie die dazugehörende PFS-Eigenschaft sind in Protokollen zum Schlüsselaustausch in Sensornetzen allerdings völlig unüblich. Dies liegt daran, daß bislang die PFS-Eigenschaft ausschließlich durch Public-Key Mechanismen, insbesondere den Diffie-Hellman-Schlüsselaustaussch [156] beziehungsweise Varianten davon, zugesichert werden kann [219]. Diffie-Hellman und Public-Key Mechanismen wiederum gelten per Annahme als in Sensornetzen zu teuer, siehe die Diskussion in Abschnitt 2.6.1.6, S. 40.

Solange keine effizientere, für Sensornetze geeignetere Möglichkeit zur Perfect Foward Secrecy existiert, können Schlüsselaustauschprotokolle in Sensornetzen diese auch nicht erfüllen – und damit macht auch Re-Keying keinen Sinn. Die Entwicklung eines effizienten Perfect Forward Secrecy Mechanismus ist ein rein kryptographisches Problem und nicht Teil dieser Arbeit.

Ansätze	Entwurfsziele: *Funktionalität*	*Sicherheit*	*Dezentralität*	*Selbstorganisation, Spontaneität*	*Dynamik*	*Effizienz*
Public-Key Varianten						
[219](PKI)	√	√	◯	√	√	◯
[33, 48, 84, 108, 110, 132, 134, 137, 139, 146, 150, 152, 249, 253–255]	√	√	√	√	√	◯
Dedizierte Knoten						
[47, 138]	√	√	◯	√	◯	√
[71, 130, 172–174, 215]	√	√	◯	◯	◯	√
Sicheres Deployment						
[10, 241]	√	◯	√	√	◯	√
[257, 258]	√	◯	◯	√	◯	√
[85]	◯	◯	√	√	◯	◯
[147]	√	◯	√	√	√	◯
Zufallsverteilte Schlüssellisten						
[56, 58, 73, 74, 81, 88, 89, 109, 117, 135, 220, 251]	√	◯	√	√	◯	◯
Deterministische Schlüssellisten						
[53, 54, 57, 94, 140, 141, 198, 210]	√	◯	√	√	◯	◯
[61, 62, 232]	◯	√	√	√	√	◯
Diverse						
[207]	√	◯	√	√	◯	√
[145, 159, 256]	√	√	√	√	√	◯
[83]	√	◯	√	√	√	◯
[229–231]	√	√	√	◯	√	◯

Tabelle 3.1 Stand der Forschung *Schlüsselaustausch*

3.2.8 Zusammenfassung

Tabelle 3.1 faßt den Stand der Forschung zusammen und zeigt, welche der bisherigen Arbeiten welche der geforderten Entwurfsziele erfüllen. Erfüllen Arbeiten ein Entwurfsziel *nicht*, so ist dies in Tabelle 3.1 mit ◯ vermerkt, im anderen Falle mit √. Betrachtet man auf diese Weise den Stand der Forschung, so ist zunächst zu erkennen, daß bisher noch keine Arbeit alle geforderten Entwurfsziele erfüllt. Insbesondere stellt man fest, daß sehr

viele Veröffentlichungen *Effizienz*, *Dynamik* und *Sicherheit* nicht erfüllen. Das bedeutet, der Schlüsselaustausch verbraucht zuviel Hauptspeicher oder Energie, die Arbeiten unterstützen nicht den Ausfall von Knoten und können nicht mit byzantinischen Knoten umgehen.

Es besteht demnach noch Bedarf für ein Schlüsselaustauschprotokoll, das alle Entwurfsziele erfüllt und dabei insbesondere effiziente, rein symmetrische Mechanismen verwendet, weniger als $O(n)$ Hauptspeicher und Nachrichten benötigt, den Ausfall von Knoten erlaubt und sicher gegen korrumpierte Knoten ist.

3.3 Das Protokoll SKEY

Ein großes Problem der bisherigen Veröffentlichungen liegt in der Annahme über den möglichen Kommunikationsfluß in Sensornetzen. Die Arbeiten gehen davon aus, daß Knoten beliebig untereinander kommunizieren können müssen und daher auch jeweils paarweiser Schlüssel bedürfen. Diese Annahme gilt jedoch nicht für die dieser Arbeit zugrunde liegenden aggregierenden Sensornetze. Wie bereits in Abschnitt 3.1.1.4 beschrieben, müssen Sensoren in solchen Netzen ausschließlich anhand ihrer Aggregationsbeziehungen Daten miteinander austauschen.

Im Rahmen dieser Arbeit ist daher das Verfahren SKEY, *Secure KEYing*, entwickelt worden [25, 26], welches Schlüssel angepaßt an die sensornetztypische Aggregation verteilt. SKEY zeichnet sich im besonderen durch folgende Eigenschaften aus:

- SKEY erfüllt sämtliche der in Abschnitt 3.1.2 formulierten Entwurfsziele.
- SKEY benötigt pro Knoten lediglich logarithmischen Speicherverbrauch, eine logarithmische Anzahl von zu versendenden Nachrichten, logarithmische Anzahl kryptographischer Operationen und damit einen logarithmischen $O(\log n)$ Gesamtenergieverbrauch bei steigender Gesamtanzahl n von Knoten.
- Die Sicherheit des Protokolls gegenüber k (eine beliebige Ganzzahl) vielen, hintereinanderliegenden byzantinischen Knoten auf dem Aggregationspfad läßt sich parametrisieren.

Weiterer Aufbau dieses Kapitels

Die folgenden Abschnitte erklären die Funktionsweise von SKEY. Zunächst enthält Abschnitt 3.3.1 eine grobe Übersicht über die Idee und das Konzept hinter SKEY. Daran anschließend erklärt Abschnitt 3.3.3 sehr ausführlich und detailliert die induktive Funktionsweise des Protokolls. An den entsprechenden Stellen wird außerdem der Pseudocode von SKEY präsentiert und erklärt. Abschnitt 3.4 stellt die Evaluierung von SKEY im diskreten Ereignissimulator GloMoSim vor.

Zunächst k=1

Der Einfachheit halber und zum besseren Verständnis wird SKEY zunächsteinmal beispielhaft in Gegenwart *genau eines* korrumpierten Knotens *hintereinander* pro Aggregationspfad beschrieben, das heißt $k = 1$. Erst Abschnitt 3.3.5 geht auf die Erweiterung mit beliebigen k-korrumpierten Knoten, die hintereinanderliegen, ein.

3.3.1 Protokollidee

Die grundlegende Funktionsweise von SKEY ist induktiv über die Gesamtanzahl n aller Sensorknoten im Aggregationsbaum.

Induktionsvoraussetzung

Zu einem bestimmten Zeitpunkt sei die Induktionsvoraussetzung, daß alle Knoten x_i im Aggregationsbaum Schlüssel mit den Knoten auf ihrem Aggregationspfad $\mathbb{P}_{x_i}$ besitzen.

Induktionsschluß

Der Induktionsschluß ist, daß ein neuer Knoten, der dem Aggregationsbaum als Blatt hinzugefügt wird, mit Hilfe von SKEY Schlüssel austauscht. SKEY tauscht dabei so Schlüssel aus, daß

1. nach dem Schlüsselaustausch wieder die Induktionsvoraussetzung gilt und
2. dabei alle zuvor spezifizierten Entwurfsziele erfüllt werden.

Induktionsanfang

Für $n = 1$ Sensorknoten ist die Induktionsvoraussetzung trivialerweise erfüllt, da dieser Sensorknoten einen leeren Aggregationspfad besitzt.

Protokollablauf

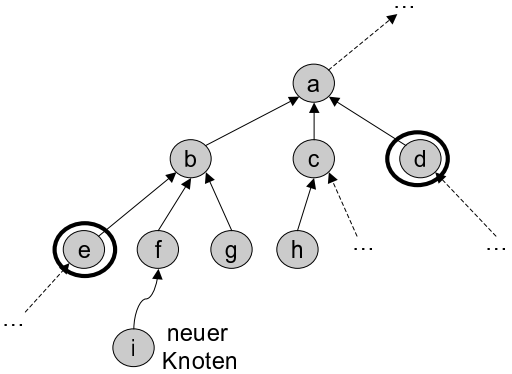


Abbildung 3.2 Beitritt von Knoten i, initiale Zufallsknoten sind e und d.

Die Situation sei im folgenden wie in Abbildung 3.2 dargestellt. Ein neuer Knoten i will dem Sensornetz als Nachfolger von f beitreten und muß dementsprechend kryptographische Schlüssel austauschen.

Das gesamte Vorgehen bei SKEY läßt sich nun wie folgt zusammenfassen:
Unmittelbar vor dem Deployment übergibt der Benutzer einem neuen Sensorknoten i eine Liste mit *Tickets* für eine Reihe von zufällig gewählten Sensorknoten $e, d, \ldots$, den sogenannten *initialen Zufallsknoten* IRN (Initial Random Nodes). Dieser Vorgang heißt bei SKEY Paaren (Pairing). Danach wird der Sensor im Netz ausgebracht.

Nun beginnt der eigentliche Protokollablauf von SKEY, den auch Abbildung 3.3 skizziert.

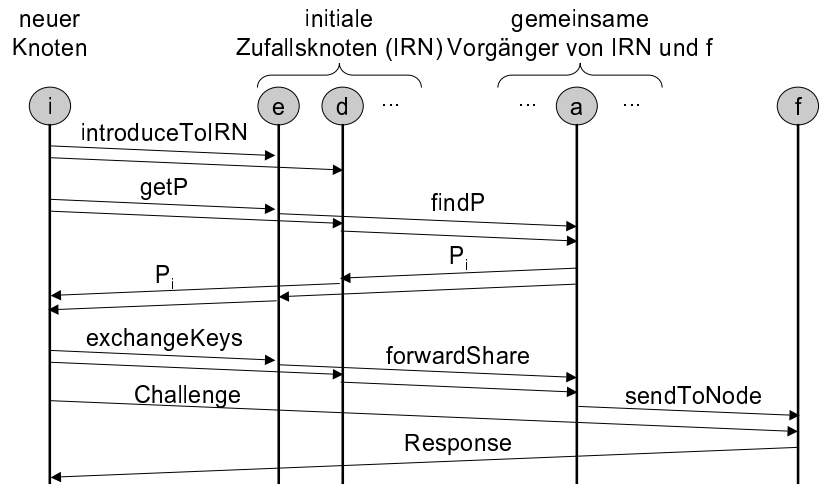


Abbildung 3.3 Veranschaulichung des Protokollablaufs

1. Der neue Sensor i meldet sich bei den IRN mit Hilfe der Funktion $\mathrm{introduceToIRN}$. Dazu benutzt er die zuvor vom Benutzer erhaltenen Tickets. Auf diese Weise erkennen die IRN, daß es sich bei i um einen legitimen Knoten handelt, der dem Netz beitreten möchte.

2. Sensor i bittet weiterhin alle IRN um Unterstützung bei der Suche nach i's Aggregationspfad $\mathbb{IP}_i$. Dies geschieht mit getIP.

3. Die IRN suchen auf ihren eigenen Aggregationspfaden mit findIP nach Vorgängern, die mit i auf Grund von Aggregationsbeziehungen Schlüssel austauschen müssen. Diese Knoten, zum Beispiel a, antworten mit einem Aggregationspfad $\mathbb{IP}_i$.

4. Die IRN überprüfen, ob die empfangenen Aggregationspfade $\mathbb{IP}_i$ für i alle übereinstimmen und schicken diesen Aggregationspfad $\mathbb{IP}_i$ an i.

5. Mit jedem Sensorknoten auf dem Aggregationspfad $\mathbb{IP}_i$, zum Beispiel dem Knoten f, will i nun sicher einen Schlüssel austauschen. Dazu generiert i einen zufälligen Schlüssel $K_{i,f}$ und zerlegt ihn in sogenannte *Shares* (Teile von Schlüsseln). Jeweils ein Share wird mit $\mathrm{exchangeKeys}$ an einen der IRN verschickt.

6. Die IRN e und d suchen nun auf ihren Aggregationspfaden nach Knoten, die auch Vorgänger von f sind. Im Beispiel seien das wieder $a, \ldots$. Die IRN teilen die von i empfangenen Shares nochmals in Teile auf und verschicken diese an die gemeinsamen Vorgänger von f, zum Beispiel a. Dies geschieht mit $\mathrm{forwardShare}$.

7. Schließlich schicken alle gemeinsamen Vorgänger wie a ihre erhaltenen Schlüsselteile mit $\mathrm{sendToNode}$ an f.

8. Knoten f kann damit aus allen empfangenen Schlüsselteilen den mit i gemeinsamen Schlüssel $K_{i,f}$ rekonstruieren.

9. Um zu überprüfen, ob i und f über einen gemeinsamen Schlüssel verfügen, führen sie ein Challenge-Response-Verfahren [156] durch.

3.3.2 Erläuterungen zum Pseudocode

Da im folgenden Abschnitt SKEY nicht nur erklärt, sondern auch auf dessen Pseudocode eingegangen wird, sind vorab noch einige Hinweise zu diesem Pseudocode notwendig. Im Pseudocode bezeichnet:

- die Schreibweise $a \rightarrow b : C$, daß der Knoten mit der ID a eine Nachricht an den Knoten mit der ID b *sendet*. Inhalt dieser Nachricht ist das Datum C.
- die Schreibweise $a \leftarrow b : C$, daß der Knoten mit der ID a eine Nachricht vom Knoten mit der ID b *empfängt*. Inhalt dieser Nachricht ist das Datum C.
- Π das Array mit den IDs der Vorgänger eines Knotens x im Aggregationsbaum. So bezeichnet zum Beispiel $\Pi[1]$ den direkten Vorgänger von x im Baum. Das Π eines Knotens x *im Pseudocode* entspricht damit trivialerweise nur den Vorgängerknoten aus dem im Grundlagenkapitel definierten *Aggregationspfad* $\mathbb{P}$, $\Pi \subseteq \mathbb{P}$. Für einen neuen Knoten, der dem Netz beitritt und für den noch keine Nachfolger im Aggregationsbaum existieren, gilt $\mathbb{P} = \Pi$.
- i den neuen Knoten, der dem Sensornetz hinzugefügt wird.
- $i.\mathrm{purpose}$ die „Dienstbeschreibung“ von i, beispielsweise „Temperatursensor“. Diese Information könnte auf dem Sensor beispielsweise bei seiner Fertigung ab Werk gespeichert worden sein. Aber auch andere Möglichkeiten sind denkbar, so zum Beispiel, daß der Benutzer dem Knoten diese Information vor seiner Installation zuordnet oder sogar das Sensornetz dies für den Knoten auf sichere Weise selbst bestimmt.
- $\mathrm{MAXNODES}$ die Anzahl der Knoten, die dem Sensornetz bisher hinzugefügt worden sind.
- IRN die *Initial Random Nodes*, die initialen Zufallsknoten, die dem neuen Sensor i helfen, seine Schlüssel auszutauschen.
- k die maximale Anzahl der korrumpierten Knoten, die auf einem Aggregationspfad hintereinanderliegen.
- error beziehungsweise **STOP_ALL_WORK** eine Fehlerfunktion sowie einen *Fehlerzustand*. Sobald ein Knoten ein Fehlverhalten bemerkt, beispielsweise durch unterschiedliche Antworten bei der Suche nach seinem Aggregationspfad $\mathbb{P}$, oder bei der Verifikation des ausgetauschten Schlüssels, schickt er eine Warnmeldung an die Senke und schaltet in einen Fehlerzustand. Was das bedeutet, wird am Ende des Kapitels diskutiert.
- ein Kommentar mit geschweiften Klammern $\{\ldots\}$ eine Erklärung, was genau in den nächsten Zeilen Pseudocode passieren wird.

In Anlehnung an Abbildung 3.3 beschreibt Algorithmus 1 den groben Protokollablauf von SKEY in Pseudocode. Er wird im folgenden Abschnitt im Detail erläutert.

Eingabe: Node i, int k, Key K_{MD}, int MAXNODES, String i.purpose
1: {Benutzer paart neuen Knoten i mit Hilfe seines Masterdevice MD:}
2: pairNode$(i, k, K_{MD}, \text{MAXNODES})$ {siehe Algorithmus 2}
3:
4: {Benutzer installiert i physisch im Sensornetz, *Deployment*:}
5: deployNode(i) {nicht Teil dieser Arbeit}
6:
7: {i nimmt Verbindung mit IRN auf:}
8: i.introduceToIRN() {siehe Algorithmus 3}
9:
10: {i sucht sich seinen Aggregationspfad $\mathbb{P}$ mit Hilfe der IRN:}
11: i.get$\mathbb{P}$ {siehe Algorithmus 9}
12:
13: {i tauscht mit Knoten aus $\mathbb{P}$ mit Hilfe der IRN Schlüssel aus:}
14: i.exchangeKeys(){siehe Algorithmus 4}

Algorithmus 1 Pseudocode von SKEY: Hinzufügen eines neuen Knotens i ins Sensornetz

3.3.3 Protokollbeschreibung

Grundsätzlich arbeitet das gesamte Protokoll *induktiv* mit der Gesamtanzahl der Knoten im Netz n. Man nehme an, daß zu einem bestimmten Zeitpunkt der Teilbaum eines insgesamt größeren Aggregationsbaums wie in Abbildung 3.2 aufgebaut ist.

Die Knoten e, f und g senden ihre gemessenen Werte an b, welcher diese aggregiert und das daraus resultierende Aggregat an a weiterleitet usw. Die Kanten im Graph aus Abbildung 3.2 repräsentieren wie gehabt jeweils eine Aggregationsbeziehung zwischen zwei Knoten. Dies bedeutet, daß zwischen zwei benachbarten Knoten, wie zum Beispiel g und b, paarweise Schlüssel existieren – allerdings auch zwischen g und a, siehe Abschnitt 3.1.1.4. Grundsätzlich gibt es Schlüssel zwischen jedem Knoten im Baum und all seinen Vorgängern und Nachfolgern, den Knoten auf seinem Aggregationspfad. Die aktuelle Konfiguration des Aggregationsbaums zu diesem Zeitpunkt und damit einhergehend der Besitz paarweiser Schlüssel stellt die *Induktionsvoraussetzung* dar.

Zu diesem Zeitpunkt soll nun i dem Netzwerk beigefügt werden. Laut Aggregationsbaum soll i zukünftig mit den Knoten $f, b, a, \ldots$ sicher kommunizieren, und dafür müssen zwischen diesen Knoten und i Schlüssel ausgetauscht werden. Woher allerdings Knoten i zu diesem Zeitpunkt genau weiß, daß er auf diesem Aggregationspfad, mit den Knoten $\mathbb{P} = \{f, b, a, \ldots\}$, liegt und damit Schlüssel mit diesen Knoten benötigt, soll an dieser Stelle zunächst unerheblich sein und wird im Detail in Abschnitt 3.3.7 erklärt. Man nehme nun vorübergehend an, daß i diejenigen Knoten, für die er zukünftig Schlüssel braucht, kennt.

3.3.3.1 Initiales Paaren über ein Master Device

Wie in Abschnitt 3.1.1.3 erklärt, verfügt der neue Knoten i über keinerlei Informationen bezüglich des Sensornetzes: Knoten i muß daher auf seinen Einsatz im Sensornetz vorbereitet werden. Das heißt, i benötigt Informationen, um überhaupt seinen Schlüsselaustausch starten zu können. Dieses einmalige, initiale Vorbereiten von i heißt *Paaren* und wird mit pairNode aus Algorithmus 2 spezifiziert. store$(i$.target, value$)$ gibt hier an,

daß beim Paaren des Knotens i an die Speicherstelle target von Knoten i der Wert value geschrieben wird.

Eingabe: Node i, int k, Key K_{MD}, int MAXNODES
1: **Declaration:** ID m, Key $K_{m,i}$, Ticket T_m
2: {Paaren eines neuen Knotens i durch MD:}
3: $\mathrm{store}(i.\mathrm{ID}, \mathrm{MAXNODES} + 1)$
4: $\mathrm{store}(i.k, k)$
5: $\mathrm{store}(i.K_{\mathrm{MD},i}, E_{K_{\mathrm{MD}}}(i))$
6: **for** $j := 1$ to $2k + 1$ **do**
7: {Wähle zufällige initiale Knoten (IRN):}
8: $m := \mathrm{generateRandomNumber}(1 \ldots \mathrm{MAXNODES})$
9: $\mathrm{store}(i.\mathrm{IRN}[j].\mathrm{ID}, m)$
10: {Erzeuge Schlüssel:}
11: $K_{m,i} := \mathrm{generateRandomKey}()$
12: $\mathrm{store}(i.\mathrm{IRN}[j].\mathrm{Key}, K_{m,i})$
13: {Erzeuge Ticket:}
14: $T_m := E_{K_{\mathrm{MD},m}}(i, \mathrm{legitim}, K_{m,i})$
15: $\mathrm{store}(i.\mathrm{IRN}[j].\mathrm{Ticket}, T_m)$
16: **end for**
17: $\mathrm{MAXNODES} := \mathrm{MAXNODES} + 1$

Algorithmus 2 $\mathrm{pairNode}(i, k, K_{MD}, \mathrm{MAXNODES})$

Der oder diejenigen Personen, die Sensoren im Sensornetz installieren, beispielsweise ein Administrator oder der Benutzer selbst, übernehmen nun die Aufgabe, i zu paaren. Sie fungieren als Trusted Dealer [65, 219]. Dabei kommt stellvertretend für den Menschen ein spezielles Gerät, ein sogenanntes *Master Device* (MD), zum Einsatz. Der Vorgang des Paarens selbst kann dabei über einen speziellen Kanal, einen sogenannten *Location Limited Channel* [13] geschehen. Ein Beispiel für einen solchen speziellen Kanal ist physikalischer Kontakt. Während des physikalischen Kontakts tauschen MD und i Informationen aus. Da ein solcher Kanal im allgemeinen als absolut abhörsicher angenommen wird, eignet er sich perfekt für den sicheren initialen Informationsaustausch zwischen MD und i, siehe Balfanz et al. [13], Nicholson et al. [165], Wacker et al. [229, 230, 231] und die im Rahmen dieser Arbeit entstandenen Blaß und Zitterbart [26], Blaß et al. [28], Hof und Zitterbart [102], Hof et al. [103, 104].

Typische Beispiele für geeignete MDs sind Ringe, Uhren oder Schlüsselanhänger. In Blaß und Zitterbart [26], Blaß et al. [28], Hof und Zitterbart [102], Hof et al. [103, 104], Wacker et al. [229, 230, 231] wird der Einsatz solcher Geräte für das initiale Paaren von Sensoren diskutiert: Da *jeder* Knoten einmalig mit dem MD gepaart werden muß, eignen sich insbesondere kleine, simple Geräte für diese Aufgabe, die der Benutzer ohne großen Aufwand mit sich führen kann. Es sei an dieser Stelle angemerkt, daß ein Master Device nur ein Werkzeug des Benutzers darstellt, das stellvertretend für ihn das Paaren übernimmt. Das MD übernimmt an dieser Stelle die Kommunikation zwischen Sensor und Benutzer.

Wie im folgenden gezeigt wird, beschränkt sich die Interaktion zwischen MD und dem Sensor ausschließlich auf die initiale Paarung vor dem Deployment des Knoten. Das MD benötigt dabei keinerlei Informationen über die aktuelle Netzwerktopologie oder dessen Konfiguration. Während des normalen, alltäglichen Betriebs des Netzes bleibt das MD offline unerreichbar für die Knoten und assistiert ihnen in keinster Weise. Es ist keine

Form einer Infrastrukturkomponente oder eines zentralen Online-Servers und tritt nur vor dem Ausbringen der Sensoren einmalig in Erscheinung. Es ist daher ein in Sensornetzen angemessenes Werkzeug zum Paaren oder Vorbereiten neuer Knoten. Geräte wie MD, die stellvertretend für den Benutzer lediglich vorab zum Paaren von Knoten dienen, nennt man auch häufig „(Trusted) Dealer“, siehe zum Beispiel Cramer und Damgård [65] oder van Tilborg [219].

Grundsätzlich gilt, daß ein solch spezielles Gerät besonders geschützt werden muß. Wird es gestohlen oder kompromittiert, so könnte ein Angreifer damit eigene, bösartige Knoten legal dem Netz beitreten lassen und unter Umständen die komplette Netzwerksicherheit ruinieren. Daher wird vorgeschlagen, daß die Hardware gegen Angriffe wie Korrumpierung geschützt ist (sog. *tamper-proof* Hardware) und der Benutzer das Gerät mit besonderer Sorgfalt einsetzt.

Details zum Paaren

Die Funktion $\mathrm{pairNode}$ spezifiziert die Operationen, die das MD ausführt und die Informationen, die das MD an den neuen Sensor i sendet. Während des Paarens, dem Moment des physikalischen Kontakts, geschieht das folgende:

Nur das MD kennt einen fest einprogrammierten Schlüssel K_{MD}. Dieser könnte zum Beispiel bei der Fabrikation eines bestimmten MDs vom Hersteller generiert worden sein und verläßt das MD niemals. Mit Hilfe von K_{MD} ist MD in der Lage, zu jeder Zeit paarweise verschiedene Schlüssel mit allen Sensorknoten zu berechnen. Jeder Knoten besitzt eine eindeutige ID, wie zum Beispiel f oder g oder irgendeine andere eindeutige Identifikation. Damit kann MD zu jeder Zeit einen gemeinsamen geheimen Schlüssel mit Knoten ID durch $K_{\mathrm{MD,ID}} = E_{K_{\mathrm{MD}}}(\mathrm{ID})$ berechnen, siehe auch Davis und Swick [70], Zhu et al. [257]. Während des Paarens schickt MD $K_{\mathrm{MD,ID}}$ an Knoten ID über den Location Limited Channel, Algorithmus $\mathrm{pairNode}$, Zeile 5. Knoten ID speichert $K_{\mathrm{MD,ID}}$ ab. MD braucht $K_{\mathrm{MD,ID}}$ hingegen nicht zu speichern, da es jederzeit wieder $K_{\mathrm{MD,ID}}$ mit Hilfe von K_{MD} berechnen kann. Das Master Device muß demnach keinerlei Informationen speichern und kann komplett zustandslos bleiben.

Als Ergebnis besitzt jeder Knoten, der einmal erfolgreich gepaart worden ist, einen gemeinsamen geheimen Schlüssel mit MD. Dadurch ist MD stets in der Lage, Nachrichten an einen Knoten ID zu verschicken, die mit $K_{\mathrm{MD,ID}}$ chiffriert sind. Empfängt ein Knoten ID solch eine Nachricht und kann sie mit $K_{\mathrm{MD,ID}}$ entschlüsseln, so weiß er, daß der Urheber dieser Nachricht MD gewesen ist [43]. Diese Technik erlaubt es MD, *Tickets* an Knoten zu verschicken. Solche Tickets werden im weiteren Verlauf des Protokolls von neuen Knoten dazu genutzt werden, sich alten, bereits im Netz befindendlichen Knoten vorzustellen. Vorstellen bedeutet hier, daß ein neuer Knoten mit Hilfe eines Tickets den alten Knoten beweisen kann, daß er ein legitimer, vom Benutzer hinzugefügter Knoten ist. Dies funktioniert ähnlich wie Kerberos [158], wobei das MD quasi dies Rolle des *Key Distribution Servers* (KDS) übernimmt, allerdings *offline*. Details dazu werden weiter unten beschrieben.

Wenn also ein neuer Knoten i dem Netz beitreten möchte, so generiert das MD den Schlüssel $K_{\mathrm{MD},i}$ und übergibt ihn an i. Danach wählt das MD *zufällig* zwei (beziehungsweise allgemeiner k+1) Knoten aus dem existierenden Netzwerk aus, im Beispiel aus Abbildung 3.2 die umrandeten Knoten e und d. Knoten e und d stellen die sogenannten $\mathrm{InitialRandomNodes}$ (IRN) von i dar, siehe Algorithmus $\mathrm{pairNode}$, Zeilen 7ff. Wie genau das MD zufällig zwei Knoten aus dem Sensornetz auswählen kann, ohne die aktuelle

Netzkonfiguration zu kennen, stellt Abschnitt 3.3.6 vor. An dieser Stelle soll zunächst wiederum nur angenommen werden, daß MD zwei beliebige Knoten e und d auswählen kann.

Schließlich berechnet MD zwei Tickets T_e und T_d sowie zwei Schlüssel $K_{e,i}$ und $K_{d,i}$, Zeilen 10ff. Zusammengefaßt überträgt das MD die folgenden Daten über den Location Limited Channel an i:

- $K_{\mathrm{MD},i}$, den geheimen Schlüssel zwischen i und MD. Knoten i kann damit zukünftig sicher Tickets von anderen, neuen Knoten entschlüsseln.
- Die IDs der initialen Zufallsknoten e und d sowie dafür paarweise symmetrische Schlüssel $K_{e,i}$ und $K_{d,i}$. Mit Hilfe der IDs und der Schlüssel weiß i, mit wem es im weiteren Verlauf des Protokolls sicher kommunizieren muß.
- Das Ticket $T_e = E_{K_{\mathrm{MD},e}}(\text{i, legitim, } K_{e,i})$ und
  das Ticket $T_d = E_{K_{\mathrm{MD},d}}(\text{i, legitim, } K_{d,i})$

1: **Declaration:** ID m
2: $\{i$ stellt sich den IRN vor:$\}$
3: **for** $j := 1$ to $2k + 1$ **do**
4: $\quad m := \mathrm{IRN}[j].\mathrm{ID}$
5: $\quad i \rightarrow m : \mathrm{IRN}[j].\mathrm{Ticket}$
6: **end for**

Algorithmus 3 $\mathrm{i.introduceToIRN}()$

Damit ist das Paaren beendet. Der Knoten i kann nun im Sensornetz plaziert werden. Anschließend schickt er mit Hilfe seiner normalen Funkschnittstelle die Tickets T_e und T_d an die Knoten e und d, vergleiche Spezifikation in $\mathrm{introduceToIRN}$, Algorithmus 3. Da beide Tickets mit den entsprechenden Schlüsseln zwischen dem MD sowie e beziehungsweise d chiffriert sind, wissen nun sowohl e als auch d, daß die Tickets ursprünglich vom MD stammen. Darüberhinaus sind sie durch den Ausdruck *legitim* (das heißt nicht-korrumpiert) im Ticket darüber informiert, daß es sich bei dem Knoten mit der ID i um einen gültigen Knoten handelt, und sie besitzen jeweils geheime symmetrische Schlüssel zur Kommunikation mit ihm. Knoten e und d speichern die Schlüssel $K_{e,i}$ und $K_{d,i}$ und verwerfen die Tickets.

3.3.3.2 Sichere Schlüsselweiterleitung

Nun beginnt i Schlüssel auszutauschen, siehe Spezifikation in $i.\mathrm{exchangeKeys}$, Algorithmus 4. Unter der zuvor getroffenen Annahme, daß i seinen Aggregationspfad $\mathbb{P} = \{f, b, a, \dots\}$ bereits kennt, generiert i nun einen neuen, symmetrischen Schlüssel $K_{i,f}$ für die sichere Kommunikation zwischen i und f und teilt $K_{i,f}$ mit Hilfe von $\mathrm{splitKey}$, siehe Algorithmus 6, perfekt in zwei Hälften K^1 und K^2. Da i bereits Schlüssel mit den initialen Zufallsknoten e und d besitzt, wird es diese Knoten benutzen, um die beiden Schlüsselhälften sicher an f weiterzuleiten.

Dies geschieht folgendermaßen:

1: **Declaration:** Key $K_{i,x}$, KeyShares $K^1, \ldots, K^{k+1}$, ID m, Key $K_{m,i}$, int l
2: {$\forall x \in i.\mathbb{P} : i$ sendet Key-Shares an alle IRN:}
3: **for all** $x \in \mathbb{P}$ **do**
4: $K_{i,x} := \mathrm{generateRandomKey}()$
5: $K^1, \ldots, K^{k+1} := \mathrm{splitKey}(K_{i,x}, k+1)$ {siehe Algorithmus 6}
6: **for** $j := 1$ to $2k+1$ **do**
7: {i sendet Shares an die IRN:}
8: $m := \mathrm{IRN}[j].\mathrm{ID}$
9: $K_{m,i} := \mathrm{IRN}[j].\mathrm{Key}$
10: $i \rightarrow m : C_j := (i, E_{K_{m,i}}(K^j, x))$
11: {m leitet anschließend K^j mit $\mathrm{forwardShare}(i, K^j, x)$ weiter, siehe Algorithmus 7}
12: **end for**
13: {i überprüft $K_{i,x}$ mit Challenge-Response-Verfahren:}
14: $l := \mathrm{generateRandomNumber}(1 \ldots 2^{64})$
15: $i \rightarrow x : (\mathrm{ID}, E_{K_{i,x}}(l))$
16: $i \leftarrow x : E_{K'_{i,x}}(l')$
17: {i erwartet $l+1$ als Antwort:}
18: **if** $l + 1 \neq l'$ **then**
19: **error** {siehe Algorithmus 5}
20: **end if**
21: **end for**

Algorithmus 4 i.exchangeKeys()

1: $m \rightarrow \Pi[|\Pi|] : E_{K_{m,\Pi[|\Pi|]}}(\mathrm{Error})$
2: **STOP_ALL_WORK**

Algorithmus 5 m.error

- Knoten i sendet
$$C_e = (i, E_{K_{e,i}}(K^1, f))$$
an e. Dies ist die Aufforderung von i an e, K^1 an f weiterzuleiten, exchangeKeys, Zeile 10. Das vorwegstehende Klartext „i“ in C_e hilft dem empfangenen Knoten e, das folgende Chiffrat als ursprünglich von i stammend zu verstehen – es kann mit $K_{e,i}$ entschlüsselt werden.

- In der selben Weise bittet i den Knoten d, Teilschlüssel K^2 an f weiterzuleiten. Er sendet
$$C_d = (i, E_{K_{d,i}}(K^2, f)).$$

Jetzt beginnt die sichere Weiterleitung der beiden Schlüsselteile mit Hilfe der Funktion forwardShare, siehe Algorithmus 7. Wenn Knoten e bereits einen gemeinsamen geheimen Schlüssel $K_{e,f}$ mit f hätte, so würde er einfach

$$\gamma_e = (e, E_{K_{e,f}}(i, K^1))$$

an f schicken, forwardShare, Zeile 5. Da jedoch in Abbildung 3.2 erkennbar ist, daß beide Knoten nicht auf einem Aggregationspfad liegen, existiert auch kein Schlüssel zwischen e und f. Daher versucht e nun sukzessive, einen Vorgänger auf seinem Aggregationspfad zu finden, der einen gemeinsamen Schlüssel mit f besitzt, Zeilen 8ff. (Der Ausdruck

Eingabe: Key K, int k
1: **Declaration:** KeyShares $K^1, \ldots, K^s$
2: {Teile Schlüssel K perfekt in s Key-Shares auf:}
3: **for** $j = 1$ to $s - 1$ **do**
4: $K^j := \mathrm{generateRandomNumber}(1 \ldots 2^{64})$
5: **end for**
6: $K^s := K^1 \oplus \ldots K^{s-1} \oplus K$
7: **return** $\{K^1, \ldots, K^s\}$

Algorithmus 6 $\mathrm{splitKey}(K, s)$, Idee siehe Abschnitt 2.6.3, S. 46

knowsNode soll dabei nur die Anfrage von e an einen Vorgänger darstellen, und response steht für dessen Antwort.) Knoten e fängt bei seinem direkten Vorgänger, dem Knoten b, an und fragt, ob b einen gemeinsamen Schlüssel mit d besitzt. Besäße b keinen gemeinsamen Schlüssel mit f, so würde e weiter fragen und zwar mit b's Vorgängern a usw. Da Knoten f an der Aggregation teilnimmt, muß auf jeden Fall einer von e's Vorgänger einen gemeinsamen Schlüssel mit f besitzen.

Alle Weiterleitungsanfragen an dieser Stelle sind mit den Schlüsseln gesichert, welche die Knoten jeweils mit ihren Vorgängern auf dem Aggregationspfad besitzen, siehe Zeile 12f. Aufgrund der Induktionsvoraussetzung kennen beispielsweise e und b den gemeinsamen Schlüssel $K_{e,b}$, Knoten e und a haben $K_{e,a}$ gemeinsam usw.

Im vorliegenden Beispiel besitzt bereits der direkte Vorgänger von e, sein Vaterknoten b, einen gemeinsamen Schlüssel $K_{b,f}$ mit f und meldet diese Tatsache zurück an e. Da e nun weiß, daß b über einen paarweisen Schlüssel mit f verfügt, muß auch dessen direkter Vorgänger a einen gemeinsamen Schlüssel mit f besitzen. Darüberhinaus existiert auch ein gemeinsamer Schlüssel $K_{e,a}$ zwischen a und e, da a auf dem Aggregationspfad von e liegt. Knoten e kennt demnach zwei Knoten, die je einen gemeinsamen Schlüssel mit f besitzen und kann mit diesen sicher kommunizieren.

Das Protokoll verläuft wie folgt weiter:

- e teilt K^1 in zwei Schlüsselhälften $K^{e,1}$ und $K^{e,2}$, Zeile 19.
- e berechnet
$$C_{e,1} = (e, E_{K_{e,b}}(i, f, K^{e,1}))$$
und
$$C_{e,2} = (e, E_{K_{e,a}}(i, f, K^{e,2})).$$
- Schließlich schickt e das Chiffrat $C_{e,1}$ an b und $C_{e,2}$ an a. Dies zeigt die Abbildung 3.4(a) – wobei in der Abbildung nicht die Chiffrate (C) dargestellt sind, sondern dem Verständnis halber nur der Fluß der Schlüssel-Shares wie K^1, $K^{e,1}$ usw. Zu erkennen ist, welches Key-Share von welchem Knoten erzeugt und verschickt wird.

  Abbildung 3.4(b) zeigt, welche Chiffrate wohin geschickt werden.

Nach dem Entschlüsseln schicken b und a an f:

$$b: \ \gamma_{e,1} = (b, E_{K_{b,f}}(i, K^{e,1}))$$

Eingabe: ID i, KeyShare K^m, ID x
1: **Declaration:** KeyShares $K^{m,1}, \ldots, K^{m,k}$
2: {IRN führen forwardShare aus, sobald sie ein Key-Share von i empfangen:}
3: **if** $x \in \mathbb{P}$ **then**
4: {m kennt x selbst und verschickt direkt an x:}
5: $m \rightarrow x : \gamma_m := (m, E_{K_{m,x}}(i, K^m))$
6: **else**
7: $j := 0$
8: **repeat**
9: {Suche Vorgänger, der x kennt:}
10: $j := j + 1$
11: {m fragt $\Pi[j]$, ob x Teil dessen Aggregationspfades ist:}
12: $m \rightarrow \Pi[j] : (m, E_{K_{m,\Pi[j]}}(\text{knowsNode}, x)))$
13: $m \leftarrow \Pi[j] : E_{K_{m,\Pi[j]}}(\text{response})$
14: **until** response = **true**
15: **if** $j < |\Pi| - k$ **then**
16: {Schicke K^m direkt an Wurzel:}
17: $m \rightarrow \Pi[|\Pi|] : (m, E_{K_{m,\Pi[|\Pi|]}}(K^m))$
18: **else**
19: $K^{m,1}, \ldots, K^{m,k} := \text{splitKey}(K^m, k)$
20: **for** $l := 0$ to $k - 1$ **do**
21: {Schicke Shares $K^{m,l}$ an k Vorgänger, die x kennen:}
22: $m \rightarrow \Pi[j + l] : C_{m,l+1} := (m, E_{K_{m,\Pi j+l}}(i, x, K^{m,l+1}))$
23: {Anschließend schickt Knoten $\Pi[j + l]$ das Share $K^{m,l+1}$ an x:
$\Pi[j + l] \rightarrow x : \gamma_{m,l+1} := (\Pi[j + l], E_{K_{\Pi[j+l],x}}(i, K^{m,l+1}))$,
siehe dazu Algorithmus 8.}
24: **end for**
25: **end if**
26: **end if**

Algorithmus 7 $m.\text{forwardShare}(i, K^m, x)$

$$a : \ \gamma_{e,2} = (a, E_{K_{a,f}}(i, K^{e,2})).$$

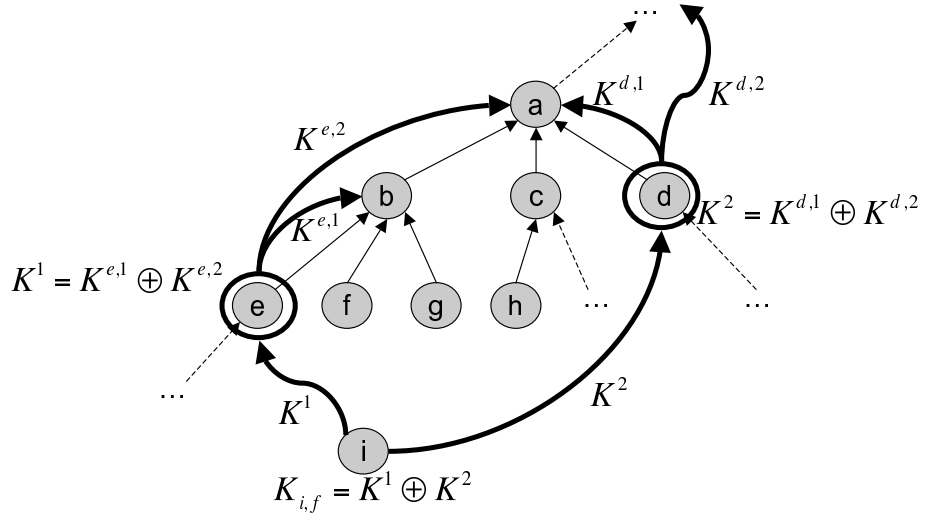
Dies wird mit sendToNode, siehe Algorithmus 8, spezifiziert und in Abbildung 3.5(a) gezeigt, wobei wiederum der Einfachheit halber zunächst nur der Fluß der Shares abgebildet ist. Abbildung 3.5(b) zeigt hingegen wieder den Versand der Chiffrate.

Eingabe: ID i, KeyShare K^m, ID x
1: {Knoten m schickt Share K^m an x:}
2: $m \rightarrow x : \gamma := (m, E_{K_{m,x}}(i, K^m))$

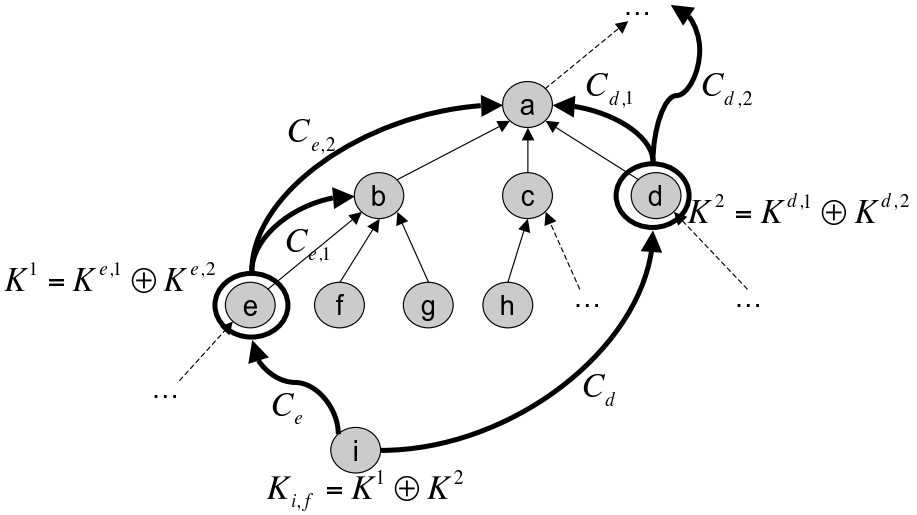
Algorithmus 8 $m.\text{sendToNode}(i, K^m, x)$

Analog handelt Knoten d, sobald er C_d empfangen hat. Er generiert zunächst $K^{d,1}$ und $K^{d,2}$. Dann bittet er seinen direkten Vorgänger a sowie a's Vater (nicht abgebildet in den Abbildungen 3.4 und 3.5) die beiden Hälften $K^{d,1}$ und $K^{d,2}$ an f weiterzuleiten. Es sei an dieser Stelle vorweggenommen, daß es kein Problem darstellt, falls a keinen Vaterknoten mehr besitzt, es sich bei a bereits um die Datensenke im Netz handelt, Zeilen 15f.

Schließlich empfängt f die Chiffrate $\gamma_{e,1}, \gamma_{e,2}, \gamma_{d,1}, \gamma_{d,2}$ und entschlüsselt sie zu $K^{e,1}$, $K^{e,2}$, $K^{d,1}$, $K^{d,2}$, siehe Abbildung 3.5(a).



(a) Weiterleitung der Schlüsselteile baumaufwärts



(b) Darstellung der versendeten Chiffrate

Abbildung 3.4 Teilen und Versenden von $K_{i,f}$. Shares werden chiffriert versendet, s. Text.

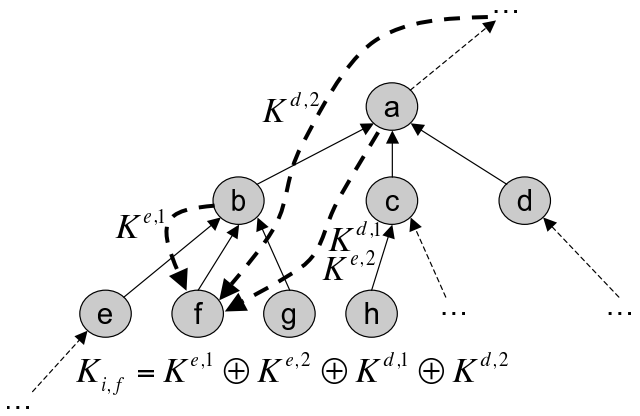
Damit kann f schließlich $K_{i,f}$ berechnen:

$$K_{i,f} = K^{e,1} \oplus K^{e,2} \oplus K^{d,1} \oplus K^{d,2}$$

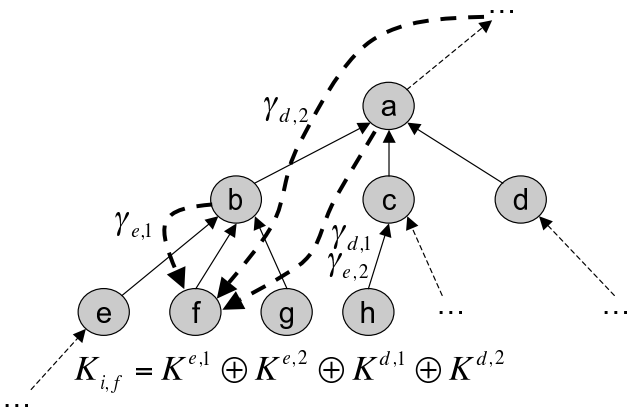
Wenn einer der beiden Knoten e oder d bereits einen gemeinsamen Schlüssel mit f besessen hätte, dann würde f nur $\gamma_e, \gamma_{d,1}, \gamma_{d,2}$ empfangen, daraus $K_1, K^{d,1}, K^{d,2}$ entschlüsseln und schließlich $K_{i,f}$ berechnen durch:

$$K_{i,f} = K^1 \oplus K^{d,1} \oplus K^{d,2}$$

Damit ist ein gemeinsamer geheimer Schlüssel $K_{i,f}$ zwischen i und f ausgetauscht. Knoten f kann zudem sicher sein, daß $K_{i,f}$ tatsächlich von i stammt, da sämtliche γ's diese Information enthalten. Knoten i kann noch mit Hilfe eines normalen Challenge-Response-Verfahrens [156] überprüfen, ob f auch tatsächlich den Schlüssel $K_{i,f}$ erhalten hat. Siehe dazu Algorithmus 4, exchangeKeys, Zeile 18.



(a) Knoten f empfängt Schlüsselteile und berechnet $K_{i,f}$



(b) Darstellung der versendeten Chiffrate

Abbildung 3.5 Zurücksenden der Schlüsselteile an f

Auf diese Weise kann i völlig analog auch Schlüssel mit b, a und allen anderen Vorgängern auf seinem Aggregationspfad austauschen. Knoten i bittet dabei jedesmal e und d, die neu generierten und aufgeteilten Schlüssel an i's Vorgänger weiterzuleiten.

Ein etwas komplizierteres Beispiel

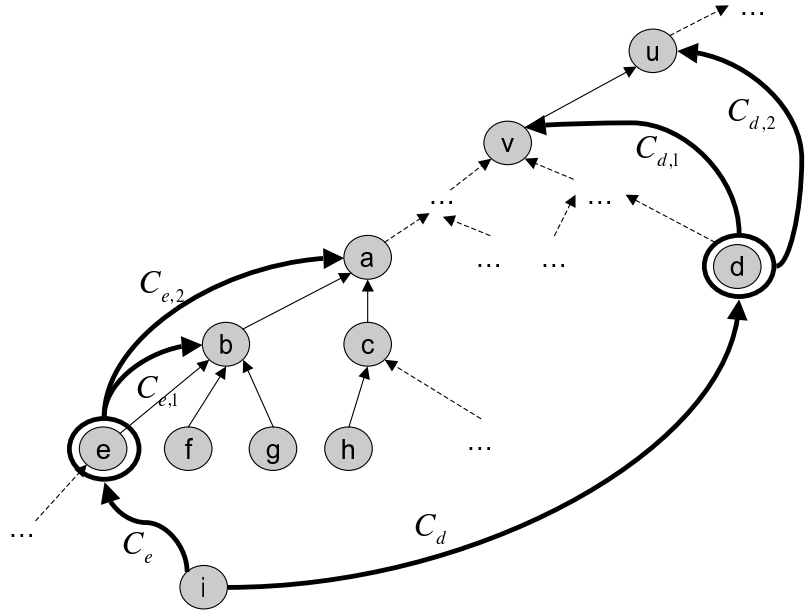
Im vorangegangenen, einfachen Beispiel war a direkter Vorgänger sowohl von d als auch Vorgänger vom Vorgänger von e. Damit hat a sowohl Shares von e als auch von d bekommen. Abbildung 3.6(a) zeigt den ersten Schritt von SKEY, den Versand der Chiffrate, in einem etwas komplizierteren Beispiel. Hier liegt d weit entfernt von e und auch f im Aggregationsbaum. Knoten d sucht dennoch solange nach einem Vorgänger auf seinem Aggregationspfad, bis er einen Vorgänger findet, der auch gleichzeitig Vorgänger von f ist. In diesem Falle ist das v. Es existiert in jedem Fall immer mindestens ein gemeinsamer Vorgänger zwischen d und f – mindestens die Senke. Wie zuvor schickt d nun $C_{d,1}$ an v und $C_{d,2}$ an den direkten Vorgänger von v, das ist u.

Da v und u auch auf dem Aggregationspfad von f liegen, können sie f seine Shares wie zuvor sicher zuschicken.

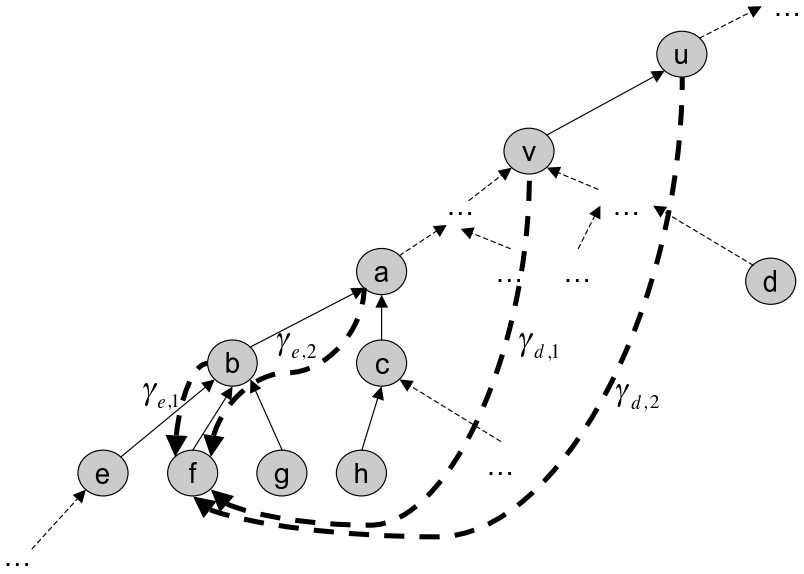
Abbildung 3.6(b) zeigt den Versand der Chiffrate baumabwärts zu f.

Induktionsschluß und Induktionsanfang

Hat i wie beschrieben mit seinen Vorgängern Schlüssel ausgetauscht, stellt dies automatisch den Induktionsschluß von SKEY dar: Alle Knoten im Netz besitzen Schlüssel mit



(a) Versenden der Chiffrate an Vorgängerknoten



(b) Versenden der Chiffrate an f

Abbildung 3.6 Versenden der Chiffrate, wenn u und v gemeinsame Vorgänger mit d und f sind

allen anderen Knoten, mit denen sie kommunizieren müssen – mit den Knoten auf ihrem Aggregationspfad.

Um die Induktion zu vervollständigen, fehlt allerdings noch der *Induktionsanfang* bei $n = 1$ Knoten. Dieser stellt sich als relativ trivial dar. Besteht das Netz nur aus der Basisstation ($n = 1$) und möchte nun ein neuer, zweiter Knoten dem Netz beitreten, dann teilt ihm das MD direkt einen Schlüssel mit der Basisstation aus. Der weitere Schlüsselaustausch ist damit nicht mehr notwendig. Selbiges gilt auch für ein Netz aus $n = 2$ Knoten. Ein neuer Sensor bekommt vom MD automatisch beide Schlüssel.

Reduzierung des Speicherverbrauchs

Jeder Aggregationsknoten benötigt paarweise Schlüssel mit jedem Knoten in seinen Unterbäumen. Dies hat zunächst die Konsequenz, daß Knoten im oberen Teil des Baumes sehr viele Schlüssel speichern müssen. So steigt die Anzahl der Schlüssel pro Knoten exponentiell mit der Höhe, auf der sich der Knoten im Baum befindet, siehe dazu auch Abschnitt 3.4.1. Damit Knoten im oberen Teil des Aggregationsbaums nicht zu viele Schlüssel speichern müssen, kommt dafür erneut eine zu Davis und Swick [70] ähnliche Technik zum Einsatz: Ähnlich wie das MD kennt jeder Knoten einen geheimen, unter Umständen fest einprogrammierten Schlüssel K_{ID}. So kennt zum Beispiel Knoten b den Schlüssel K_b. Diese Schlüssel können vom Hersteller generiert worden sein und verlassen die Knoten nie. Man nehme nun an, daß ein neuer Knoten i dem Netz beitritt und mit b, einem Vorgänger auf seinem Aggregationspfad befindet, einen gemeinsamen Schlüssel benötigt. Wie zuvor beschrieben tauschen beide einen neuen Schlüssel $K_{i,b}$ aus. Dieser Schlüssel wird jetzt jedoch nicht von den beiden Knoten als ihr zukünftiger gemeinsamer Kommunikationsschlüssel gespeichert, sondern nur temporär genutzt, um einen weiteren Schlüssel $K'_{i,b}$ wie folgt auszutauschen. Der Vorgängerknoten (oberhalb) im Aggregationspfad, b, berechnet den Schlüssel $K'_{i,b} = E_{K_b}(i)$ und schickt ihn sicher an i: $E_{K_{i,b}}(K'_{i,b})$. Knoten b kann jetzt sowohl $K_{i,b}$ als auch $K'_{i,b}$ aus seinem Speicher löschen. Knoten i entschlüsselt das empfangene Chiffrat mit $K_{i,b}$, gewinnt so $K'_{i,b}$, löscht $K_{i,b}$ und speichert schließlich $K'_{i,b}$. Von nun an nutzen beide Knoten $K'_{i,b}$ als ihren gemeinsamen Kommunikationsschlüssel: i hat ihn gespeichert und b kann ihn jederzeit berechnen über $K'_{i,b} = E_{K_b}(i)$, genau wie das MD in Abschnitt 3.3.3.1.

Diese Technik reduziert den für Schlüssel notwendigen Speicherverbrauch, da Knoten sich nur noch Schlüssel mit ihren *Vorgängern* auf dem Aggregationspfad merken müssen und nicht mehr mit sämtlichen *Nachfolgern*, die auf ihrem Aggregationspfad liegen. Zur Erinnerung: δ bezeichnet den durchschnittlichen Aggregationsgrad eines Aggregationsbaums, h bezeichnet die Höhe des Aggregationsbaums. Ohne die Reduzierung des Speicherverbrauchs würde ein Blatt im Aggregationsbaum im Durchschnitt $h \approx \log_\delta(1 + (\delta - 1)n)$ viele Schlüssel speichern müssen, genau so viele Schlüssel wie der Aggregationsbaum hoch ist plus den Schlüssel mit dem Master Device. Ein Knoten auf Höhe h' im Baum würde ohne die Verbesserung

$$h' + 1 + \sum_{i=1}^{h-h'} \delta^i$$

viele Schlüssel brauchen: h' mit seinen Vorgängern im Aggregationsbaum, ein Schlüssel für das MD sowie Schlüssel mit sämtlichen Knoten in seinem Teilbaum, in dem er die „Wurzel“ ist. Da die Höhe des Baums in Abhängigkeit der Gesamtanzahl der Knoten im Netz n logarithmisch wächst, entspricht dies linear steigendem Speicherverbrauch $O(n)$. Durch die vorgeschlagene Reduzierung jedoch braucht nun ein Knoten auf Höhe h' im Aggregationsbaum nur noch $h'+1$-viele Schlüssel, die Senke beispielsweise nur einen. Da die Höhe des Aggregationsbaums logarithmisch wächst, steigt auch der Speicherbedarf nur logarithmisch mit $O(\log n)$.

3.3.4 SKEY Sicherheit

Dieser Abschnitt diskutiert die Sicherheit des Protokolls zunächst gegenüber einem Angreifer mit den Fähigkeiten aus Abschnitt 2.4, S. 24. Es stellt sich heraus, vorweggenommen, daß ein Angreifer ausschließlich eine Art *Denial-of-Service*-Attacke (DoS) gegen

das Protokoll starten kann, um so einen erfolgreichen Schlüsselaustausch zu verhindern. Er wird jedoch niemals in den Besitz eines neuen Schlüssel kommen können, da sämtliche Daten individuell verschlüsselt sind.

Die möglichen Angriffe auf SKEY lassen sich grob in 3 Kategorien einteilen:

1. **Angriffe durch einen korrumpierten Knoten**, der am Protokoll teilnimmt. Da der Angreifer zufällig Knoten korrumpiert, kann es vorkommen, daß ein korrumpierter Knoten am Schlüsselaustausch teilnimmt. Dieser Knoten wird versuchen, in den Besitz des auszutauschenden Schlüssels zu gelangen.

2. **Denial-of-Service-Angriffe**. Der Angreifer versucht, den Schlüsselaustausch zu verhindern, genauer: Er will verhindern, daß i und f über einen gemeinsamen Schlüssel $K_{i,f}$ verfügen.

3. **Spoofing-Angriffe** durch Veränderung von Absenderinformationen in Nachrichten. Da der Angreifer laut Angreifermodell in der Lage ist, beliebige per Funk übertragene, nicht chiffrierte Daten zu modifizieren, kann er auch Absenderadressen von Nachrichten fälschen.

3.3.4.1 Sicherheit gegen korrumpierte Knoten

Die Sicherheit von SKEY basiert auf der Idee, einen neuen Schlüssel $K_{i,f,}$ zwischen i und f in einzelne Teile (Shares) zu zerlegen und diese so im Netz zu verteilen, daß bis auf f kein einzelner Knoten im Netz $K_{i,f}$ komplett berechnen kann.

Der aus Sicherheitssicht ungünstigste Fall

Aus Sicherheitssicht wäre die folgende Situation die „bedrohlichste“: Beide initiale Zufallsknoten liegen derart im gleichen Teilbaum, so daß beide denselben Vorgängerknoten finden, der einen gemeinsamen Schlüssel mit f besitzt. Seien beispielsweise h und c die beiden Zufallsknoten. Beide werden a als denjenigen Vorgänger im Baum erkennen, der einen gemeinsamen Schlüssel mit f besitzt. Falls nun Knoten a böse sein sollte, so bekommt er dennoch nicht sämtliche Schlüsselteile, da h und c jeweils auch Teile sicher an den Vaterknoten von a schicken. In einem weiteren Szenario könnte Knoten c der angreifende Knoten sein: Knoten c würde hier auf die Anfrage von Knoten h, ob c einen gemeinsamen Schlüssel mit f besitzt, lügen und bejahen. Trotzdem wird h einen Schlüsselteil an a verschicken. Demzufolge verfügt c nur über eine unvollständige Teilmenge aller Shares.

An dieser Stelle sei darauf hingewiesen, daß während des Schlüsselaustauschs einer der Vorgängerknoten im Aggregationsbaums die Senke selbst sein kann. Dies stellt jedoch kein Problem dar, denn diese Arbeit geht von einer vertrauenswürdigen, nicht kompromittierten Senke aus: Falls die Senke kompromittiert wird, macht auch ein sicherer Datentransport zu ihr keinen Sinn mehr. Wenn demnach einem initialen Zufallsknoten e während eines Protokollaustauschschrittes klar wird, daß die Senke einer der notwendigen Vorgängerknoten sein sollte, ist es nicht notwendig, weitere k Vorgänger zu suchen – e hat bereits einen Knoten, dem er vertrauen kann, gefunden.

Grundsätzlich gilt, daß auf diese Weise die beiden initialen Zufallsknoten *immer* eine knotendisjunkte Menge von Vorgängern finden, an die sie ihre Schlüsselteile weiterleiten können. Auf diese Weise kommt ein Angreifer nie in den Besitz aller Schlüsselteile.

3.3.4.2 Denial-of-Service

Es ist jedoch für den Angreifer möglich, chiffrierte Teilschlüssel abzufangen und zu verwerfen, also das Chiffrat vor der Weiterleitung zu modifizieren oder die Weiterleitung selbst zu verhindern. Er kann jedoch keinerlei gezielte Modifikationen an den chiffrierten Teilschlüsseln vornehmen. Dies bedeutet unvollständige oder fehlerhafte Schlüsselmengen beim Zielknoten f, die dazu führen, daß f einen Schlüssel $K'_{i,f}$ rekonstruiert, der sich von $K_{i,f}$ unterscheidet. Die Knoten i und f können folglich nicht miteinander kommunizieren – ein Denial-of-Service-Angriff. Solch einen DoS-Angriff zu erkennen ist jedoch relativ einfach: i und f können ihre gegenseitig ausgetauschten Nachrichten nicht entschlüsseln. Will i den ausgetauschten Schlüssel überprüfen, könnte er eine Zufallszahl l chiffrieren und an f schicken, $C = E_{K_{i,f}}(l)$. Knoten i erwartet nun von f als Antwort ein verschlüsseltes $l + 1$. Jedoch entschlüsselt f das Chiffrat bereits jetzt schon falsch mit seinem Schlüssel $K'_{i,f}$: $l' = D_{K'_{i,f}}(C)$. Knoten f schickt an i zurück: $C' = E_{K'_{i,f}}(l' + 1)$. Knoten i entschlüsselt wieder zu: $l'' = D_{K_{i,f}}(C')$. Da $l + 1 \neq l''$ fällt i schließlich auf, daß ein falscher Schlüssel ausgetauscht worden ist. Auf diese Weise kann f verifizieren, ob zwischen i und f der gleiche Schlüssel existiert. Solch ein Verfahren nennt sich Challenge-Response-Verfahren [156] und ist in Algorithmus 4, $\mathrm{exchangeKeys}$ Zeile 18, enthalten. In diesem DoS-Fall ist nur ungeschützte Kommunikation im Klartext zwischen beiden Knoten möglich, so wie komplett ohne Schlüsselverteilung. Ein Angreifer kann dabei auf keinen Fall f dazu bringen, einen *bestimmten* Schlüssel zu rekonstruieren (Man-in-the-Middle-Angriff), da er sich nicht im Besitz aller Schlüsselteile befindet.

Die Bedeutung eines solchen DoS-Angriffs bleibt jedoch fraglich: Der Schlüsselaustausch wird verhindert. Grundsätzlich sind Sensornetze aufgrund der simplen Hardware sehr anfällig gegen DoS-Angriffe. Da Angreifer Sensorknoten kompromittieren können, sind sie dadurch in der Lage, viele Arten von DoS-Angriffen zu starten. Das simple *Verhindern* des Schlüsselaustauschs durch Verwerfen oder Modifizieren von Nachrichten stellt dabei den harmlosesten Angriff dar. DoS-Angriffe wurden bereits im Grundlagenkapitel als in dieser Arbeit akzeptabel hingenommen.

3.3.4.3 Spoofing

Jeder Sensorknoten besitzt eine eindeutige ID, zum Beispiel i, mit der ihn andere Knoten adressieren. Grundsätzlich besteht dabei die Gefahr sogenannter Adress-*Spoofing*-Angriffe, siehe unter anderem Baker und Savola [12], CA CERT [46], Ferguson und Senie [90, 91], bei denen sich ein Knoten i' als ein Knoten i auszugeben versucht. Knoten i' maskiert sich als Knoten i.

Da allerdings in SKEY jeder Knoten mit einer ID i vom Master Device Tickets erhält, welche die ID von i enthalten, sind Spoofing-Angriffe nicht möglich: Die initialen Zufallsknoten e und d können nur mit dem Knoten kommunizieren, der vom MD die Schlüssel $K_{e,i}$ und $K_{d,i}$ während des Paarens erhalten hat. Das ist der Knoten i. Die initialen Zufallsknoten helfen auch nur diesem Knoten i beim Schlüsselaustausch mit ausschließlich dessen Vorgängern im Aggregationsbaum. Dabei teilen sie den Vorgängern von i bei jedem Schlüsselaustausch auch die Identität des Knotens mit, der den Schlüssel austauschen möchte. Ein Knoten i' kann sich damit nicht als Knoten i ausgeben, wenn er nicht die Schlüssel von i besitzt.

3.3.5 Erweiterung auf k korrumpierte Knoten

Der sogenannte *Schlüsselaustauschpfad* oder kurz *Austauschpfad* sei hier definiert als die Menge der Knoten, an die in den Funktionen $\mathrm{exchangeKeys}$ sowie $\mathrm{forwardShare}$

während eines Schlüsselaustausches hintereinander Key-Shares verschickt werden. Jeder Knoten, der im Rahmen dieser beiden Funktionen in den Besitz eines Key-Share kommt, ist Teil des Austauschpfades.

Beispiel: In der „komplizierteren“ Abbildung 3.6 besteht der Austauschpfad aus der Menge der Knoten $\{e, d, b, a, v, u\}$.

Wie bereits zu Beginn auf Seite 63 erwähnt ist der Schlüsselaustausch mittels SKEY bisher nur sicher in der Gegenwart eines einzelnen kompromittierten Knotens *hintereinander* auf dem Schlüsselaustauschpfad. Das bedeutet, SKEY ist bisher zum Beispiel auch dann sicher, wenn 2 oder mehrere Knoten auf einem Austauschpfad liegen, jedoch nicht hintereinander: Es ist sicher, sobald sich mindestens ein nicht korrumpierter Knoten zwischen zwei hintereinanderliegenden Knoten befindet. Die Begründung wurde oben bereits erwähnt: In diesem Fall kommt der Angreifer nicht in den Besitz *aller* Shares.

Das Protokoll SKEY ist jedoch parametrisierbar für eine beliebige Anzahl k von angreifenden Knoten auf dem Austauschpfad, die in ihren Möglichkeiten denen des Modells aus Abschnitt 2.4, S. 24 entsprechen. Die Anzahl k der Angreifer steht dabei nicht für die Gesamtanzahl von korrumpierten Knoten im Netz, sondern nur für die Anzahl korrumpierter Knoten, die unter Umständen an einem Protokolldurchlauf teilnehmen – eine weit geringere Zahl. Bei angenommener Gleichverteilung der korrumpierten bzw. angreifenden Knoten ist der relative Anteil korrumpierter Knoten auf dem Austauschpfad genauso hoch wie der Anteil korrumpierter Knoten im gesamten Netz, siehe Abschnitt 2.4, S. 24. An dieser Stelle wird nun erklärt, wie SKEY mit der Gegenwart von $k > 1$ Angreifern auf dem Austauschpfad umgeht.

3.3.5.1 Beschreibung

In der Gegenwart von $k > 1$ korrumpierten Knoten dürfen diese Knoten auch weiterhin nicht in den Besitz aller Shares zu gelangen. Genau wie im bereits diskutierten Fall mit $k = 1$ ist aus Sicherheitssicht auch hier wieder der bedrohlichste Fall der, daß alle k korrumpierten Knoten direkt hintereinander im Aggregationsbaum liegen.

Das MD wählt anstatt zwei jetzt initial insgesamt $k + 1$ Zufallsknoten aus. Ein neuer Knoten i teilt seinen neuen Schlüssel $K_{i,f}$ in $k + 1$ Shares $K^1, \ldots, K^{k+1}$, die er an seine initialen Zufallsknoten verteilt. Die Zufallsknoten teilen die empfangenen Shares weiter auf in jeweils $k + 1$ neue Shares, zum Beispiel $K^{1,1}, K^{1,2}, \ldots, K^{k+1,k+1}$. Jeder Zufallsknoten muß nun mit Hilfe der Funktion $\mathrm{forwardShare}$ $k + 1$ Vorgänger in seinem Aggregationsbaum finden, die jeweils einen gemeinsamen Schlüssel mit f besitzen. Wie oben beschrieben geschieht dies Schritt für Schritt. Sobald ein Zufallsknoten x einen Vorgänger y gefunden hat, der einen gemeinsamen Schlüssel mit f besitzt, schickt x ein Share an y sowie dessen k Vorgänger im Aggregationsbaum.

Abbildung 3.7 illustriert dieses Vorgehen. Die Knoten e, $d, \ldots$ sind die $k + 1$ initialen Zufallsknoten von i. Dieses suchen nach jeweils $k + 1$ gemeinsamen Vorgängern mit f, zu denen sie dann ihre Shares schicken.

Dementsprechend können k zusammenarbeitende, hintereinander auf dem Austauschpfad liegende Knoten auch nur k der insgesamt $k + 1$ Teilschlüssel kennen – es gibt immer mindestens einen Teilschlüssel, den ausschließlich ein nicht-kompromittierter, legitimer Knoten kennt. SKEY ist sicher, solange $k' \leq k$ Knoten auf jedem Austauschpfad vom Angreifer korrumpiert sind.

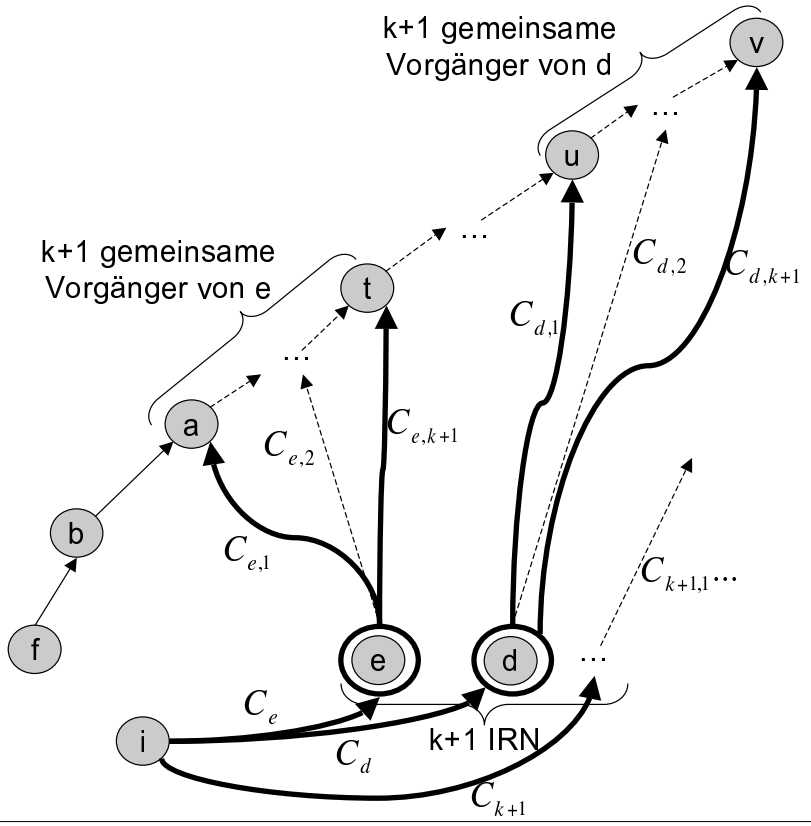


Abbildung 3.7 SKEY bei k korrumpierten Knoten

3.3.5.2 Bestimmen von k

Wie groß letztendlich k bei einer angenommenen Anzahl korrumpierter Knoten im Netz ist, hängt von der Höhe h des Aggregationsbaums ab und der Wahrscheinlichkeit, daß einzelne Knoten korrumpiert sind.

Unter den in Abschnitt 2.4, S. 24 getroffenen Annahmen gilt, daß ein Angreifer insgesamt $\mathcal{B}$ Knoten korrumpieren kann. In einem Netz bestehend aus insgesamt n Knoten korrumpiert er damit einen Anteil von $\beta = \frac{\mathcal{B}}{n}$. Bei der dabei angenommenen Gleichverteilung gilt, daß von jeder Teilmenge von Knoten auch β-Prozent korrumpiert sind. Die Anzahl der Knoten, die *maximal* auf einem Schlüsselaustauschpfad hintereinanderliegen ist $(h-1)$, die Höhe des Baumes ohne die als nicht-korrumpiert angenommenen Blätter. Bei einem Schlüsselaustausch können demnach im Durchschnitt *maximal*

$$\beta \cdot (h-1) = \frac{\mathcal{B}}{n} h \approx \frac{\mathcal{B}(\log_\delta(1+(\delta-1)n)-2)}{n} =: s(\delta, n, \mathcal{B})$$

korrumpierte Knoten hintereinanderliegen. Dieser Ausdruck konvergiert für steigende Knotenzahlen n und festen δ und $\mathcal{B}$ gegen 0. Der Benutzer kann damit jedoch, wenn er die Gesamtknotenzahl n, den erwarteten Aggregationsgrad δ und die Anzahl der möglicherweise korrumpierten Knoten $\mathcal{B}$ abschätzt, k bestimmen: k sollte so groß gewählt werden, wie maximal Angreifer auf dem Austauschpfad zu erwarten sind, das heißt $k \geq \lceil s(\delta, n, \mathcal{B}) \rceil$. (Die Funktion s muß deshalb aufgerundet werden, da k nur ganzzahlige Werte, die Anzahl der Vorgänger, annehmen kann.)

Eine weitere Bedingung für k ergibt sich aus der Forderung, daß der Angreifer von den $(k+1)$ initialen Zufallsknoten höchstens k korrumpiert hat und damit mindestens ein

Knoten legitim ist. Bei Gleichverteilung hat der Angreifer von den $(k+1)$ initialen Zufallsknoten im Durchschnitt $\beta \cdot (k+1)$ korrumpiert. Es muß gelten

$$\begin{aligned} \beta \cdot (k+1) &\leq k \\ \beta &\leq k - \beta h \\ \beta &\leq k \cdot (1-\beta) \\ \frac{\beta}{1-\beta} &\leq k. \end{aligned}$$

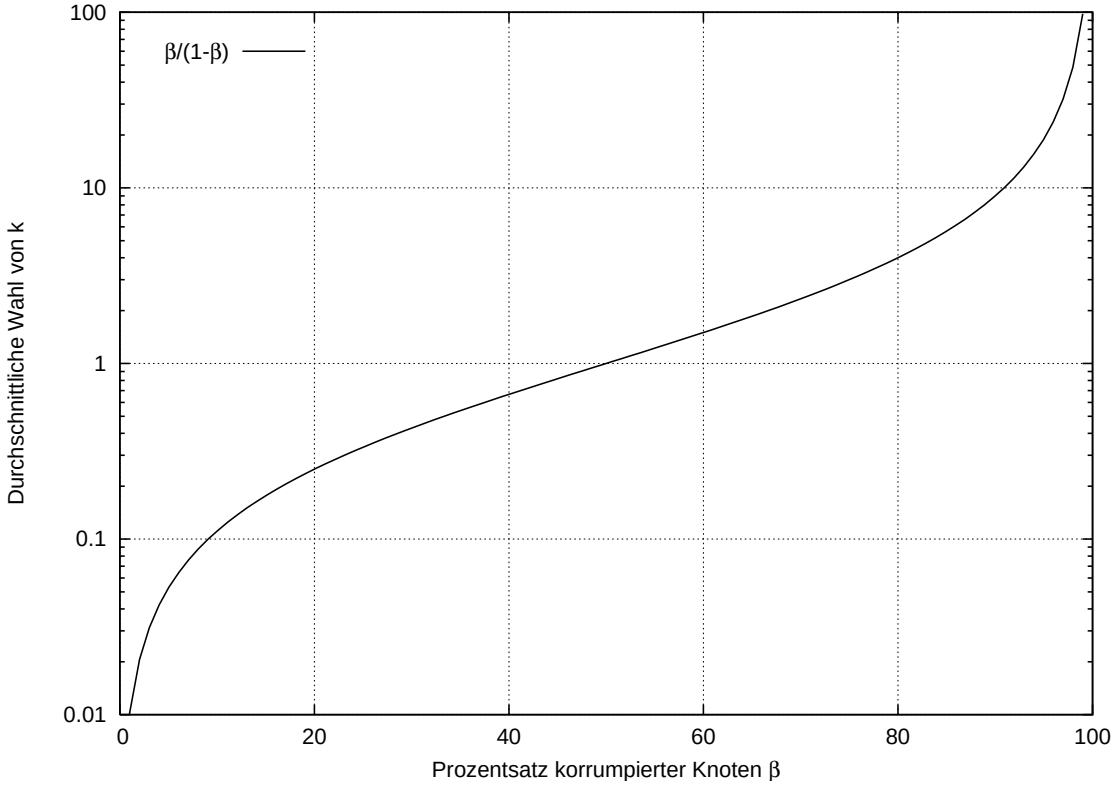


Abbildung 3.8 Theoretischer Verlauf von k in Abhängigkeit von β

Abbildung 3.8 zeigt den Verlauf zur Wahl von k in Abhängigkeit von β. Man kann erkennen, daß $k < 1$ für $\beta \leq 50\%$ gelten muß. Falls also der Angreifer nicht außerordentlich stark ist und *mehr* als die Hälfte der Knoten korrumpiert, reicht $k = 1$ aus, um bei den initialen Zufallsknoten zufällig mindestens einen legitimen Knoten zu wählen. Ersetzt man β durch die Systemparameter $\mathcal{B}$ und n, dann ergibt sich

$$\begin{aligned} \frac{\frac{\mathcal{B}}{n}}{1-\frac{\mathcal{B}}{n}} &\leq k \\ s'(n,\mathcal{B}) := \frac{\mathcal{B}}{n-\mathcal{B}} &\leq k. \end{aligned}$$

Der Benutzer wählt somit schließlich k als Maximum von $s(\delta, n, \mathcal{B})$ und $s'(n, \mathcal{B})$:

$$k \geq \max(\lceil s(\delta, n, \mathcal{B}) \rceil, \lceil s'(n, \mathcal{B}) \rceil).$$

Es ist allerdings auch noch zu bedenken, daß der Benutzer durch diese Wahl von k nur *im Durchschnitt* sicher Schlüssel austauscht: Die zum Bestimmen von k zum Einsatz kommenden Parameter δ und β stehen jeweils nur für Durchschnitte. In dem Fall, in dem sich die initialen Knoten e und d beide derart in einem Teilbaum befinden, so daß sie beide auf ihrer Suche nach einem Vorgänger mit Schlüssel für f auf $k' > k$ direkt hintereinanderliegende korrumpierte Knoten treffen, kann SKEY den Schlüssel nicht sicher austauschen. Der Angreifer wäre in diesem Fall in der Lage, den auszutauschenden Schlüssel mitzulesen oder zu ändern. Dieser Fall ist allerdings sehr unwahrscheinlich – die Evaluierung wird das genauer untersuchen.

3.3.6 Finden mehrerer initialer Zufallsknoten

Wie oben beschrieben benötigt ein neuer Knoten i zwei beziehungsweise $(k+1)$ Tickets für initiale Zufallsknoten, die sich bereits im Netz aufhalten. Der Einfachheit halber nehmen bisherige Arbeiten über Schlüsselaustauschprotokolle wie Eschenauer und Gligor [88] immer solche Knoten an die das Netzwerk niemals verlassen. Sobald sie ihm einmal beigetreten sind, bleiben Sie immer erreichbar und können daher für immer neue Knoten beim Schlüsselaustausch helfen. In derart idealisierten Szenarien funktioniert auch SKEY perfekt. Jedoch sind solche Szenarien mit statischen Netzen und „unsterblichen“ Knoten äußerst unrealistisch. In der Realität wird man damit rechnen müssen, daß Batterieresserven endlich sind, Knoten aufgrund technischer Defekte ausfallen, mutwillig zerstört werden usw.

Eine zufällige Auswahl von $k+1$ initialen Zufallsknoten aus den n Knoten im Netz zu finden, ist dementsprechend alles andere als einfach für den Benutzer oder das MD. Es ist dem Benutzer unklar, welche Knoten sich noch im Netz befinden oder das Netz bereits verlassen haben, weil sie, beispielsweise wegen aufgebrauchter Energiereserven, einfach funktionsunfähgig sind. Der Benutzer läuft daher Gefahr, Tickets für solche Knoten auszuhändigen, die i nicht mehr beim Schlüsselaustausch assistieren können.

Zufällige Auswahl von $2(k+1)$ Knoten

Für diesen kritischen Punkt existiert jedoch eine relativ einfache Lösung: Während des Paarens werden neuen Knoten auch neue IDs zugeteilt – der Reihenfolge nach. Ein neuer Knoten, der dem Netz beitritt erhält eine ID, welche höher ist als alle anderen IDs der bereits im Netz befindlichen Knoten. Auf diese Weise muß das MD nur die höchste ID, $\mathrm{MAXNODES}$, die Gesamtzahl der Knoten im Netz, speichern. Die gegenwärtige Konfiguration des Netzes bleibt immer noch unbekannt. Wenn k korrumpierte Knoten direkt hintereinander auf dem Austauschpfad liegend erwartet werden, sind $k+1$ Zufallsknoten initial notwendig. Während des Paarens eines neuen Knotens i mit der ID $i := \mathrm{MAXNODES} + 1$ wählt das MD $k+1$ Knoten aus, indem es $k+1$ Zufallszahlen zieht: $r_1, \ldots, r_{k+1} \in \{1, \ldots, \mathrm{MAXNODES}\}$. Die initialen Zufallsknoten für den Knoten i wären damit die Knoten mit den IDs $r_1, \ldots, r_{k+1}$. In dynamischen Netzen sind diese $k+1$ Knoten jedoch nur noch mit einer bestimmten Wahrscheinlichkeit funktionsfähig. Sobald einer von ihnen nicht mehr funktionsfähig sein sollte, scheitert der komplette Schlüsselaustausch.

Daher ist weiterhin sinnvoll, prinzipiell mehr als $k+1$ Knoten auszuwählen. Die Wahrscheinlichkeit, mindestens $k+1$ *funktionsfähige*, also sich noch im Netz befindende, aktive Knoten auszuwählen, steigt durch die Auswahl von $j > k+1$ Knoten. Angenommen

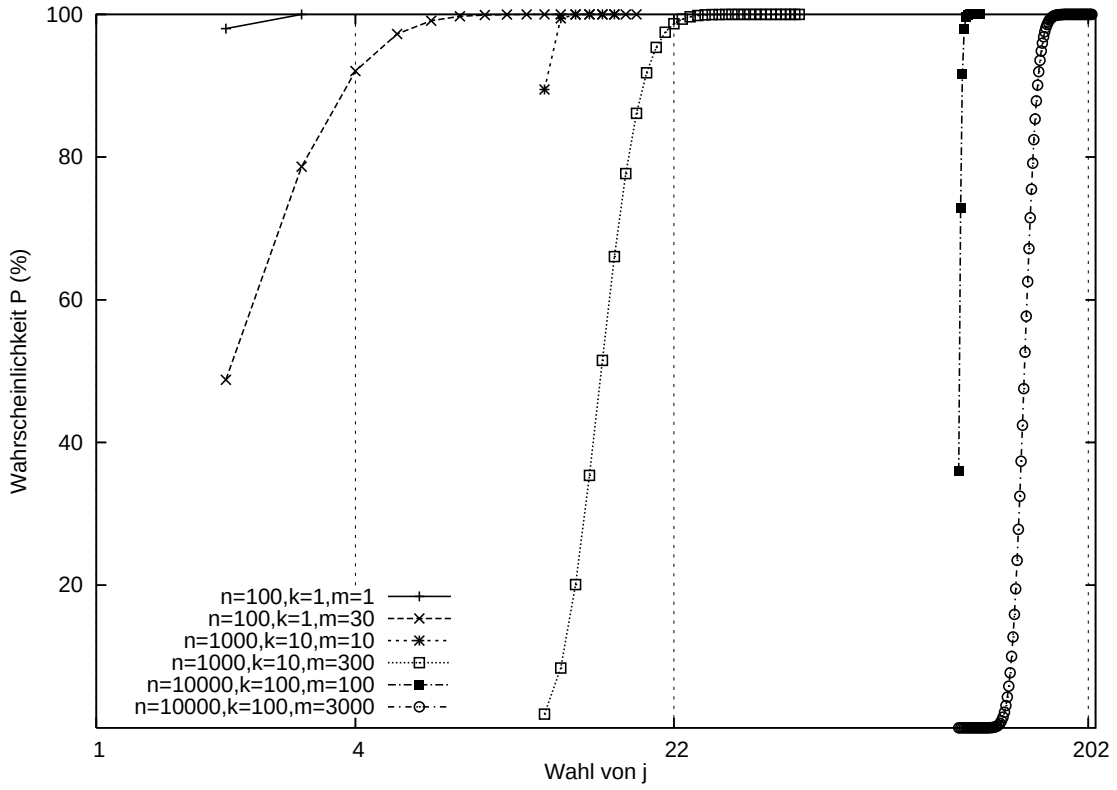


Abbildung 3.9 Wahrscheinlichkeiten für mindestens $k+1$ funktionsfähige Knoten bei Auswahl von $j > k$-vielen, Vertikalen bei $j = 2(k+1)$

sei eine Gesamtzahl von n Knoten, die jemals dem Sensornetz beigetreten sind. Zu dem Zeitpunkt, an dem Knoten i dem Netz beitreten möchte, haben bereits m von diesen n Knoten das Netz verlassen, beziehungsweise verfügen über keine Energie mehr und sind funktionsunfähig – demnach verbleiben $(n-m)$ funktionsfähige Knoten im Netz. Das Ziel ist nun, mit einer hohen Wahrscheinlichkeit mindestens $k+1$ lebende Knoten durch Auswahl von $j > k$ Knoten insgesamt zu finden.

Dieses Problem ist äquivalent zu folgendem bekannten Problem der Kombinatorik (mindestens $k+1$ richtige Kugeln beim 6 aus 49 Lotto): Gegeben ist eine Urne mit n Kugeln, m roten und $(n-m)$ grünen Kugeln. Beim zufälligen Ziehen von insgesamt $j \geq k+1$ Kugeln aus dieser Urne, wie hoch ist die Wahrscheinlichkeit mindestens $k+1$ grüne Kugeln und $(j-k-1)$ rote Kugeln zu ziehen? Diese Wahrscheinlichkeit P kann wie folgt berechnet werden:

$$P = \sum_{k'=k+1}^{j} \frac{\binom{n-m}{k'}\binom{m}{j-k'}}{\binom{n}{j}}$$

Die Funktion P steigt sehr schnell mit j, wie in Abbildung 3.9 ersichtlich. Die verschiedenen Kurven, welche hier dargestellt werden, zeigen drei verschiedene Szenarien mit der Gesamtzahl von Knoten $n = 100, 1000, 10000$. In den Szenarien gibt es weiterhin unterschiedlich starke Angreifer, die derart irgendeine Anzahl $\mathcal{B}$ von Knoten korrumpieren, daß k vom Benutzer gewählt werden soll zu $k = 1, 10$ und 100. Außerdem sind pro Szenario zwei verschiedene Anteile funktionsunfähiger Knoten m angenommen, jeweils 1% und 30% der Knoten sind funktionsunfähig und können einem neuen Knoten i beim

Schlüsselaustausch nicht mehr helfen. Die 1% und 30% repräsentieren einen geringen und hohen Anteil an Dynamik durch Knotenausfall im Netz. Die k kompromittierten Knoten gehören immer zum „lebenden“, noch aktiven Teil der $(n - m)$ Knoten. Die Y-Achse in Abbildung 3.9 zeigt die Wahrscheinlichkeit, mindestens $k + 1$ lebende Knoten zu finden, wenn aus dem Gesamtnetz von n Knoten, bei Ausfall von m Knoten, das MD j initiale Zufallsknoten auswählt. Die X-Achse ist logarithmisch skaliert.

Als Ergebnis kann man erkennen, daß P sehr hohe Werte von über 90% erreicht, sobald j sich $2(k+1)$ nähert. Für die drei Szenarien mit $k = 1, 10$ oder 100 korrumpierten Knoten wäre $2(k + 1)$ entsprechend 4, 22 und 202. An dieser Stelle ist es wichtig, festzuhalten, daß $2(k+1)$ eine obere Schranke darstellt – unabhängig von n: Wenn der Benutzer $k+1$ korrumpierte Knoten bei $n = 1000$ Knoten animmt, ist die Wahrscheinlichkeit $2(k + 1) = 4$ lebende Knoten zu finden in diesem Netz wesentlich höher als in einem Netz mit $n = 100$ Knoten.

Sobald k korrumpierte Knoten hintereinander auf dem Austauschpfad angenommen werden und Knotenausfall möglich ist, wählt das MD nicht nur $k + 1$ initiale Zufallsknoten, sondern $2(k + 1)$ Zufallsknoten aus dem Netzwerk aus. Im Pseudocode, siehe $\mathrm{pairNode}$, Zeilen 6ff, werden der Übersichtlichkeit halber immer $2(k + 1)$ Knoten ausgewählt. Der neue Sensor i wird schließlich von den $2(k+1)$ Knoten sukzessive versuchen, $k+1$ funktionierende zu erreichen. Sobald er $k+1$ funktionierende Knoten erreicht hat, kann er mit dem normalen SKEY-Protokollablauf fortfahren.

Der Ausdruck $2(k + 1)$ ist konstant und unabhängig von der Gesamtgröße n des Sensornetzes. Dieser Aspekt von SKEY skaliert daher im Hinblick auf die Gesamtanzahl der Knoten n sehr gut.

3.3.7 Finden der Vorgängerknoten

Bevor ein neuer Knoten i überhaupt mit all den Vorgängern $\mathbb{P} = \{f, b, a, ..\}$ auf seinem Aggregationspfad Schlüssel austauscht, muß er diese zunächsteinmal kennen. Er muß wissen, daß $\mathbb{P}$ seine zukünftigen Kommunikationspartner sein werden, mit diesen Knoten seines Aggregationsbaumes wird er später kommunizieren. Die Suche nach $\mathbb{P}$ spezifiziert die Funktion $\mathrm{get}\mathbb{P}()$ aus Algorithmus 9. Aus Sicherheitssicht handelt es sich hierbei wiederum um ein schwieriges Problem: Würde i einfach, zum Beispiel per Broadcast, in seiner Nachbarschaft nach seinen Vorgängern im Aggregationsbaum fragen, so könnte ein Angreifer diese Anfrage abfangen, modifizieren und i mit einer gefälschten Vorgängerliste $\mathbb{P}'$ antworten.

Kein spezifisches SKEY-Problem

Dieses Problem ist zunächst *kein* spezielles Problem des sicheren Schlüsselaustausches, und man könnte argumentieren, daß es sich hierbei um ein grundsätzliches Problem sicherer Aggregation handelt: Eine entsprechende sichere Software-Komponente müßte diese Informationen bereitstellen, beispielsweise i mittels *out-of-band* Mechanismen wie einem verteilten Dienstverzeichnis [28, 103] über $\mathbb{P}$ informieren. Eine andere, in vielen Szenarien wahrscheinlich eher unrealistische Möglichkeit wäre, daß der Benutzer die Aggregationsknoten eines neuen Sensors einfach kennt, also einem neuen Temperatursensor i die Knoten seines Aggregationspfades mitteilt.

Grundsätzlich ist festzuhalten, daß *jedes* Schlüsselaustauschverfahren, auch die Arbeiten aus dem Abschnitt über den Stand der Forschung, voraussetzen, daß Knoten jeweils wissen, mit wem sie zu kommunizieren haben: Im Beispiel sicherer Aggregation genau mit

1: **Declaration:** ID $\mathbb{P}_{\text{temp}}[]$, ID m, Key $K_{m,i}$
2: {i sucht $\mathbb{P}$ mit Hilfe der IRN:}
3: $\mathbb{P}_{\text{temp}}$:= new Array $[1 \ldots 2k+1]$
4: **for** $j := 1$ to $2k+1$ **do**
5: $m := \text{IRN}[j].\text{ID}$
6: $K_{m,i} := \text{IRN}[j].\text{Key}$
7: $i \rightarrow m : E_{K_{m,i}}(\text{findIP}(\text{ID}, \text{purpose}))$ {siehe Algorithmus 10}
8: $i \leftarrow m : E_{K_{m,i}}(\mathbb{P}')$
9: $\mathbb{P}_{\text{temp}}[j] := \mathbb{P}'$
10: **end for**
11: {Überprüfe, ob alle Antworten $\mathbb{P}'$ übereinstimmen:}
12: **if** $\mathbb{P}_{\text{temp}}[1] = \ldots = \mathbb{P}_{\text{temp}}[2k+1]$ **then**
13: $\mathbb{P} := \mathbb{P}_{\text{temp}}[1]$
14: **else**
15: **error** {siehe Algorithmus 5}
16: **end if**

Algorithmus 9 $i.\text{get}\mathbb{P}()$

den Knoten auf ihrem Aggregationspfad. Häufig wird, insbesondere bei den verwandten Arbeiten, auf diese Problematik erst gar nicht hingewiesen.

Technik von SKEY als Lösungsansatz

Andererseits kann sich i mit einer Technik analog zur bisher beschriebenen Technik von SKEY sicher über $\mathbb{P}$ informieren. Wenn alle Knoten auf Aggregationspfaden sich ihrer aktuellen Aufgabe im Aggregationsbaum bewußt sind, zum Beispiel : „Ich aggregiere gerade die Temperatursensoren in Raum 2.“ ,dann wissen sie auch, daß beispielsweise ein neuer Temperatursensor i zukünftig mit ihnen kommunizieren muß und daher gemeinsame Schlüssel brauchen wird. Nun kann man wieder wie bei SKEY initiale Zufallsknoten sowie die sichere Verteilung von Geheimnissen verwenden, um $\mathbb{P}$ zu lernen. Da i in der Lage ist, sich mit Hilfe von Tickets zumindest bei e und d sicher zu identifizieren, kann i diese beiden Knoten mit Hilfe der Funktion $\text{get}\mathbb{P}$, siehe $\text{get}\mathbb{P}()$, auch bitten, für sich Informationen über den Aggregationspfad herauszufinden – die Knoten e und d suchen i's Vorgänger. Knoten i gibt e und d dazu seine Dienstbeschreibung purpose. Dieses Verfahren funktioniert genau dann, wenn neue Knoten wissen, was ihre Aufgabe sein wird, purpose, und sich bereits im Netz befindende Knoten purpose eines neuen Knotens verstehen.

Knoten e fragt in der Funktion findIP, Algorithmus 10 schrittweise seine direkten Vorgänger im Aggregationsbaum, zunächst b, nach dem Aggregationspfad für einen neuen Temperatursensor i, siehe Zeilen 8ff. Hier soll der Begriff responsible andeuten, daß e seine Vorgänger fragt, ob sich ein Vorgängerknoten für eine bestimmte Dienstbeschreibung (purpose) als responsible betrachtet, das heißt, ein Aggregationsknoten für diese purpose ist. Knoten b kennt den Pfad bereits, da b selbst Teil davon ist. Daher antwortet b mit dem ihm bekannten Pfad $\mathbb{P} = \{f, b, a...\}$, den i benötigt. Schließlich fragt e weitere k Vorgänger von b nach dem Aggregierungspfad für einen neuen Temperatursensor, Zeilen 19ff. Nur wenn alle Antworten der k Vorgänger übereinstimmen, schickt e Pfad $\mathbb{P}$ zurück an i, Zeilen 27ff. Analog bittet i schließlich d und alle weiteren initialen Zufallsknoten, seinen Aggregierungspfad zu finden. Knoten d antwortet i dann schließlich

Eingabe: ID i, String purpose
1: **Declaration:** ID $\mathbb{P}_{\text{temp}}[]$
2: {IRN suchen nach Aggregationspfad $\mathbb{P}$ für neuen Knoten i:}
3: **if** $\text{responsible}(\text{purpose}) =$ **true then**
4: $m \rightarrow i : E_{K_{m,i}}(m.\mathbb{P}_i)$ {m gibt Pfad für i zurück}
5: **else**
6: $j := 0$
7: **repeat**
8: {Frage Vorgänger nach $\mathbb{P}_i$ für i:}
9: $j := j + 1$
10: $m \rightarrow \Pi[j] : (m, E_{K_{m,\Pi[j]}}(\text{responsible}, \text{purpose}))$
11: $m \leftarrow \Pi[j] : E_{K_{m,\Pi[j]}}(\mathbb{P}'_i)$
12: **until** $\mathbb{P}'_i \neq \varnothing$
13: **if** $j < |\Pi| - k$ **then**
14: {Frage direkt die Wurzel nach $\mathbb{P}_i$:}
15: $m \rightarrow \Pi[|\Pi|] : (m, E_{K_{m,\Pi[|\Pi|]}}(\text{responsible}, \text{purpose}))$
16: $m \leftarrow \Pi[|\Pi|] : E_{K_{m,\Pi[|\Pi|]}}(\mathbb{P}'_i)$
17: $m \rightarrow i : E_{K_{m,i}}(\mathbb{P}'_i)$
18: **else**
19: {Frage zusätzlich die k Vorgänger von $\Pi[j]$ nach $\mathbb{P}_i$ für i:}
20: $\mathbb{P}_{\text{temp}} :=$ new Array $[1 \ldots k + 1]$
21: $\mathbb{P}_{\text{temp}}[1] := \mathbb{P}'_i$
22: **for** $l := 1$ to k **do**
23: $m \rightarrow \Pi[j + l] : (m, E_{K_{m,\Pi[j+l]}}(\text{responsible}, \text{purpose}))$
24: $m \leftarrow \Pi[j + l] : E_{K_{m,\Pi[j+l]}}(\mathbb{P}')$
25: $\mathbb{P}_{\text{temp}}[l + 1] := \mathbb{P}'_i$
26: **end for**
27: {Überprüfe, ob alle $\mathbb{P}'_i$ übereinstimmen:}
28: **if** $\mathbb{P}_{\text{temp}}[1] = \ldots = \mathbb{P}_{\text{temp}}[k + 1]$ **then**
29: {Antworte an i:}
30: $m \rightarrow i : E_{K_{m,i}}(\mathbb{P}_{\text{temp}}[1])$
31: **else**
32: **error**
33: **end if**
34: **end if**
35: **end if**

Algorithmus 10 $m.\text{find}\mathbb{P}(i, \text{purpose})$

mit Pfad $\mathbb{P}'$. Falls alle Aggregierungspfade übereinstimmen, $\text{get}\mathbb{P}$, Zeilen 11ff, verfügt i selbst in Gegenwart von k korrumpierten Knoten über einen korrekten Aggregationspfad $\mathbb{P}$.

3.3.8 Dynamische Aggregation

Ein Sensorknoten i kann zu jedem beliebigen Zeitpunkt ausfallen. In solch einer Situation sind alle bei i gespeicherten Schlüssel verloren und langfristig gesehen unsicher. Im Gegensatz zu verwandten Arbeiten, siehe Abschnitt B.2.2, stellt dies jedoch für SKEY kein Problem dar: Die Kommunikation zwischen den Vorgängern von i, seinen Nachfolgern

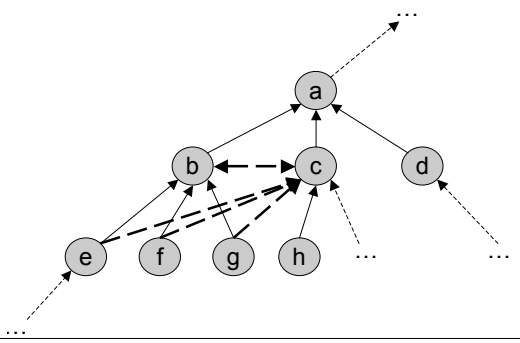


Abbildung 3.10 Knoten c übernimmt b's Aggregationsbeziehungen

und überhaupt allen anderen Knoten im Netz ist davon in keiner Weise betroffen, da diese eigene, paarweise verschiedene Schlüssel verwenden. Jeder Schlüssel zwischen zwei Knoten a und b ist „einzigartig“, er wird ausschließlich zum Absichern der Kommunikation zwischen a und b verwendet und nicht gleichzeitig zum Absichern der Kommunikation anderer Knoten. Deshalb müssen im Gegensatz zu auf Schlüssellisten-basierenden Protokollen bei Knotenausfall keine Schlüssel gestrichen und neue Schlüssel ausgetauscht werden. Die dafür nötige Energie kann eingespart werden, was selbst bei geringem Knotenausfall einen enormen Vorteil gegenüber beispielsweise *Random Key Pre-Distribution*-Protokollen darstellt – wie die Evaluierung noch zeigen wird.

Das einzige Problem, daß beim Ausfall eines Knotens im Sensornetz entstehen kann, liegt darin, daß seine spezielle Funktionalität nicht mehr erbracht wird. Dies impliziert allerdings keine neuen Sicherheitsprobleme, die nicht auch ohne Schlüsselaustausch existieren würden.

3.3.8.1 Auswirkungen durch Änderungen der Aggregation

Andererseits ist es durchaus vorstellbar, daß Knoten *freiwillig* für längere Zeitspannen das Netz verlassen, um ausgedehntere Schlafphasen beispielsweise zur Batterieschonung durchzuführen. Auch die Konfiguration der Aggregation selbst ändert sich dynamisch: Eine entsprechende Aggregations-„Middleware“ ändert von Zeit zu Zeit die einzelnen Aggregationsbeziehungen der Knoten untereinander. Dadurch ändern sich einzelne Teilbäume oder gar der komplette Aggregationsbaum. Dieser Wechsel der Aggregationsbeziehungen erfordert eine teilweise Neuverteilung von Schlüsseln, wie im folgenden Beispiel ersichtlich.

Man nehme an, daß Knoten b aus Abbildung 3.10 seine Aggregationsbeziehungen aufgeben muß und durch Knoten c ersetzt wird. Knoten c soll demnach zusätzlich zu seiner bisherigen Aggregation noch die Aggregation von Knoten b übernehmen. Dafür benötigt c allerdings mit den Knoten e, f, g sowie allen anderen Knoten, bei denen b auf dem Aggregationspfad liegt, gemeinsame Schlüssel. Die Idee hier besteht nun darin, daß b seine gemeinsamen Schlüssel mit $e, f, g, \ldots$ an c schickt, sodaß c sicher mit $e, f, g, \ldots$ kommunizieren kann. Es lohnt sich für c nicht, mit $e, f, g, \ldots$ neue paarweise geheime Schlüssel auszutauschen, da für symmetrische Kryptographie Perfect Forward Secrecy nicht erreicht erfüllt ist – zu dieser Problematik siehe nochmals Abschnitt 3.2.7.

Da b und c jedoch nicht auf dem gleichen Aggregationspfad liegen, besitzen sie keinen gemeinsamen Schlüssel und können nicht sicher Daten austauschen. Der Knoten b benutzt deshalb zum Austausch eines neuen gemeinsamen Schlüssels mit c in der Gegenwart von k korrumpierten Knoten die Funktion $\mathrm{forwardShare}$ aus Algorithmus 7: Knoten b führt

die Funktion $b.\mathrm{forwardShare}(b, K_{b,c}, c)$ aus. Wie oben beschrieben generiert b einen zufälligen Schlüssel $K_{b,c}$, teilt ihn in $k+1$ Teile $K_{b,c} = K^1 \oplus \ldots \oplus K^{k+1}$ auf und versendet die Teile sicher an diejenigen $k+1$ Vorgänger auf seinem Aggregationspfad, die behaupten, einen gemeinsamen Schlüssel mit c zu besitzen. Diese Vorgänger schicken sämtliche Schlüsselteile sicher an c. Schließlich kann c den gemeinsamen Schlüssel $K_{b,c}$ berechnen. Diese Vorgänger findet er wie zuvor bei SKEY mit Hilfe von $\mathrm{forwardShare}$, siehe Algorithmus 7. Wie bei SKEY ist auch hier sichergestellt, daß mindestens ein gemeinsamen Vorgänger zwischen b und c existiert – im ungünstigsten Fall die Senke.

Der für diesen speziellen Schlüsselaustausch zwischen b und c notwendige Energieaufwand, Nachrichten und kryptographische Operationen, ist so hoch wie der Aufwand für das Austauschen eines Schlüssels zwischen einem neuen Knoten und einem anderen, sich bereits im Aggregationsbaum befindenden Knoten: Der Energieaufwand steigt mit der Höhe des Aggregationsbaums, das heißt logarithmisch. Er wächst in Abhängigkeit der Gesamtanzahl Knoten n mit $O(\log n)$ pro Knoten. Genaueres zum Energieverbrauch für einen Schlüsselaustausch untersucht Abschnitt 3.4.

3.3.8.2 Dynamisches Anpassen von k

Es sei zum Abschluß noch erwähnt, daß SKEY prinzipiell ein sich zur Laufzeit des Sensornetzes variierendes k unterstützt. Der Benutzer könnte beispielsweise entscheiden, daß ein vorab bestimmtes k nicht ausreicht oder ein a prio gewähltes k zu hoch für den aktuellen Angreifer gewählt worden ist und deshalb durch ein $k' \neq k$ ausgetauscht werden soll.

In diesem Fall müssen alle Knoten im Sensornetz sicher über das neue k informiert werden. Diese ginge zum Beispiel mit Hilfe des folgenden, einfachen Protokolls von der Senke aus: Eine Senke w als Wurzel des Aggregationsbaums schickt jedem Sensor i im Netz stellvertretend für den Benutzer auf sichere Weise das neue k', $w \rightarrow i : E_{K_{w,i}}(k')$. Daraufhin würden alle Knoten im Sensornetz SKEY mit dem neuen k' verwenden. Auf diesen Aspekt des dynamischen Anpassens von k während des Sensornetz-Betriebs ist allerdings in dieser Arbeit nicht weiter eingegangen worden.

3.4 Evaluierung

Dieser Abschnitt untersucht die Leistung des vorgestellten Protokolls SKEY im Hinblick auf den dadurch implizierten Speicherverbrauch sowie die Anzahl der deswegen zu versendenden Nachrichten und notwendigen kryptographischen Operationen und damit zusammen den erforderlichen Energieverbrauch. Diese anfallenden *Kosten* werden im folgenden zunächst theoretisch diskutiert und anschließend in Abschnitt 3.4.3 durch verschiedene Simulationen überprüft.

3.4.1 Speicherverbrauch

Grundsätzlich hängt die Leistung von SKEY von dem Aggregationsbaum ab, anhand dessen Schlüssel ausgetauscht werden.

Wie in Abschnitt 3.3.3.2 beschrieben skaliert die Anzahl der Schlüssel, die jeder Knoten in seinem Speicher vorhalten muß, in Abhängigkeit von der Höhe des Aggregationsbaums: logarithmisch mit $O(\log n)$. Bei Höhe $h \approx \log_\delta(1+(\delta-1)n)-1$ eines Aggregationsbaumes, müssen Knoten höchstens h viele Schlüssel speichern. Bei einer Bitbreite von b Bit pro Schlüssel entspricht das $h \cdot \frac{b}{8}$ Bytes Speicherbedarf.

Diese Eigenschaft von SKEY ist deutlich besser als der schneller steigende und höhere Speicherverbrauch verwandter Arbeiten wie beispielsweise Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [88, 89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251].

3.4.2 Energieverbrauch

Zum Austausch von Schlüsseln eines neuen Knotens muß in den beiden Funktionen findIP und $\mathrm{forwardShare}$ im schlechtesten Fall der komplette Aggregationsbaum der Höhe $h \in O(\log n)$ zunächst von unten nach oben und danach wieder zurück nach unten „traversiert“ werden.

Das bedeutet, daß im schlechtesten Fall

1. zunächst $k+1$ Shares an die IRN verschickt werden ($\mathrm{introduceToIRN}$) – das sind $k+1$ Nachrichten.
2. alle $(k+1)$ IRN die Aufforderung zu getIP bekommen – das sind $k+1$ Nachrichten.
3. alle $(k+1)$ IRN findIP ausführen, im schlechtesten Fall
   - $(k+1)\cdot(h-k-1)$ Nachrichten auf der Suche nach gemeinsamen Vorgängern.
   - $(k+1)\cdot k$ Nachrichten zu den weiteren k Vorgängern der gemeinsamen Vorgänger.
   - insgesamt $(k+1)\cdot(h-1)$ Nachrichten als Antworten an die IRN, entweder wenn Knoten keine gemeinsamen Vorgänger sind oder die IP_i's der gemeinsamen Vorgänger.
   - $(k+1)$ Nachrichten mit Inhalt IP_i als Antwort von den IRN an i.
4. alle $(k+1)$ IRN von i Shares mit $\mathrm{exchangeKeys}$ erhalten – das sind $k+1$ Nachrichten.
5. alle $(k+1)$ IRN $\mathrm{forwardShare}$ ausführen, das sind im schlechtesten Fall
   - $(k+1)\cdot(h-k-1)$ Nachrichten auf der Suche nach gemeinsamen Vorgängern.
   - $(k+1)\cdot k$ weitere Nachrichten zu den k Vorgängern der gemeinsamen Vorgänger. Diese Nachrichten enthalten Shares.
   - $(k+1)\cdot(h-k-1)$ Nachrichten als Antworten von Vorgängern, die keine gemeinsamen Vorgänger mit f sind.
6. alle Knoten, die über ein Shares verfügen, schicken diese an f. Das sind im schlechtesten Fall $(k+1)\cdot(k+1)$ Nachrichten.
7. 2 Nachrichten Challenge-Response zwischen i und f.

Siehe hierzu die Zeilen 7 bis 12 in Algorithmus 10 sowie die Zeilen 8 bis 14 in Algorithmus 7.

Insgesamt sind damit im schlechtesten Fall $k+3+4\cdot k\cdot h+4\cdot h$ Nachrichten notwendig.

Hinzukommt der Energieverbrauch durch das Ver- und Entschlüsseln *jeder* einzelnen Nachricht – es sind genausoviele Ver- und Entschlüsselungen notwendig wie Nachrichten verschickt werden.

Der Energieverbrauch hängt demnach linear vom SKEY-Parameter k ab: k bestimmt die Anzahl der initialen Zufallsknoten sowie die Anzahl der Vorgängerknoten auf dem Austauschpfad. Da k allerdings *nicht* von n, sondern nur von den Möglichkeiten des Angreifers abhängt, siehe Abschnitt 2.4, S. 24, fällt es mit steigendem n. Damit skaliert der Energieverbrauch von SKEY mit $O(\log n)$.

3.4.3 Simulation

Zur Überprüfung der Leistungsfähigkeit SKEYs sind im Rahmen dieser Arbeit einige vergleichende Simulationen zwischen SKEY und der Arbeit von Eschenauer und Gligor aus Eschenauer und Gligor [88] durchgeführt worden [111]. Es sei jedoch darauf hingewiesen, daß es sich hierbei *keinesfalls* um einen echten Vergleich der beiden Arbeiten handeln soll – dieser wäre für beide Protokolle unfair. Die vergleichende Darstellung hat die folgende Aufgabe: Sie soll die Leistung von SKEY bewerten. Die Arbeit Eschenauer und Gligor [88] soll nur die Leistung von SKEY relativieren – zur besseren Einschätzung und Bewertung des Verfahrens SKEY. Die Darstellung zeigt, welche Vorteile ein an *aggregierenden Datentransport* angepaßter Schlüsselaustausch bezüglich Speicherkosten, Energieverbrauch usw. mit sich bringt.

Aus folgenden Gründen ist ein direkter Vergleich für beide Protokolle unfair:

- SKEY leistet mehr als Eschenauer und Gligor [88]: Das Protokoll kann mit dynamischen Szenarien umgehen, Knotenausfall sowie Wechsel in der Aggregation kompensieren. Das Angreifermodell aus Eschenauer und Gligor [88] betrachtet wesentlich schwächere Angreifer, die keine Knoten korrumpieren können. SKEY hingegen kann in Gegenwart von bis zu k korrumpierten Knoten noch sicher Schlüssel austauschen.
- Auf der anderen Seite leistet SKEY allerdings auch weniger als Eschenauer und Gligor [88]: SKEY ist ausschließlich für den Einsatz in aggregierenden Sensornetzen entworfen, während Eschenauer und Gligor [88] in beliebigen Netzen Schlüssel verteilt. In solchen nicht-aggregierenden Szenarien, in denen Schlüsselaustausch zwischen beliebigen Knoten im Netz zu leisten ist, wird SKEY zwangsläufig keinerlei Vorteil gegebenüber Eschenauer und Gligor [88] besitzen, sondern voraussichtlich sogar leistungsmäßige Nachteile. Der Entwurf des Protokolls will aber auch nicht den Anspruch erheben, in solchen Netzen effizient zu funktionieren.

Die durchgeführten Simulationen lassen sich grob in drei Kategorien einteilen, Details zu notwendigen Simulationsparametern folgen dann im Anschluß:

1. Kategorie: „Statische Simulationen“
   Die erste Kategorie soll mit wechselnden Netzparametern (n, δ, k) den Speicherverbrauch, die Anzahl der Nachrichten beziehungsweise die Gesamtenergie messen, die für den Schlüsselaustausch eines Sensornetzes bestehend aus insgesamt n Knoten notwendig sind. Konkret beginnt die Simulation des Sensornetzes dabei mit einem einzelnen Knoten, der Datensenke, und fügt dem Netz sukzessive $n-1$ neue

Knoten hinzu: Ein neuer Sensorknoten beginnt sofort, anhand von Algorithmus 1 Schlüssel auszutauschen. Die Vorschrift, nach der neue Knoten Schlüssel austauschen müssen, wird dabei durch einen zufälligen erzeugten Aggregationsbaum wie folgt vorgegeben:

Um Schlüssel zwischen Knoten auszutauschen, wird vor Beginn der Simulation zunächst ein zufälliger Aggregationsbaum aus insgesamt n Knoten generiert. Hierbei stellt sich unmittelbar die Frage, welchen Verzweigungs- oder Aggregationsgrad δ die einzelnen Knoten im Baum besitzen – im allgemeinen sicherlich abhängig vom Szenario. In den Simulationen sind normalverteilte Aggregationsgrade wie in Kapitel 2 angenommen. Insgesamt sind Simulationen mit verschiedenen erwarteten Aggregationsgraden durchgeführt worden, beispielsweise $\delta = 2, 3, 4, 5$ und 6 – das sind für viele Szenarien typische Grade von Aggregationen.

Durch den vorab erstellen Aggregationsbaum stehen die Aggregationsbeziehungen zwischen Knoten, die Notwendigkeit, zwischen Knoten Schlüssel auszutauschen, fest. In das anfangs nur aus der Datensenke/Wurzel bestehende Sensornetz werden in den Simulationen sukzessive neue Sensorknoten als neue „Blätter“ hinzugefügt – und zwar so, wie es der zufällig erzeugte Aggregationsbaum vorgibt. Jeder neue Sensor wird dazu über die Funktion $\mathrm{pairNode}$ gepaart und tauscht danach mittels der SKEY-Algorithmen neue Schlüssel aus. Dabei wird sowohl der Energieverbrauch durch die zu verschickenden Nachrichten und kryptographische Operationen gemessen als auch der Speicherverbrauch durch zu speichernde Schlüssel.

Mit den „statischen Simulationen“ werden genau diejenigen Kosten, Speicher und Energie, ermittelt die allein für den sicheren Aufbau eines Sensornetzes, den Austausch der notwendigen Schlüssel der einzelnen Knoten bezahlt werden müssen – ohne Betrachtung von Dynamik in Form von Knotenausfall.

2. Kategorie: „Dynamische Simulationen“
   In vielen Fällen sind Sensornetze dynamisch: Knoten verlassen das Netz, neue Knoten kommen hinzu, und die Aggregationsbeziehungen ändern sich. Simuliert wurde ein Szenario, bei dem Knoten an zufälligen Zeitpunkten das Netz verlassen. Der Energieaufwand für die anschließende Reorganisation des Netzes wurde gemessen. Bei Schlüssellisten-basierten Protokollen bedeutet Knotenausfall einiges an Mehraufwand: Die im Netz verbleibenden Knoten müssen ihre Key-Ringe von dem des ausgefallenen Knotens bereinigen. Wenn dabei unter Umständen Schlüssel bestehender Kommunikationsassoziationen betroffen sind, müssen entsprechend neue Schlüssel ausgetauscht werden. Außerdem wird gezeigt, daß dies einen enormen Energieaufwand bedeutet.

   Beim Start der Simulationen war ein Sensornetz bestehend aus n Knoten mit bereits ausgetauschten Schlüsseln gegeben, in dem zufällig und gleichverteilt im Netz 1% der Knoten ausfielen. Gemessen wurde der Energieaufwand, zum *Reorganisieren* des Netzes, das heißt zum notwendigen Austauschen neuer Schlüssel.

3. Kategorie: „Simulationen von Angreifern“
   Abschnitt B.2 zeigt, daß Schlüsselaustauschverfahren in Gegenwart korrumpierter Knoten schnell versagen können: Der Anteil an *Kommunikationsassoziationen* beziehungsweise Schlüsseln zwischen Knoten, die durch Korrumpieren in den Besitz des Angreifers gelangen, *kann* überproportional mit dem Anteil korrumpierter Knoten zunehmen. Eine (Kommunikations-)*Assoziation* zwischen zwei Knoten a

und b stellt dabei möglichen Datenaustausch zwischen a und b dar, der mit Hilfe eines Schlüssels gesichert wird. Solch eine Assoziation gilt als *gebrochen*, falls der Angreifer den Schlüssel dieser Assoziation kennt. Die Unterscheidung in Assoziationen und Schlüssel macht deshalb Sinn, weil zum Beispiel bei *Random Key Pre-Distribution*-Protokollen häufig mehrere Assoziationen mit ein und demselben Schlüssel gesichert sind. Kennt der Angreifer einen Schlüssel, kann er damit gleichzeitig mehrere Assoziationen abhören oder manipulieren. Bei SKEY gilt dies allerdings nicht: Hier wird eine Assoziation jeweils durch einen eigenen Schlüssel gesichert. Da allerdings die korrumpierten Knoten zufällig im Netz verteilt liegen, kann es vorkommen, daß zufällig $k' > k$ Angreifer auf dem Schlüsselaustauschpfad liegen. Dann ist auch bei SKEY der auszutauschende Schlüssel dem Angreifer bekannt.

Aus diesem Grund wird die Sicherheit von Assoziationen mit SKEY simulativ untersucht. Bei variierender Anzahl korrumpierter Knoten soll der sich daraus ergebene Prozentsatz gebrochener Assoziationen beim Austausch neuer Schlüsseln gemessen werden. Dabei gilt es zu beachten, daß alleine durch die Korrumpierung eines Knotens a der Angreifer in den Besitz aller Schlüssel von a kommt und somit alle Assoziationen dieses Knotens automatisch als gebrochen gelten. Dies ist grundsätzlich unabhängig vom Schlüsselaustauschverfahren: Alleine durch das Korrumpieren von Knoten werden bereits Assoziationen gebrochen.

Die Simulation untersucht daher, ob sich der Prozentsatz gebrochener Assoziationen nach dem Austauschen neuer Schlüssel mit Hilfe der Schlüsselaustauschprotokolle von dem unterscheidet, der sich alleine durch das Korrumpieren von Knoten automatisch ergibt. Weicht der Anteil der gemessenen Assoziationen (nach oben) ab, zeigt dies ein Schwäche des Schlüsselaustauschprotokolls auf.

Konkret werden Sensornetze simuliert, in denen sich bereits $n/2$ „legitime“, nicht korrumpierte Knoten befinden und miteinander Schlüssel ausgetauscht haben. Dann korrumpiert ein Angreifer zufällig insgesamt $\mathcal{B}$ viele von den $n/2$ Knoten und „bricht“ so automatisch einen bestimmten Anteil an Assoziationen zwischen Knoten. Anschließend werden dem Netz weitere $n/2$ Knoten hinzugefügt, die anhand des SKEY-Protokolls Schlüssel austauschen müssen. Der Angreifer versucht dabei, mit Hilfe der korrumpierten Knoten noch weitere Assoziationen zu brechen. Genauer: Durch die zufällige Verteilung der korrumpierten Knoten können sich auch zufällig mehr als k auf einem solchen Pfad befinden. Falls sich also auf einem Schlüsselaustauschpfad hintereinander $k' > k$ korrumpierte Knoten befinden, gelangt der Angreifer in den Besitz des auszutauschenden Schlüssels und bricht somit die auf diesem Schlüssel basierende Assoziation. In mehreren Simulationen, mit variierenden n, $\mathcal{B}$ und k, wird gemessen, wieviele Assoziationen der Angreifer insgesamt brechen kann.

3.4.3.1 Simulationsumgebung

Die Protokolle SKEY und Eschenauer und Gligor [88], letzteres stellvertretend für die Klasse der *Random Key Pre-Distribution*-Protokolle wie Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251], sind für den diskreten Ereignissimulator GloMoSim [221] implementiert worden. Das auf der C-ähnlichen Beschreibungssprache Parsec [222] basierende GloMoSim stellt eine umfangreiche Umgebung zur Simulation von Netzen aller Art bereit und implementiert dafür bereits eine

Vielzahl verschiedenster Protokolle, beispielsweise zu den Schichten 1 bis 4 des ISO/OSI-Referenzmodells [260]. Durch die Verfügbarkeit des Quellcodes ist eine Erweiterung um eigene Protokolle und (Netzwerk-)Anwendungen problemlos möglich.

Vor der Vorstellung der Simulationsergebnisse, folgt zunächst eine Erläuterung der für die Simulationen vorab notwendigen Entscheidungen bzgl. der *Simulationsparameter*.

Simulationsparameter

Simuliert werden sollte der Schlüsselaustausch zwischen allen Knoten, die jeweils auf den gleichen Aggregationspfaden eines Aggregationsbaumes liegen. Schließlich sind als entscheidende Kosten der Speicherverbrauch für Schlüssel, der durch kryptographische Operationen (Ver- und Entschlüsseln) aufzuwendende Energieverbrauch sowie der Energieverbrauch durch den Versand von Nachrichten pro Knoten berechnet worden.

Als Grundlage der Simulation ist GloMoSim soweit wie möglich auf die weit verbreitete MICA2-Plattform [66, 224] unter TinyOS [225] parametrisiert worden. Aufgrund der enormen Popularität und Verbreitung der MICA2/TinyOS-Plattform ermöglicht diese Parametrisierung praxisnahe Aussagen und erlaubt prinzipiell Vergleiche mit anderen Arbeiten. Im folgenden werden die wichtigsten Simulationsparameter aufgezählt und in Tabelle 3.2 zusammengefaßt. Weitere Details zu den Simulationen finden sich in Huttel [111].

Das Nachrichtenformat entspricht dem von TinyOS auf MAC-Schicht. Die Nachrichtengröße beträgt insgesamt 56 Bytes, wovon 29 Bytes Nutzdaten darstellen [69]. Der Energieverbrauch zum Versenden einer solchen Nachricht bei 38400 Bit/s Datenrate [69] beträgt bei 16 mA Energieverbrauch durch die Funkschnittstelle [69] und 5 mA durch den Mikrocontroller [11, 151] insgesamt $245\ \mu$As Energie, zum Beispiel hergeleitet in Blaß et al. [30]: Zum Übertragen von 56 Byte benötigt die MICA2-Funkschnittstelle bei 38400 Bit/s ungefähr $\frac{56\,\mathrm{Byte}}{38400\,\mathrm{Bit/s}} = 11,67$ ms. In dieser Zeit verbrauchen Mikrocontroller und Funkschnittstelle ungefähr $11,67$ ms$\cdot(16+5)$ mA$=245\ \mu$As [30]. Diese Werte sind in verschiedenen Literaturstellen stark unterschiedlich angegeben – siehe dazu die Energiediskussionen in Anhang A, S. 173. Als symmetrische Chiffre kommt in TinyOS das Verfahren RC5 [187] mit 64 Bit (=8 Byte) Schlüssel- und Blocklänge zum Einsatz. Das Ver- sowie Entschlüsseln eines 64 Bit Datenblocks dauert auf der MICA2-Plattform jeweils $0,26$ ms [123] und kostet bei 5 mA Energieverbrauch des Mikrocontrollers [11, 151] damit $0,26$ ms$\cdot 5$ mA$=1,3\ \mu$As Energie. Als Medienzugriffsprotokoll dient in der GloMoSim-Simulation das IEEE 802.11 CSMA/CA Verfahren, gesendet wird auf einer Frequenz von 433 MHz [69]. (Weitere Untersuchungen in Bezug auf die Simulation der 802.11 MAC-Schicht finden sich in Anhang C.)

Zum Schicht-3 Routing von Nachrichten im Sensornetz kommt *Precalc* aus Baumgart [15] zum Einsatz. Bei Precalc werden unmittelbar vor Simulationsbeginn vom Simulator die optimalen Routen zwischen Knoten berechnet und in statischen Routing-Tabellen gespeichert. Die Wahl von Precalc als Routing-Protokoll hat mehrere Gründe: TinyOS selbst spezifiziert kein Routing-Protokoll, und bisher existiert noch kein allgemeines Routing-Verfahren für Sensornetze. Zudem hat Baumgart [15] gezeigt, daß Verfahren aus der Ad-Hoc-Netzwerk-Forschung wie zum Beispiel AODV [171] oder OLSR [63] für sehr große Sensornetze völlig ungeeignet sind. Damit zum einen die Simulationsergebnisse nicht unnötig verfälscht werden und zum anderen ausschließlich die Anzahl Nachrichten und damit der Energieverbrauch gemessen wird, den das Schlüsselaustauschprotokoll verursacht, erscheint die Wahl des neutralen Precalc als sinnvoll. Um weiterhin die gemessene

Parameter	Wert
Medienzugriff	IEEE 802.11 [112]
Nachrichtenlänge gesamt	56 Bytes [69]
davon Nutzdaten	29 Bytes [69]
Datenrate	38400 Bit/s [69]
Energieverbrauch Funk	16 mA [69]
Energieverbrauch CPU	5 mA [11, 151]
Energie pro Nachricht	245 μAs [30]
Chiffre	RC5 [187] 8 Byte Schlüssel-/Blocklänge [123]
Energie pro Ver-/Entschlüsselung	1,3 μAs [11, 123, 151]
Routing-Protokoll	Precalc [15]

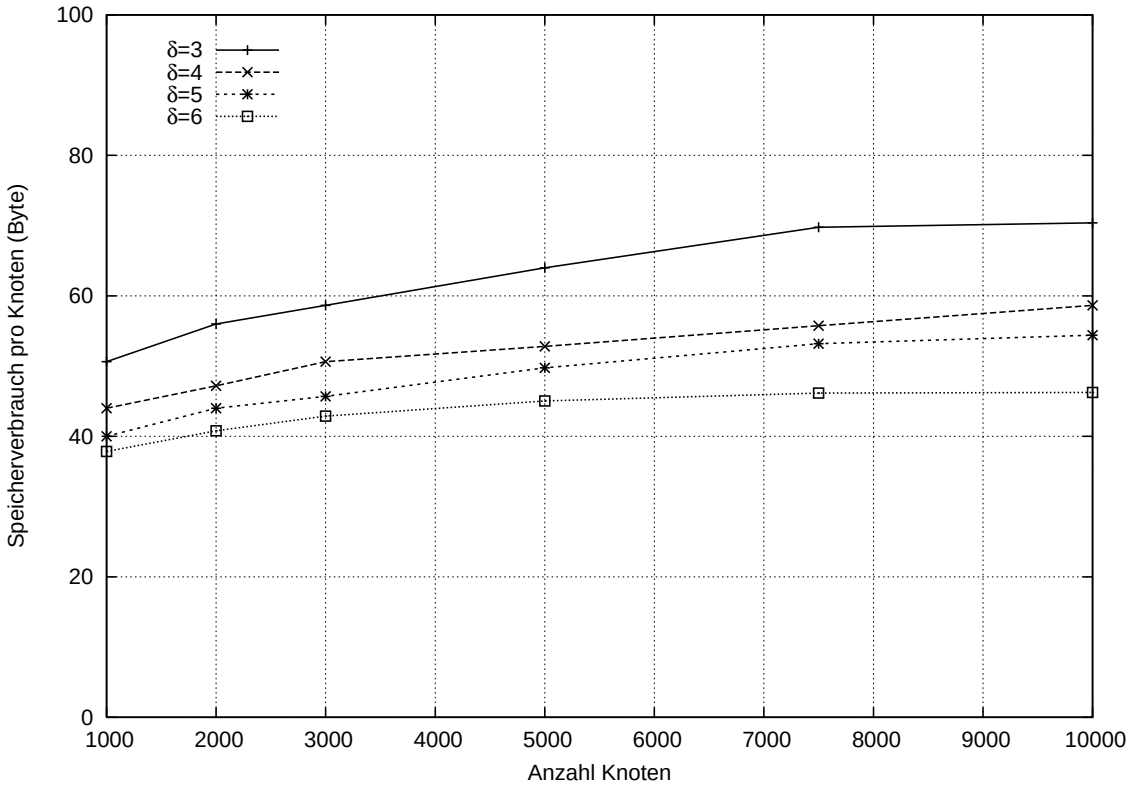
Tabelle 3.2 Übersicht: Parameter für Simulationen

Anzahl von Nachrichten nicht zu verfälschen, zum Beispiel durch die Wiederholung von Übertragungen, die durch Hintergrundverkehr bedingt sind, existiert im Sensornetz während des Schlüsselaustausch kein weiterer Netzwerkverkehr.

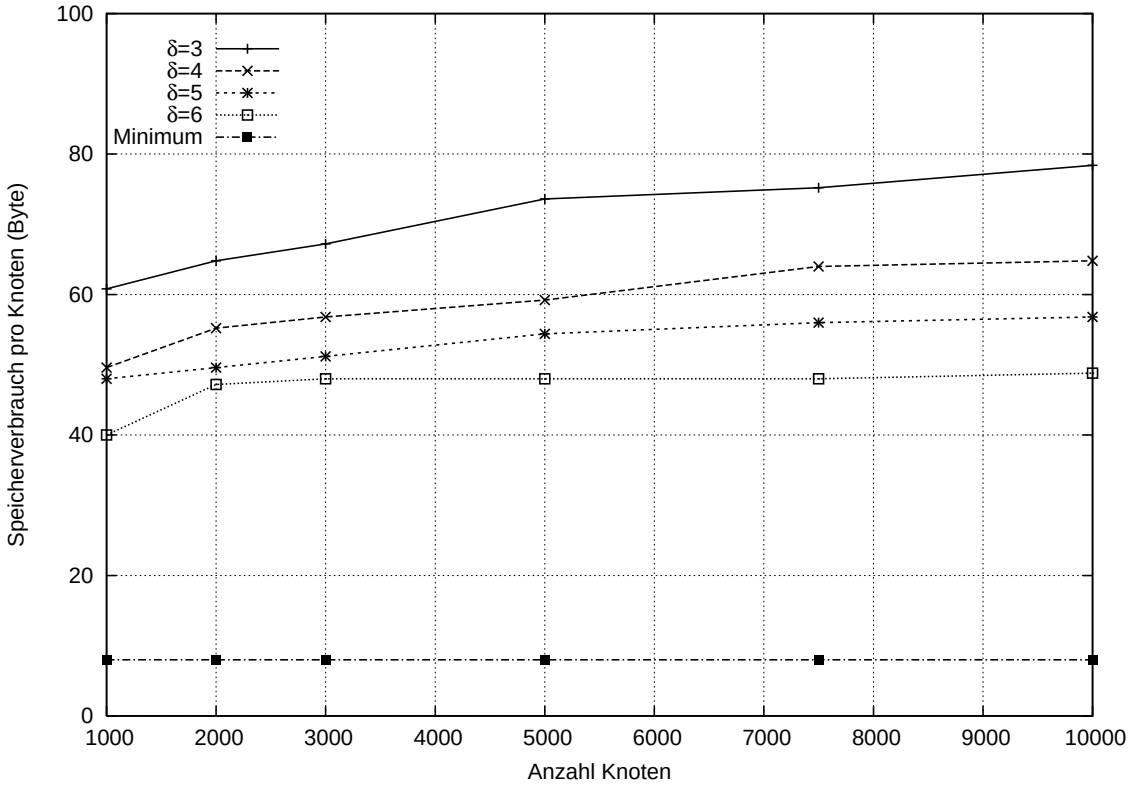
Die Gesamtanzahl von Knoten im Netz variiert in den Simulationen zwischen $n = 1000$ bis $n = 10000$ Knoten. Damit kann untersucht werden, wie gut sich die Verfahren beim Einsatz in großen Sensornetzen verhalten beziehungsweise skalieren.

3.4.3.2 Simulationsergebnisse – Statische Simulationen

Zunächst wird die Menge an Hauptspeicher in Byte gemessen, die das Protokoll SKEY durch das Speichern von Kommunikationsschlüsseln benötigt. Abbildung 3.11(a) zeigt den *durchschnittlich* benötigten Speicher pro Knoten in einem Sensornetz mit n Knoten. Die X-Achse variiert die Gesamtanzahl aller Sensorknoten n im Netz bis zu 10.000, der Aggregationsgrad variiert zwischen $\delta = 3$ und $\delta = 6$. Die Abbildung zeigt, daß je größer der Aggregationsgrad, desto geringer der Speicherverbrauch. Dies liegt darin begründet, daß bei höherem durchschnittlichen Aggregationsgrad die durchschnittliche Höhe des Aggregationsbaumes geringer ist. Geringere Baumhöhe bedeutet auch kürzere Aggregationspfade und damit schließlich weniger Schlüssel, die pro Knoten im Hauptspeicher abgelegt werden müssen. Analog dazu zeigt Abbildung 3.11(b) den *maximalen* und *minimalen* Speicherverbrauch pro Knoten. Wie eingangs erwähnt, müssen bei SKEY die Knoten jeweils nur die Kommunikationsschlüssel mit Vorgängern und Vor-Vorgängern im Baum speichern. Das führt dazu, daß genau ein Knoten, die Senke, *einen* Schlüssel (RC5 Schlüssel entspricht 8 Byte) speichert und in einem Aggregationsbaum der Höhe h die Blätter maximal $h + 1$ viele Schlüssel. Auch der maximale Speicherverbrauch steigt logarithmisch mit der Höhe des Baumes und ist abhängig vom Aggregationsgrad. Die Maxima-Kurven verlaufen deutlich unruhiger (relative Standardabweichung $< 30\%$) verglichen mit denen der durchschnittlichen Verbräuche (relative Standardabweichung $< 11\%$). Dies liegt daran, daß die Maxima, auch gemittelt über jeweils 10 Durchläufe des Simulators, ausschließlich von der Gesamthöhe des Aggregationsbaumes abhängen: Die Höhe bei verschiedenen zufälligen Aggregationsbäumen springt allerdings sehr stark zwischen *wenigen*, *kleinen* und vor allem *diskreten* Werten. Die Höhe h eines Aggregationsbaum mit $\delta = 3$ und $n = 1000$ Knoten springt beispielsweise zwischen den Werten 5, 6 und 7. Daraus resultiert die sehr hohe relative Standardabweichung.



(a) Mittlerer Speicherverbrauch, verschiedene Aggregationsgrade



(b) Maxima und Minimum an Speicherverbrauch, verschiedene Aggregationsgrade

Abbildung 3.11 SKEY-Speicherverbrauch, zur rel. Standardabweichung siehe Text

Weiterhin bemerkenswert ist die Tatsache, daß die Kurven für den mittleren Speicherverbrauch nur wenig unter dem des maximalen Speicherverbrauchs verlaufen. Dies liegt daran, daß die Blätter in den Aggregationsbäumen, die Sensorknoten mit dem höchsten Speicherverbrauch, schon einen relativ hohen Anteil am Netz ausmachen. Im Sensornetz mit Aggregationsbaum der Höhe h und durchschnittlichem Grad δ existieren δ^h Blätter, was einem Anteil von

$$\frac{\delta^h}{n} = \frac{\delta^h}{\frac{\delta^{h+1}-1}{\delta-1}} = \frac{\delta^h(\delta-1)}{\delta^{h+1}-1} =: f(\delta, h)$$

entspricht. Setzt man

$$g(\delta) := \lim_{h\to\infty} f(\delta, h),$$

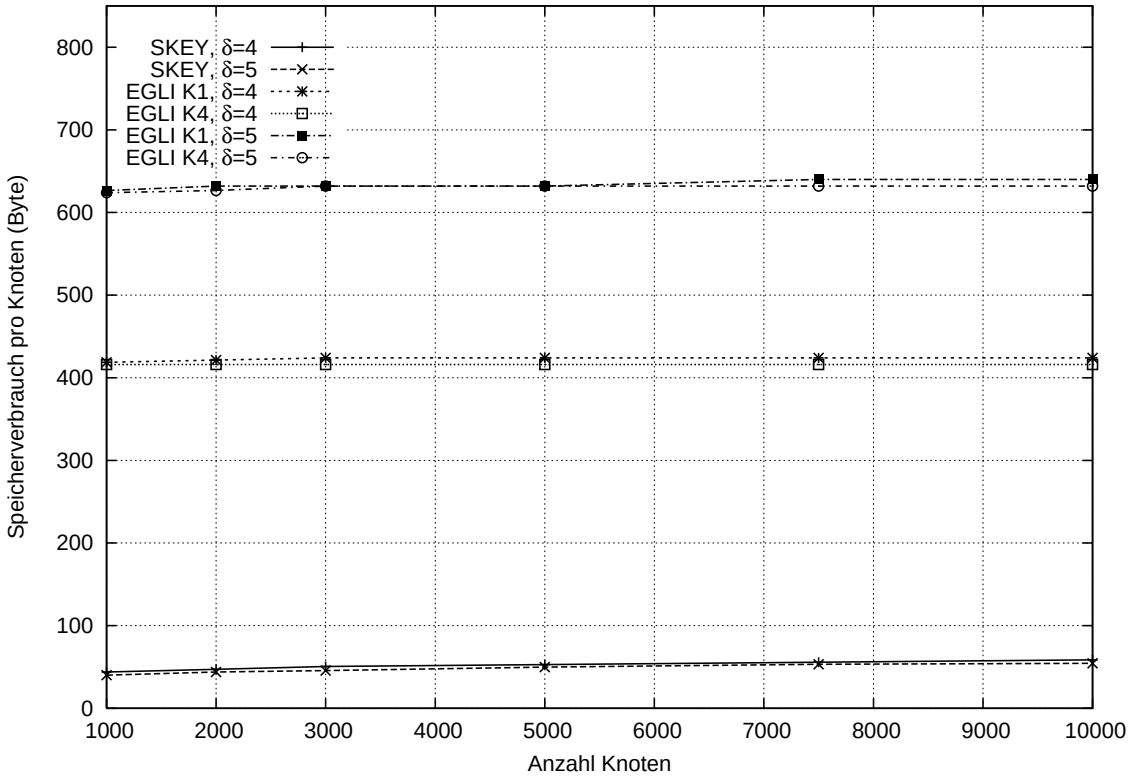
so ergibt sich z.B. für den Anteil der Blätter im Baum mit $\delta = 3$ ein Wert von $g(3) = \frac{2}{3}$, bei $\delta = 4$ ist $g(4) = \frac{3}{4}$, $g(5) = \frac{4}{5}$ usw. Kurz: In einem Netz vom Grad von beispielsweise $\delta = 3$ benötigen $\approx 66\%$ aller Knoten, die Blätter, bereits den maximalen Speicher aus Abbildung 3.11(b).

Ein maximaler Speicherverbrauch von unter 80 Byte pro Knoten, das entspricht 10 RC5 Schlüsseln, in einem Netz mit insgesamt 10.000 Knoten und einem kleinen Aggregationsgrad von $\delta = 3$ dürfte allerdings auch bei einem Gesamtspeicher von nur 4 KByte der MICA2 Plattform noch für die meisten Szenarien als durchaus akzeptabel gelten. Insbesondere wird dies deutlich, wenn man diese Ergebnisse mit dem Speicherverbrauch von *Random Key Pre-Distribution*-Protokollen in Relation setzt.

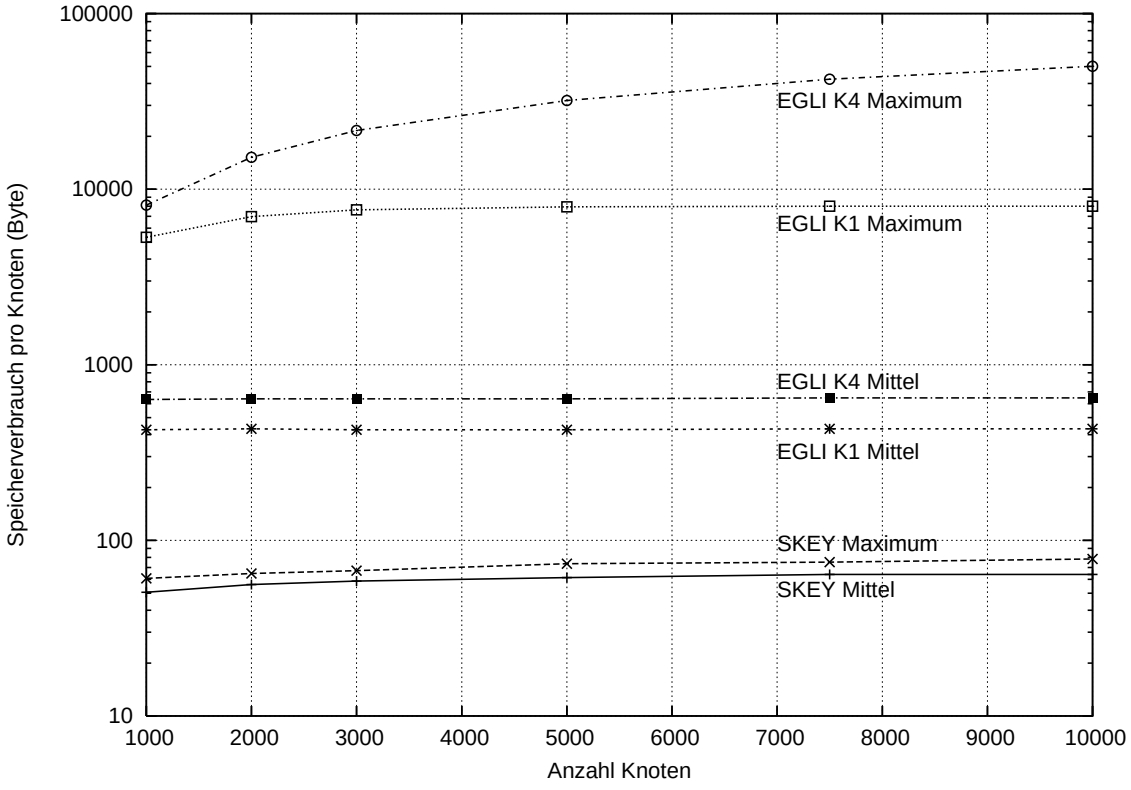
Die Abbildung 3.12 setzt die Meßwerte zum Speicherverbrauch von SKEY in Relation zum Protokoll aus Eschenauer und Gligor [88], im folgenden *EGLI* genannt. Für die *gleichen*, zufällig erzeugten Aggregationsbäume ist sowohl der Schlüsselaustausch mittels SKEY als auch der mittels EGLI simuliert worden. Dabei zeigt Abbildung 3.12(a) beispielhaft für die Aggregationsgrade $\delta = 4$ und $\delta = 5$ den mittleren Speicherbedarf. Die beiden Konfigurationen K1 und K4, siehe auch Tabelle B.1 aus Anhang B.2.1, S. 179, sollen dabei eine Art untere beziehungsweise obere Schranke für möglichen Speicherverbrauch darstellen. Es wird klar, daß der Speicherverbrauch von EGLI deutlich über dem von SKEY liegt – man beachte auch die im Vergleich zu den vorherigen Bildern geänderte Y-Achsen-Skalierung. Diese Tatsache wird in Abbildung 3.12(b) weiter veranschaulicht. Neben dem mittleren Speicherverbrauch verschiedener Konfigurationen ist hier außerdem der maximale Speicherverbrauch skizziert. Um beide Protokolle zum Schlüsselaustausch nebeneinander darzustellen, muß die Y-Achse logarithmisch skaliert werden. Wie bereits in Abschnitt B.2.3 festgestellt, läßt sich das in Eschenauer und Gligor [88] vorgeschlagene Protokoll in vielen Fällen praktisch gar nicht einsetzen. Der hohe Speicherverbrauch von EGLI liegt daran, daß Knoten „oben“ im Aggregationsbaum Schlüssel mit *allen* Nachfolgern im Baum speichern müssen. Außerdem gibt es Knoten, die laut Aggregationsbaum zwar miteinander Schlüssel austauschen müssen, jedoch über keinen gemeinsamen Schlüssel auf ihren Key-Ringen verfügen. Dementsprechend vergrößern sie ihren Key-Ring und das kostet wiederum Speicher.

Die Simulationen zeigen, daß SKEY in aggregierenden Sensornetzen deutlich weniger Speicher verbraucht als *Random Key Pre-Distribution*-Protokolle.

Abbildung 3.13 zeigt den durchschnittlichen Energieverbrauch pro Knoten von SKEY in Abhängigkeit verschiedener Protokollparameter und Knotenzahlen in mAh. Der Gesamtenergieverbrauch setzt sich zusammen aus der Energie, die für den Versand von TinyOS-Nachrichten notwendig ist sowie aus der Energie zum Chiffrieren und Dechiffrieren von

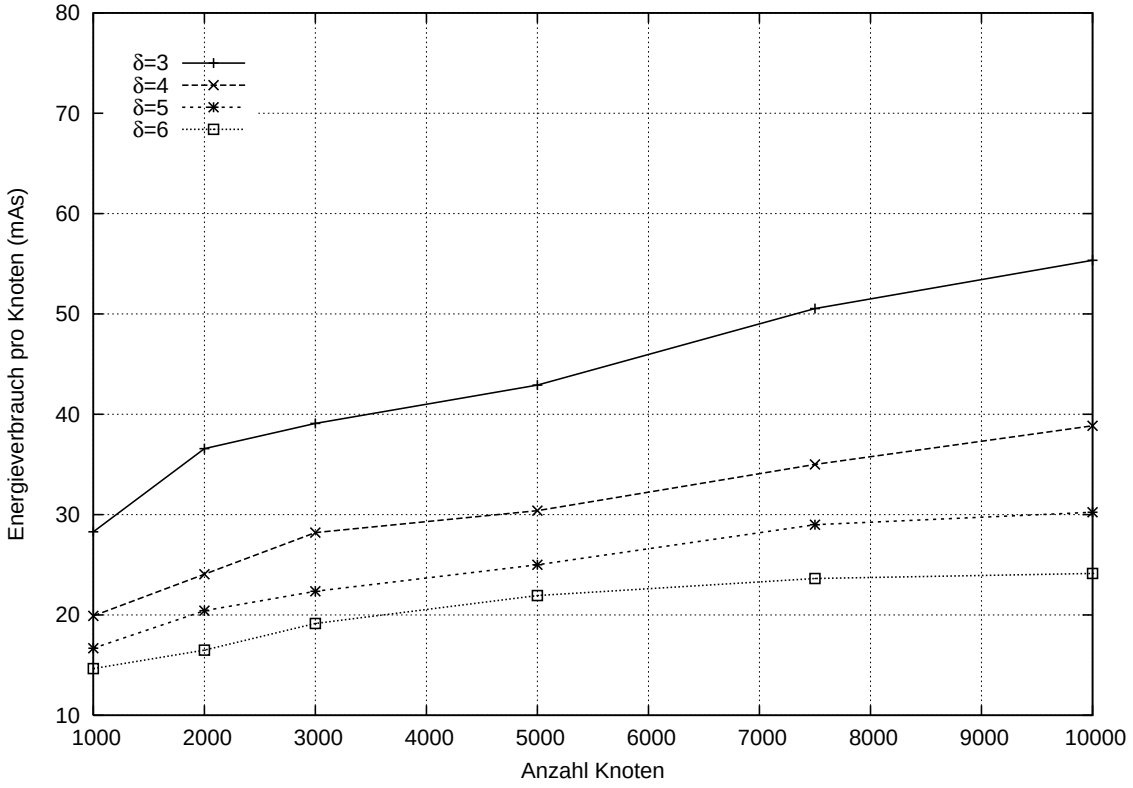


(a) Mittlerer Speicherverbrauch für verschiedene Aggregationsgrade

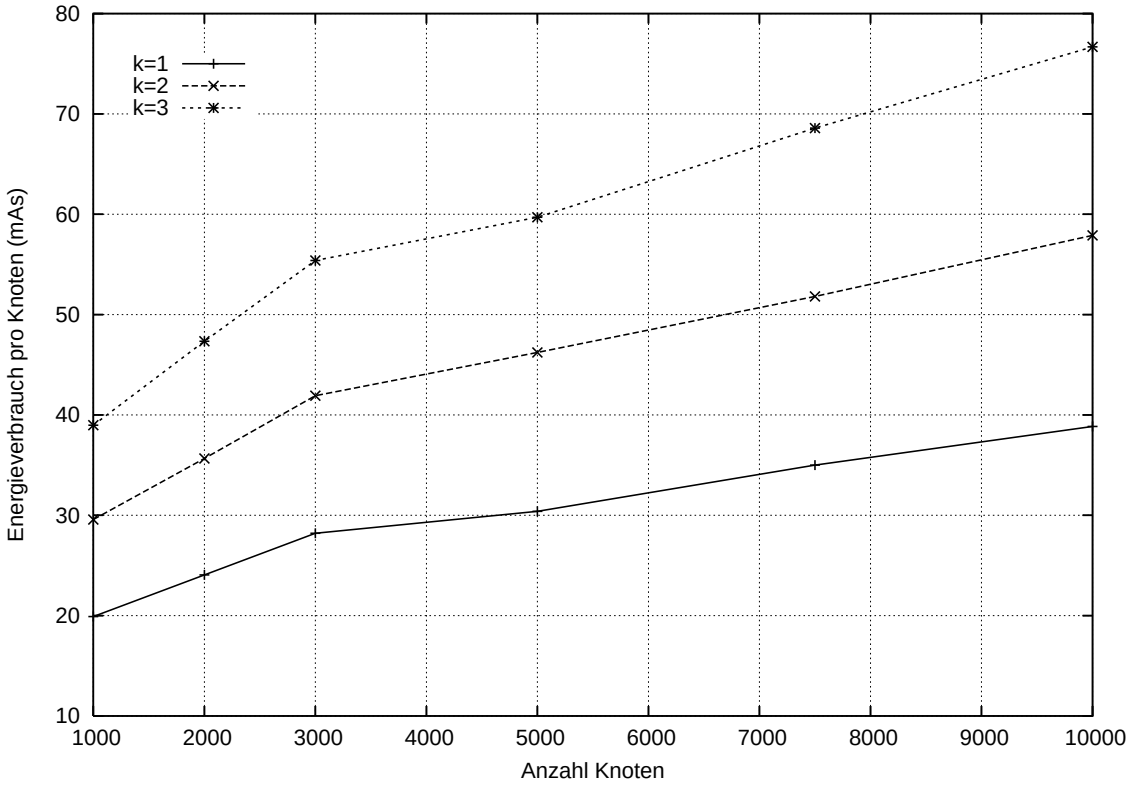


(b) Mittlerer und maximaler Speicherverbrauch, $\delta = 3$

Abbildung 3.12 Speicherverbrauch, Vergleich zwischen SKEY und [88](=EGLI)



(a) Vergleich verschiedener Aggregationsgrade bei $k = 1$



(b) Vergleich $k = 1, 2, 3$ bei Aggregationsgrad $\delta = 4$

Abbildung 3.13 Energieverbrauch SKEY mit unterschiedlichen Parametern

Daten. Abbildung 3.13(a) variiert den durchschnittlichen Aggregationsgrad δ bei festem $k = 1$. Ähnlich wie beim Speicherverbrauch gilt auch hier, daß je größer der Aggregationsgrad des Aggregationsbaums, desto niedriger der Energieverbrauch. Dies entspricht dem erwarteten Verhalten: Je höher der Aggregationsgrad, desto kürzer die Aggregationspfade, desto kürzer die Austauschpfade, desto weniger Nachrichten und Verschlüsselungen sind notwendig.

Da aus den Abbildungen ersichtlich ist, wie die Kurven mit unterschiedlichen $\delta = 3, 4, 5, 6$ zueinander verlaufen, wird in den folgenden Simulationen der Übersichtlichkeit halber δ festgehalten und nur noch k variiert.

Abbildung 3.13(b) unterscheidet verschiedene k bei festem $\delta = 4$. Die Kurven liegen im Graph beinahe parallel verschoben übereinander. Auch dies entspricht der Erwartung: Anstatt bei $k = 1$ jeweils 2 Schlüsselteile im Baum aufwärts zu versenden, sind bei $k = 2$ insgesamt jeweils 3 Teile notwendig. Dies erklärt den Faktor von rund $\frac{3}{2}$ zwischen $k = 1$ und $k = 2$ im Graphen. Analog läßt sich $k = 3$ erklären.

Schließlich setzt Abbildung 3.14 SKEYs Energieverbrauch in Relation zu EGLI [88]. Deutlich zu erkennen ist dabei der lineare Energieaufwand von EGLI, der darin begründet ist, daß für den Austausch zwischen zwei benachbarten Knoten im Aggregationsbaum häufig erst ein sogenannter „Schlüsselpfad“, siehe Anhang B.1, S. 176, über andere Knoten gefunden werden muß. Dies resultiert in einer linear steigenden Anzahl von Nachrichten und kryptographischen Operationen.

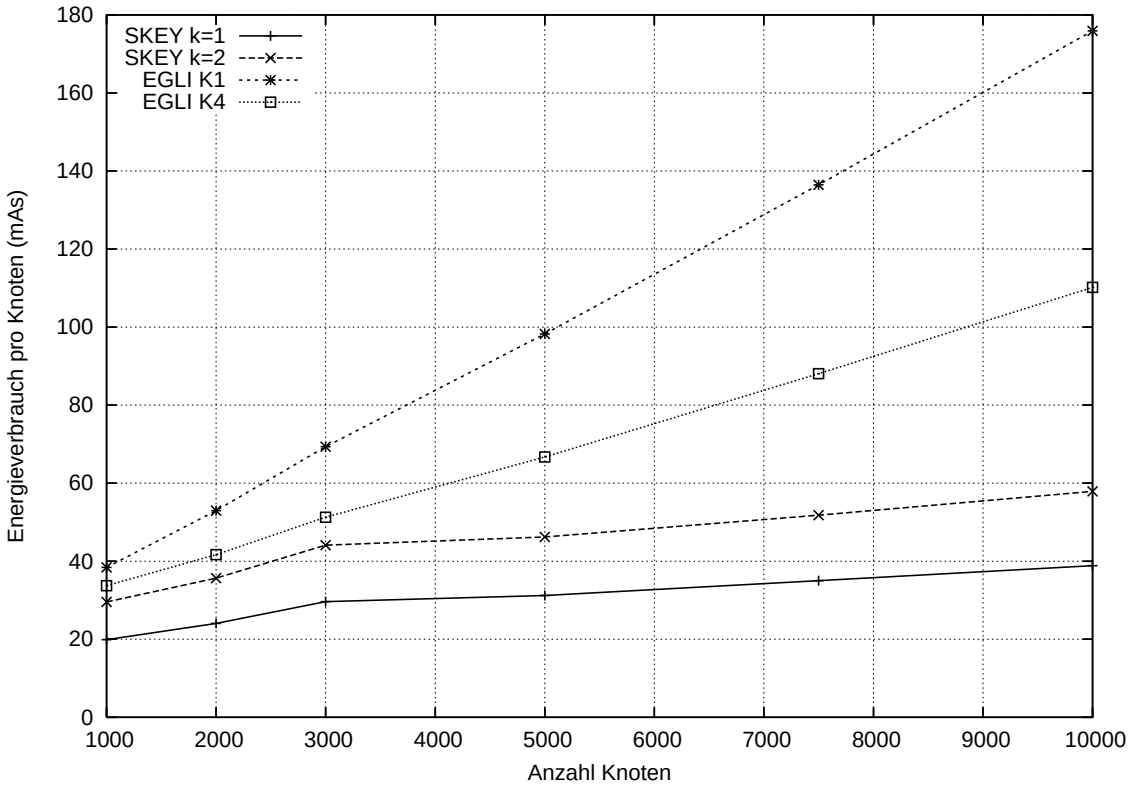
Auch für den Energieverbrauch gilt, daß SKEY in aggregierenden Sensornetzen wesentlich effizienter arbeitet als EGLI.

3.4.3.3 Simulationsergebnisse – Dynamische Simulationen

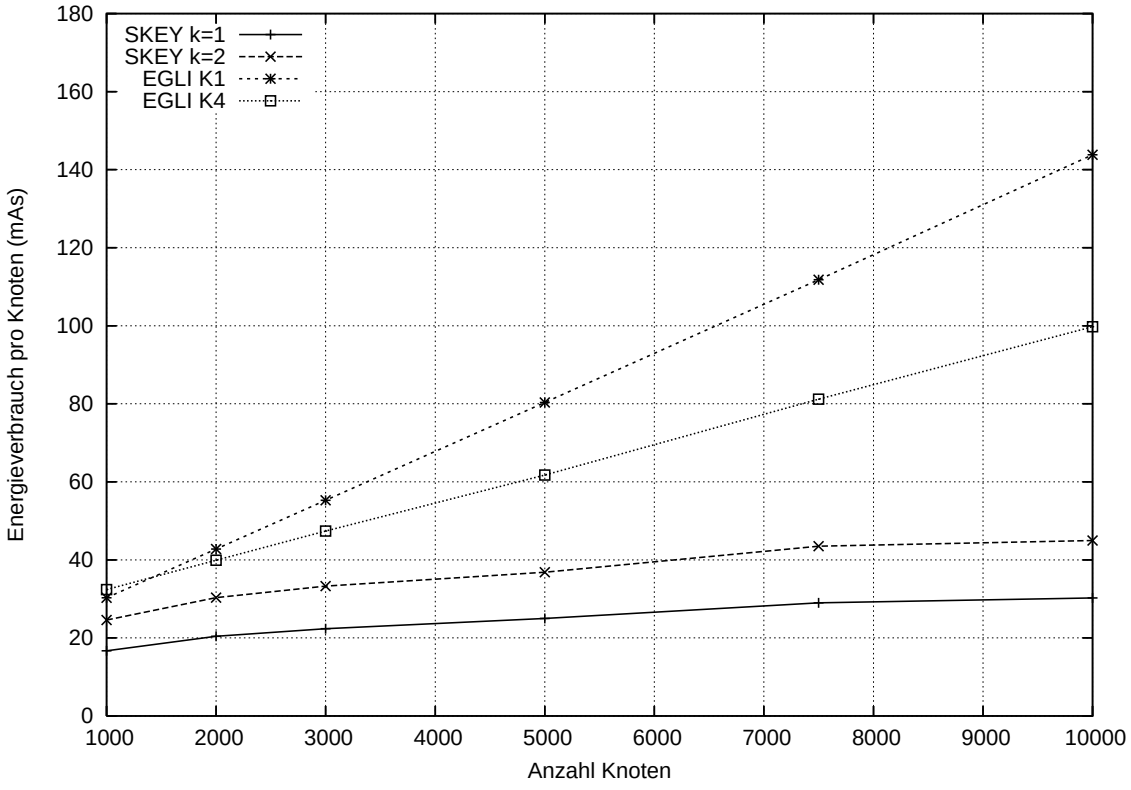
Im Falle von spontanem Knotausfall ist bei SKEY keinerlei zusätzlicher Energieverbrauch in Form von Nachrichten oder kryptographische Operationen notwendig. Dies ist ein enormer Vorteil gegenüber Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [88, 89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251], wie Abbildung 3.15 zeigt. Selbst bei der geringen Dynamik der Simulationen von nur 1% Knotenausfall steigt bei Eschenauer und Gligor [88] der Energieaufwand für das Neuverhandeln von Schlüsseln und damit verbunden die Anzahl der Nachrichten und der kryptographischen Operationen stark an. Dies liegt daran, daß die im Netz verbleibenden Knoten ihre Key-Ringe von sämtlichen Schlüsseln der Key-Ringe der ausgefallenen Knoten bereinigen müssen. Dadurch müssen mit hoher Wahrscheinlichkeit Schlüssel von bereits bestehenden Kommunikationsassoziationen neu ausgetauscht werden.

Die Simulationen, die mit Konfiguration $K1$, $K4$ durchgeführt worden sind, enden bereits bei $n = 5000$ Knoten, da bei höheren Knotenzahlen selbst der angenommene Knotenausfall von nur 1% zum völligen Zusammenbruch der Sicherheit im Netz führt: Es existieren dann keine sicheren Schlüssel mehr. Siehe hierzu allerdings die ausführliche Untersuchungen in Abschnitt B.2.2.

Die SKEY-Kurve in Abbildung 3.15 verläuft bei $Y = 0$, da keinerlei zusätzlich Nachrichten oder Verschlüsselungen bei Knotenausfall anfallen. Es ist kein zusätzlicher Schlüsselaustausch notwendig.



(a) Aggregationsgrad $\delta = 4$



(b) Aggregationsgrad $\delta = 5$

Abbildung 3.14 Energieverbrauch, Vergleich zwischen SKEY und [88]

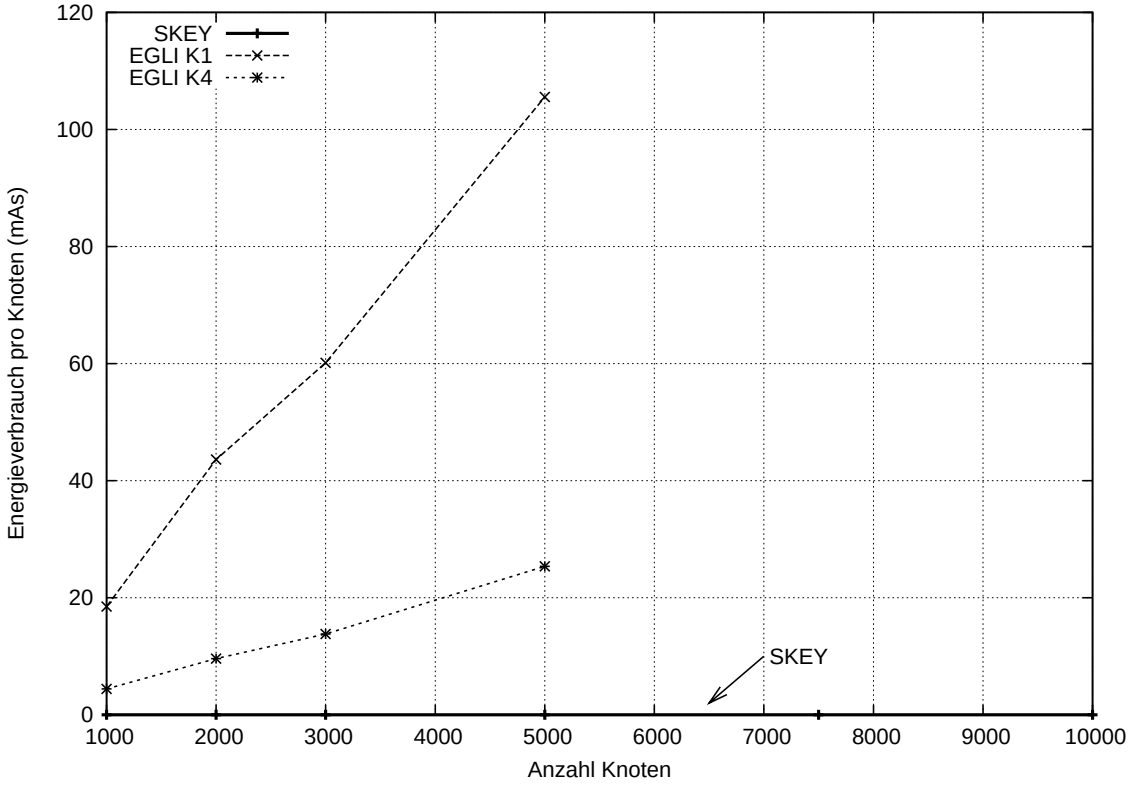
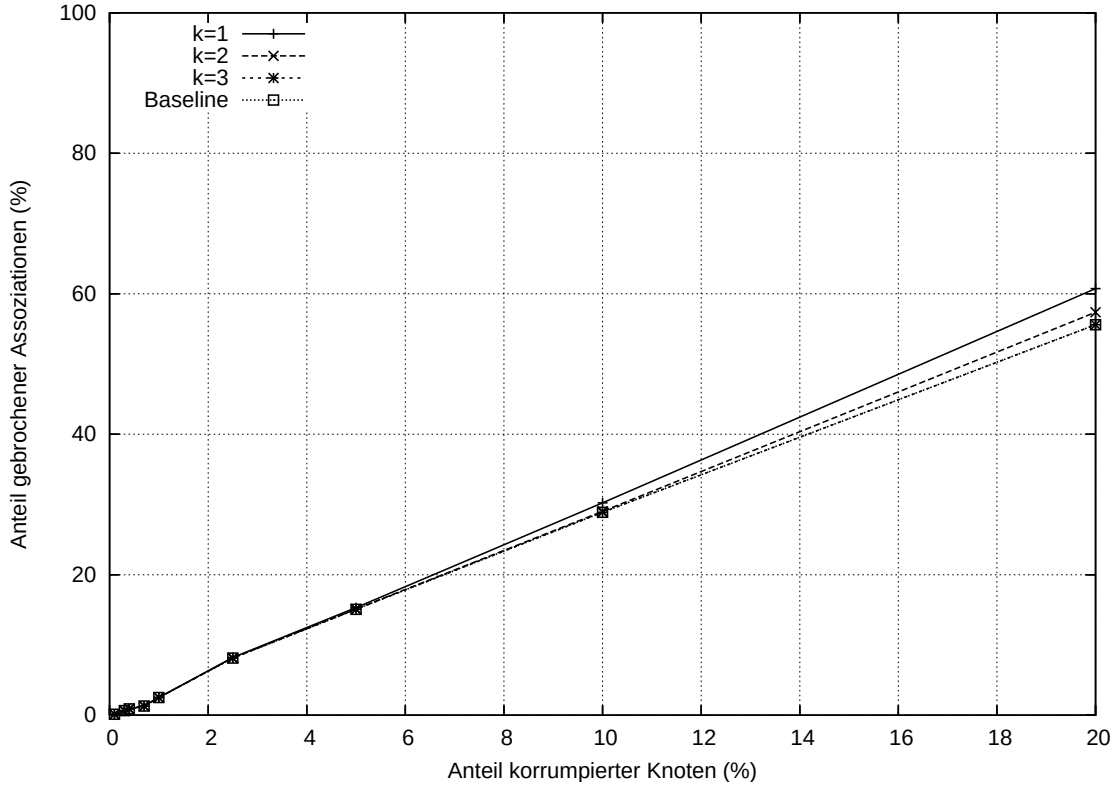


Abbildung 3.15 Reorganisation, Mehraufwand von [88] gegenüber SKEY, $\delta = 3$

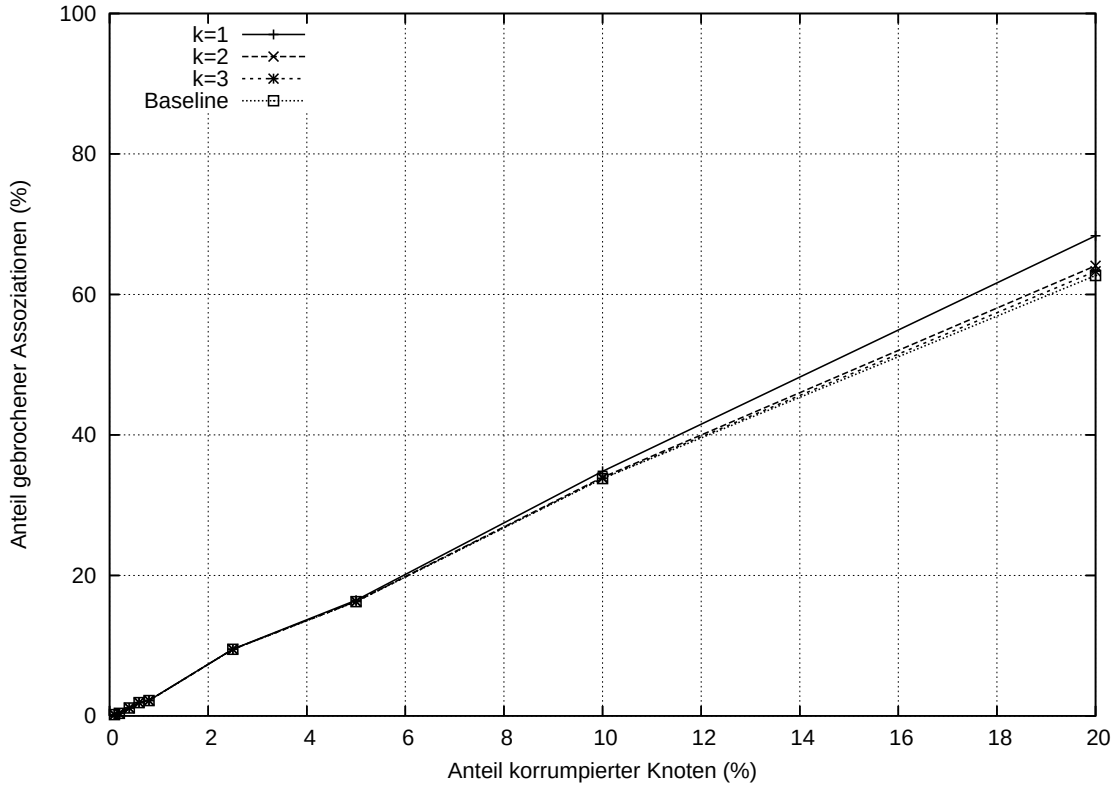
3.4.3.4 Simulationsergebnisse – Simulationen von Angreifern

Abbildung 3.16 zeigt die Ergebnisse der Angriffssimulationen sowohl bei $n = 3000$ als auch bei $n = 5000$ Knoten, wobei in den Simulationen die Anzahl korrumpierter Knoten $\mathcal{B}$ variiert wird. Die X-Achse zeigt in der Abbildung den sich ergebenen Prozentsatz β der vom Angreifer korrumpierter Knoten im Netz. Er berechnet sich aus $\mathcal{B}$ und der Gesamtanzahl Knoten n zu $\beta = \frac{\mathcal{B}}{n}$. Diese Knoten verhalten sich wie in der Einleitung zur Evaluierung auf Seite 92 beschrieben.

Der Angreifer korrumpiert, nachdem initial $\frac{n}{2}$ Knoten dem Netz beigetreten sind und bereits Schlüssel ausgetauscht haben, insgesamt $\mathcal{B}$ Knoten davon. In diesem Netz sind dementsprechend zunächst $2 \cdot \beta$ Prozent der Knoten korrumpiert. Dann treten dem Netz sukzessive weitere $\frac{n}{2}$ Knoten bei und tauschen mit SKEY Schlüssel aus. In diesem Netz sind $\beta\%$ der Knoten korrumpiert, das zeigt die X-Achse. Falls während des Schlüsselaustausches von den insgesamt $\mathcal{B}$ korrumpierten Knoten $k' > k$ auf einem Schlüsselaustauschpfad hintereinanderliegen, kennt der Angreifer den auszutauschen Schlüssel und bricht wie oben besprochen die darauf aufbauende Assoziation. Der Parameter k berechnet sich dabei laut Theorie aus Abschnitt 3.3.5.2 wie folgt: Bei $n = 3000$ muß k für $0 \leq \beta \leq 14$ mindestens den Wert 1 besitzen und für $14 < \beta \leq 20$ mindestens den Wert 2. Bei $n = 5000$ muß k für $0 \leq \beta \leq 13$ mindestens den Wert 1 besitzen und für $13 < \beta \leq 20$ mindestens den Wert 2. Soweit die Theorie: Um zu überprüfen, welche Auswirkungen vom Benutzer ein möglicherweise zu klein geschätztes $\mathcal{B}$ und damit k auf die Sicherheit vom Verfahren hat und welche Auswirkungen zufällige $k' > k$ Knoten auf den Austauschpfaden haben, ist k in den Simulationen *unabhängig* von den theoretischen Werten bei den verschiedenen β immer auf $k = 1, 2, 3$ gewählt worden.



(a) $n = 3000$ Knoten



(b) $n = 5000$ Knoten

Abbildung 3.16 Anteil gebrochener Assoziationen bei SKEY, Aggregationsgrad $\delta = 3$

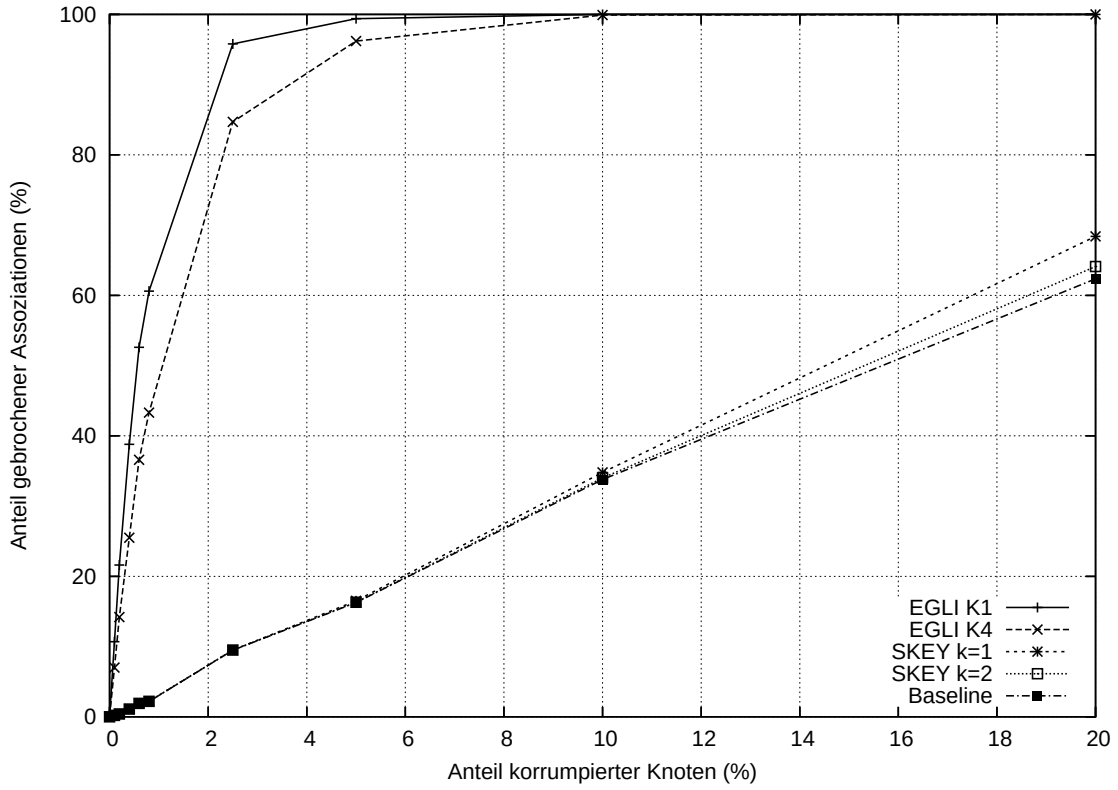


Abbildung 3.17 Assoziationssicherheit, Vergleich SKEY und [88], $n = 5000$ Knoten

Die Kurven in den Abbildungen 3.16(a) und 3.16(b) geben an, welchen Anteil an Schlüsseln und damit Assoziationen ein Angreifer auf diese Weise insgesamt *brechen* kann. Zuerst ist allerdings die mit *Baseline* beschriftete Kurve von Bedeutung: Sie gibt an, welchen Anteil an Assoziationen der Angreifer alleine durch die Übernahme der $\mathcal{B}$ Knoten brechen kann. Da pro übernommenem Knoten mehrere Assoziationen auf einmal in seine Gewalt gelangen und Knoten im „oberen Teil“ des Aggregationsbaums verglichen mit anderen im „unteren Teil“ wesentlich mehr Assoziationen eingehen, steigt diese Kurve überproportional mit β an. Diese Baseline-Kurve ist völlig unabhängig von jedwedem Protokoll zum Schlüsselaustausch, und sie kann nicht unterschritten werden: Sie hängt nur von den Fähigkeiten des Angreifers ab. Die Sicherheit eines Protokolls läßt sich daher nur messen, in wieweit ein Protokoll diese Kurve möglicherweise überschreitet.

Dies ist für SKEY mit verschiedenen k abgebildet. Je größer k, desto stärker nähert sich SKEY der Baseline an. Für $k > 3$ sind SKEY und die Baseline in den Simulationsergebnissen deckungsgleich.

In einem perfekten Sensornetz, in dem auf jedem Austauschpfad nur die wie oben theoretisch berechneten k korrumpierten Knoten hintereinanderliegen würden, lägen die SKEY Kurven mit der Baseline immer deckungsgleich übereinander. Da jedoch k sehr klein ist und aufgrund der zufälligen Verteilung der Angreifer mit einer bestimmten *Wahrscheinlichkeit* auch zufällig mehr als k korrumpierte Knoten auf Austauschpfaden hintereinanderliegen, gelingt es diesen Knoten, neue Schlüssel und damit Assoziationen abzuhören. Bei einer größeren Anzahl Knoten $n = 5000$ ist die Wahrscheinlichkeit bei gleichem β dafür häufiger als bei einer kleineren $n = 3000$, denn bei gleichem β ist beim größeren n auch die Gesamtanzahl korrumpierten Knoten $\mathcal{B}$ größer.

Unterschied zur Baseline nicht signifikant

Das Ergebnis zeigt: Der Unterschied zwischen der Baseline und SKEY mit Parameter $k = 1, 2, 3$ ist gering. Selbst bei sehr hoher Anzahl korrumpierter Knoten im Sensornetz überschreitet SKEY die Baseline sogar mit $k = 1$ bei $\beta > 13\%$ nur um wenige Prozentpunkte. Dies liegt daran, daß es trotz des festem k bei variierendem $\mathcal{B}$ doch sehr unwahrscheinlich ist, daß $k' > k$ Knoten hintereinander auf Austauschpfaden liegen. Damit kann man festhalten, daß selbst wenn sich der Benutzer beim Berechnen von k durch Abschätzen von $\mathcal{B}$ „nach unten“ verschätzt, dennoch ein hohes Maß an Sicherheit gewährleistet wird. Das gute Abschneiden von SKEY wird neben dem von Eschenauer und Gligor [88] in Abbildung 3.17 verdeutlicht. Der Anteil gebrochener Assoziationen steigt bei EGLI sehr schnell. Das liegt an den in Anhang B, S. 175 besprochenen Sicherheitsschwächen des Protokolls, die dem Angreifer viele Möglichkeiten bieten, auszutauschende Schlüssel abzuhören und Assoziationen zu brechen.

SKEY stellt sich bei denen in dieser Arbeit getroffenen Annahmen, in aggregierenden Netzen und bei byzantinischen Knoten als wesentlich sicherer als Schlüssellisten-basierte Protokolle dar.

3.4.4 Ergebnisse im Überblick

SKEY erfüllt die vorab geforderten Entwurfsziele:

- *Funktionalität*
  SKEY erlaubt den Schlüsselaustausch zwischen allen Knoten x_i und allen Kommunikationspartnern auf ihrem Aggregationspfad $\mathbb{P}_{x_i}$. Grundsätzlich ist damit zwischen diesen Knoten jeweils sichere Kommunikation möglich. Es existiert die im Grundlagenkapitel definierte Basissicherheit zwischen jeweils zwei Knoten.

- *Sicherheit*
  Das Protokoll ist dabei sicher gegen bis zu k viele korrumpierte Knoten hintereinander auf den Austauschpfaden der Schlüssel. Dem Angreifer gelingt es mit Hilfe seiner korrumpierten Knoten nicht, den Schlüsselaustausch so abzuhören oder zu manipulieren, daß er in Kenntnis des zwischen zwei Knoten auszutauschenden Schlüssels gelangt.

  Falls der Benutzer vor Einsatz des Sensornetzes eine grobe Abschätzung über die erwartete Gesamtzahl der korrumpierten Knoten treffen kann und auch die voraussichtlichen Gesamtanzahl aller Sensorknoten im Netz abzusehen ist, dann kann der Benutzer die Zahl k wie beschrieben berechnen. Letzteres wird in der Realität aber eher unwahrscheinlich sein.

  Falls sich der Angreifer und die Gesamtanzahl korrumpierter Knoten $\mathcal{B}$ a priori nicht abschätzen läßt, kann der Benutzer k so wählen, daß sein Schlüsselaustausch mindestens gegen eine bestimmte Anzahl korrumpierter Knoten $\mathcal{B}$ noch sicher funktioniert. Der Benutzer kann auf diese Weise einen Kompromiß zwischen Sicherheit gegen möglicherweise vielen korrumpierten Knoten und dem daraus resultierenden Energieverbrauch finden.

- *Dezentralität*
  SKEY verzichtet auf den Einsatz irgendwelcher zentraler Infrastrukturkomponenten, wie zentralen Servern, Certification Authorities oder besonderer Sensorkno-

ten, beispielsweise besonders ressourcenstarker Knoten, die bei jedem Schlüsselaustausch assistieren. Es gibt keinerlei besondere Abhängigkeit von irgendwelchen dedizierten Knoten.

Auch die Senke übernimmt beim normalen Schlüsselaustausch keine herausragende Stellung ein, sondern nur in seltenen Situationen, in denen $k' \leq k$ gemeinsame Vorgänger im Aggregationsbaum gefunden werden. Sie ist nicht zwangsläufig in jeden Schlüsselaustausch zwischen zwei beliebigen Knoten involviert, sondern arbeitet diesbezüglich genau wie jeder andere Knoten im Netz und muß – genau wie jeder andere Knoten als unter Umständen möglicher gemeinsamer Vorgänger – dann lediglich Shares weiterleiten. Falls die Senke als weiterleitender Knoten ausfallen würde oder nicht erreichbar wäre, hätte dies die selben Konsequenzen wie der Ausfall jedes beliebigen anderen gemeinsamen Vorgängerknotens.

Auch das Master Device MD stellt diesbezüglich keine zentrale Komponente dar. Es dient nur dem Paaren von Sensoren vor ihrem Einsatz. Es stellt lediglich einen für Sensornetze geeigneten Trusted Dealer [65, 219] dar, der für sichere Kommunikation zwischen Knoten *immer* notwendig ist [14].

- *Spontaneität und Selbstorganisation*
  Der gesamte Schlüsselaustausch eines Knotens organisiert sich selbst. Der Benutzer interagiert nach dem Ausbringen nicht mehr mit einem Knoten sondern tritt lediglich beim einmaligen Paaren eines neuen Knotens mit Hilfe eines Master Devices in Erscheinung. Ein initiales, sicheres Paaren neuer Knoten durch den Benutzer ist allerdings auf jeden Fall immer erforderlich und völlig unabhängig von einem Schlüsselaustauschprotokoll. Der Benutzer muß zudem zu keinem Zeitpunkt die Netzwerkkonfiguration kennen.

- *Dynamik*
  Dynamisches Knotenverhalten, insbesondere der spontane Ausfall von Knoten impliziert kein besonderes Sicherheitsproblem bei SKEY. Es ist keine zusätzliche Energie, Nachrichten oder kryptographische Operationen notwendig, um Knotenausfall zu kompensieren oder zu reparieren. Auch der Wechsel von Aggregationsbeziehungen innerhalb des Sensornetzes ist ohne großen Aufwand möglich. Grundsätzlich unterstützt SKEY sich während des Betriebs des Sensornetzes variierende k.

- *Effizienz*
  Der durch SKEY entstehende Aufwand in puncto Speicherverbrauch durch kryptographische Schlüssel sowie Energieverbrauch durch den Versand und das Ver- und Entschlüsseln von Nachrichten bleibt sehr gering: Sowohl Speicher- als auch Energieaufwand steigen pro Knoten nur logarithmisch mit der Gesamtgröße des Sensornetzes. Als kryptographische Primitive kommen lediglich eine günstige *symmetrische* Chiffre zum Einsatz. Dies ermöglicht den Einsatz von SKEY nicht nur auf der anvisierten MICA2 Plattform, sondern auch auf anderen ressourcenbeschränkten Plattformen. Da SKEY für jede Kommunikationsassoziationen paarweise verschiedene Schlüssel benutzt, wird keine Rücknahme von Schlüssel notwendig.

3.5 Zusammenfassung

Ein geeignetes Verfahren zum Schlüsselaustausch kann als zwingend notwendige Grundlage oder Basis jedes weiteren, komplexeren Sicherheitsprotokolls verstanden werden.

Dieses Kapitel hat zunächst analysiert, welche neuen und besonderen Herausforderungen aggregierende Sensornetze an Protokolle zum Schlüsselaustausch stellen. Im Anschluß daran sind entsprechende Kriterien oder Entwurfsziele bestimmt worden, die ein geeignetes, effizientes Schlüsselaustauschprototokoll erfüllen soll.

Eine Übersicht über den aktuellen Stand der Forschungs hat gezeigt, daß bisher veröffentlichte Lösungen, insbesondere die populären und häufig zitierten *Random Key Pre-Distribution*-Ansätze, nicht besonders gut mit diesen Entwurfszielen harmonieren: Sie stellen sich letztendlich als sehr ineffizient, unflexibel, unsicher und damit ungeeignet heraus.

Im Rahmen dieser Arbeit ist SKEY, ein neues, effizientes Schlüsselaustauschprotokoll für aggregierende Sensornetze, entworfen worden. Der Kommunikationsfluß in Sensornetzen, von Blättern bis hin zu einer Datensenke, folgt typischerweise einer vorgegebenen Hierarchie: der baumartigen Aggregation. Innerhalb eines solchen aggregierenden Sensornetzes können mit Hilfe von SKEY speicher- und energieeffizient Schlüssel zwischen Knoten ausgetauscht werden. SKEY teilt zunächst Schlüssel in sogenannte *Shares* auf, die dann chiffriert über mehrere Schlüsselaustauschpfade im Netz in Richtung Zielknoten verschickt werden.

Auf diese Weise leistet SKEY

- einen sicheren Schlüsselaustausch zwischen Knoten, die aufgrund des Aggregationsbaumes miteinander kommunizieren müssen. Der Schlüsselaustausch ist sicher gegen eine parametrisierbare Anzahl byzantinischer Sensorknoten.
- Der Ressourcenverbrauch, der Energieverbrauch und der Speicherverbrauch von SKEY, steigt dabei nur logarithmisch mit der Gesamtzahl der im Netz befindlichen Knoten.

Ausblick auf das nächste Kapitel

Ein geeigneter, effizienter Schlüsselaustausch stellt lediglich die Grundlage für komplexere Sicherheitprotokolle dar. Alle Aufgaben, die über die reine Sicherung der Kommunikation zwischen genau zwei Knoten hinausgehen, beispielsweise Authentizität beim aggregierenden Datentransport, die Verifikation des Datenursprungs, werden damit nicht gelöst, sondern bauen darauf auf.

4. Authentische Aggregation

4.1 Motivation

Durch den paarweisen Austausch von Schlüsseln im Sensornetz ist nun die Basissicherheit der transportierten Daten zwischen jeweils zwei Knoten im Netz gesichert. Ein weiteres, offensichtliches Problem bleibt allerdings auch trotz Schlüsselaustausch noch offen: die Authentizität oder Echtheitsprüfung der Aggregate. Empfängt die Senke oder irgendein beliebiger Knoten im Netz ein Aggregat von einem Aggregationsknoten, so kann sie sich zunächst der Tatsache sicher sein, daß dieses Aggregat auf dem Pfad vom Aggregationsknoten zu ihr weder modifiziert noch abgehört worden ist. Offen jedoch verbleiben die Fragen nach

1. der Korrektheit der Aggregation. Der Aggregationsknoten könnte korrumpiert worden sein und die empfangenen Daten bewußt falsch aggregieren, um der Senke falsche Meßwerte vorzugaukeln. Um beim Beispiel des betreuten Wohnens zu bleiben: Das Display als Senke würde auf Grund falsch empfangener Blutdruckwerte eine zu hohe Medikamentierung vorschlagen.

2. den ursprünglichen verantwortlichen Sensoren für das Aggregat. Die Senke oder ein anderer, das Aggregat empfangende Knoten muß in der Lage sein zu verifizieren, ob sich das Aggregat aus Daten zusammensetzt, die von legitimen beziehungsweise damit „beauftragten“ Knoten aus dem Sensornetz stammen. Die Senke muß sicher sein, daß die ursprünglichen Blutdruckwerte tatsächlich von Patient 1 stammen und nicht von Patient 2.

Ziel des Kapitels: Die beiden obigen Punkte bilden zusammengefaßt das Problem des authentischen, aggregierenden Datentransports. Zunächst wird gezeigt, daß authentischer Datentransport im Sensornetz teilweise widersprüchlich zu der Idee des Energiesparens durch Aggregation ist. Neben einer genauen Analyse der spezifischen Probleme authentischen Datentransports in Sensornetzen werden darauf die Entwurfsziele für ein geeignetes, sicheres Aggregationsprotokoll skizziert. Das sind wieder Schlagworte wie zum

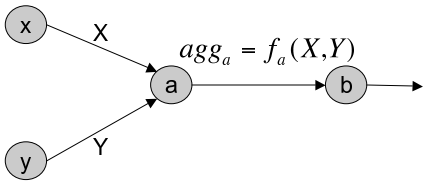


Abbildung 4.1 Einfache Aggregation

Beispiel *Dezentralität*, *Sicherheit* oder *Effizienz*, allerdings hier im Gegensatz zum Schlüsselaustausch fokussiert auf die Bedeutung für authentische Aggregation von Daten. Die Präsentation über den Stand der Forschung faßt zusammen, daß bisher noch kein Ansatz für allgemein mathematische, kaskadierende Formen von authentischer Aggregation existiert, der Sicherheit in Gegenwart von byzantinischen Knoten garantiert. Schließlich wird das in dieser Arbeit entwickelte Protokoll ESAWN präsentiert, das alle aufgestellen Entwurfsziele erfüllt. Das Prinzip hinter ESAWN ist, einen Kompromiß zwischen teilweiser Aggregation und probabilistischer Authentizität im Netz zu realisieren, der sich vom Benutzer parametrisieren läßt

Aufbau des Kapitels: In Abschnitt 4.1.1 werden die im Grundlagenkapitel erwähnten, besonderen Eigenschaften von Sensornetzen im Hinblick auf deren Auswirkungen *speziell für authentische Aggregation* analysiert. Im Anschluß daran werden in Abschnitt 4.1.2 Entwurfsziele formuliert, Richtlinien für ein geeignetes Aggregationsprotokoll. Den aktuellen Stand der Forschung zeigt Abschnitt 4.2, zusammengefaßt in Tabelle 4.1, Seite 121. Das Protokoll ESAWN wird in Abschnitt 4.3 vorgestellt; eine dazugehörende Evaluierung folgt in Abschnitt 4.4.

4.1.1 Neue Herausforderungen

Um die Authentizität von Daten zu garantieren, kommen in klassischen Netzen typischerweise digitale Unterschriften [156] zum Einsatz. Der folgende Abschnitt zeigt, daß diese Technik für den Einsatz in aggregierenden Sensornetzen nicht geeignet ist und welche Kriterien ein Protokoll für authentischen, aggregierenden Datentransport stattdessen erfüllen muß: Die *Entwurfsziele* für authentische Aggregation werden motiviert.

4.1.1.1 Kommunikationsfluß: Aggregation

Die Aggregation der Daten stellt sich als Widerspruch zur Authentizität dar. Durch die Aggregation von zwei oder mehreren Meßwerten zu einem Aggregat, siehe einfaches Beispiel in Abbildung 4.1, wird die ursprüngliche Information, die in den Meßwerten enthalten ist, nun aggregiert weitergebeben. Das von a gebildete Aggregat $\mathrm{agg}_x = f_a(X, Y)$ enthält möglicherweise keinerlei Hinweise mehr auf die zu seiner Bildung beigetragenen Daten X, Y der Knoten x und y. Der Empfänger b des Aggregats kann die Aggregation und auch die für die Messung verantwortlichen Sensoren x und y daher nicht rein anhand des von a empfangenen Aggregats überprüfen. Zur Überprüfung müßte der empfangende Knoten zusätzlich zum Aggregat von a noch weitere Information über die ursprünglichen Meßwerte kennen: Im Falle einer Hash-Funktion als Aggregationsfunktion müßte er sogar die kompletten Meßwerte kennen. Die Quellsensoren x und y müßten ihm diese Meßwerte zuschicken. Grundsätzlich mag das möglich sein, widerspricht jedoch der Idee der Aggregation: Wenn die messenden Sensoren ihren Meßwert sowieso an den empfangenden Knoten b schicken, kann die Aggregation auch eingespart werden. Die Authentizität der Daten steht demnach widersprüchlich zur Energieeinsparung durch die Aggregation.

Ein Sensornetz kann nicht gleichzeitig authentisch *und* (effizient) aggregierend Daten transportieren. Soll Authentizität überprüft werden, ist zusätzlich zur Aggregation der Versand der Originaldaten Daten an den Empfänger eines Aggregat notwendig. Um dieses Dilemma zu lösen und in einem aggregierenden Sensornetz dennoch authentisch und effizient Daten zu transportieren, wird man daher das klassischen Verständnis von Authentizität, ein empfangenes Datum ist authentisch oder nicht, erweitern müssen.

4.1.1.2 Ressourcenarmut

Die beschränkten Hardware-Ressourcen der Sensoren führen dazu, daß die auf Public-Key-Kryptographie basierenden digitalen Signaturen zwar grundsätzlich möglich sind, wie in Gura et al. [99], Malan et al. [152] und auch im Rahmen dieser Arbeit Blaß et al. [29], Blaß und Zitterbart [32, 33], Junker [119] gezeigt. Im Vergleich zu rein symmetrischen Primitiven sind sie jedoch extrem teuer und damit eher zu vermeiden. Werden im Sensornetz in extrem kurzen Perioden Daten gemessen und verschickt, kann alleine die Berechnung der zum Meßwert gehörenden digitalen Signatur die Periode überschreiten. Ein Beispiel: Die Anfertigung der Signatur benötigt knapp 6s [29], die Meßperiode der aktuellen Zimmertemperatur liegt jedoch bei 1s. Auf asymmetrische Techniken und damit digitale Signaturen muß in Sensornetzen weitestgehend verzichtet werden. Dies gilt insbesondere dann, wenn Daten relativ zeitnah durch das Sensornetz transportiert werden müssen, was meist der Fall ist.

4.1.1.3 Übrige Anforderungen

Natürlich steht authentischer Datentransport in Sensornetzen auch den übrigen Problemen wie fehlenden Infrastrukturen, Selbstorganisation oder spontaner Netzbildung gegenüber. Sie werden an dieser Stelle jedoch nicht nochmals erwähnt, da ihre Auswirkungen in Kapitel 2 analysiert wurden und sie keine besondere Herausforderung für authentischen Datentransport darstellen. Weiterhin hat das spontane Beitreten von Knoten zum Netz (Spontaneität) keinerlei Auswirkungen auf die Aggregation von Daten: An dieser Stelle wird davon ausgegangen, daß ein neuer, dem Netz beitretender Knoten, Teil des Aggregationsbaumes ist und Schlüssel anhand des Aggregationsbaumes austauscht. Auch der Ausfall eines Knotens soll an dieser Stelle keinerlei Beachtung finden: Fällt ein Knoten aus, so soll davon ausgegangen werden, daß eine entsprechende Software-Komponente den Aggregationsbaum „repariert“.

Dieses Kapitel geht davon aus, daß ein Aggregationsbaum existiert und Knoten anhand ihrer Aggregationspfade Schlüssel miteinander besitzen.

4.1.2 Entwurfsziele und erwartete Ergebnisse

An dieser Stelle lassen sich für ein ideales Protokoll zum authentischen Datentransport in Sensornetzen die folgenden Entwurfsziele aufstellen. Zunächst wieder die eigentliche Aufgabenstellung von Authentizität:

- Entwurfsziel *Funktionalität*
  Dieses Entwurfsziel besteht aus zwei Teilen:

  1. Das Protokoll soll die Authentizität der aggregierten Daten überprüfbar machen. Dies bedeutet konkret, daß jeder Aggregationsknoten wie b aus Abbildung 4.1 in der Lage ist, ein von a empfangenes Aggregat agg_a zu überprüfen.

Es gelingt dabei Knoten b nicht nur, festzustellen, ob sich das Aggregat ursprünglich aus Meßwerten der Knoten x und y zusammensetzt, sondern auch, ob a korrekt aggregiert hat.

Dabei ist wichtig, daß diese Überprüfung von agg_a durch b möglich ist, *bevor* b selbst sein Aggregat bildet und weiterschickt – und nicht zu einem späteren Zeitpunkt. Knoten b soll kein Aggregat agg_b erzeugen und weiterschicken, das auf einem unter Umständen gefälschten Aggregat basiert. Beispiel: Das Display beim betreuten Wohnen soll die Korrektheit eines empfangenen Aggregat überprüfen können, bevor es den Notarzt ruft – und nicht erst im nachhinein.

2. Grundsätzlich sollte das Protokoll dabei komplexere Aggregationsbäume, wie im Grundlagenkapitel in Abschnitt 2.2.4, S. 17 definiert, unterstützen: das heißt kaskadierende, mehrfach hintereinander ausgeführte Aggregationen mit beliebigen, berechenbaren Aggregationsfunktionen.

- Entwurfsziel *Sicherheit*
  Das Protokoll stellt diese Funktionalität sicher zur Verfügung. Das Protokoll arbeitet auch in Gegenwart von unter Umständen mehreren korrumpierten Knoten noch wie gewünscht. Aggregate werden authentisch ausgeliefert, auch wenn am Protokoll beteiligte Knoten vom Angreifer korrumpiert sind.

Außerdem muß ein geeignetes Protokoll die folgenden beiden, sensornetztypischen Eigenschaften erfüllen:

- Entwurfsziel *Dezentralität*
  Das Protokoll darf sich nicht auf die Verfügbarkeit von zentralen Komponenten oder dedizierten Knoten im Netz verlassen. Für die Authentizität steht kein besonderer Server, ein „Notar“ [202] wie häufig in bisherigen Forschungsarbeiten, zur Verfügung. Das Protokoll muß komplett dezentral ablaufen können.

- Entwurfsziel *Effizienz*
  Aggregation dient in Sensornetzen in erster Linie der Energieeinsparung. Der Mehraufwand an Energie durch Absichern der Authentizität von Aggregaten muß verglichen mit dem Aufwand, der für sicheren Datentransport komplett ohne Aggregation im Netz anfällt, immernoch geringer bleiben.

  Kurz: Das sichere Aggregieren von Daten soll energetisch günstiger sein, als sicherer Datentransport *ohne* Aggregation. Andernfalls würde sich der Einsatz sicherer Aggregation im Sensornetz aus Energiesicht nicht lohnen.

(Auch die Aktualität eines Aggregats kann von Bedeutung sein. Hierauf soll allerdings nicht weiter eingegangen werden, da sich diese bereits mit sicheren Zeitstempeln aus Abschnitt 2.6.2.4, S. 45 überprüfen läßt.)

Bisher gilt der Transport von Daten *entweder* als authentisch *oder* als nicht-authentisch – Authentizität von Daten ist eine binäre Eigenschaft. Verdeutlicht man sich jetzt nocheinmal die Tatsache, daß Authentizität und Aggregation im Widerspruch zueinander stehen, weil Authentizität die Übertragung zusätzlicher Meßdaten impliziert, ist das *erwartete Ergebnis* von authentischer Aggregation ein *Kompromiß* aus beidem.

Graduelle, probabilistische Authentizität

Da Authentizität in Form von zusätzlich übertragener Information die durch Aggregation eingesparte Energie wieder verbraucht, muß man Authentizität in Sensornetzen in ihrer klassischen Bedeutung erweitern. Die Überprüfbarkeit *jedes* Aggregates wird nicht *immer* mit beliebiger Anzahl korrumpierter Knoten im Netz und vor allem *effizient* möglich sein. Dieses Anforderung entspräche der kompletten Aufhebung der Aggregation und stellt sich damit schlicht als „zu teuer“ heraus, wobei „zu teuer“ verglichen mit Authentizität beim Verzicht auf Aggregation meint. Authentische Aggregation macht nur dann Sinn, wenn sie günstiger als authentischer, nicht-aggregierender Datenfluß bleibt.

Stattdessen soll, ähnlich wie beim Schlüsselaustausch mit SKEY, ein Sicherheit–Energie-Tradeoff erzielt werden: Anstatt der vollständigen Authentizität wie bisher garantiert ein Protokoll nur *graduelle* Authentizität. Das bedeutet Authentizität mit einer Wahrscheinlichkeit $p \leq 100\%$ in Gegenwart von k am Protokoll beteiligten, jedoch korrumpierten Knoten. Dadurch verschlechtert sich auf der einen Seite zwar die Sicherheit der Daten, auf der anderen Seite sind aber dann doch noch Energieeinsparungen möglich.

4.2 Stand der Forschung

Bisher gibt es, im Vergleich zum Schlüsselaustausch, relativ wenig Forschungsarbeiten, die sich mit Protokollen für authentischen, aggregierenden Datentransport beschäftigen. Die dazu bisher veröffentlichten Arbeiten weisen zudem noch einige Unzulänglichkeiten auf, wie die folgende Übersicht zeigt.

4.2.1 Authentizität durch Homomorphismen

Einige Arbeiten, darunter Acharya et al. [2], Castelluccia et al. [49], Girao et al. [95], Mykletun et al. [162], Westhoff et al. [241, 242], basieren auf der Idee, sogenannte *Privacy Homomorphisms* (PHs) in Sensornetzen zu verwenden. PHs entstammen ursprünglich aus der Forschung über Datenbanksicherheit und sind bereits 1978 von Rivest in Rivest et al. [189] erwähnt worden. Bei einem Privacy Homomorphism handelt es sich um eine kryptographische Verschlüsselungsfunktion F mit homomorphen, das heißt linearen Eigenschaften. Sei $\mathbb{P}$ die Menge der Klartexte und $\mathbb{C}$ die Menge der Chiffrate, dann bildet F Elemente $a, b \in \mathbb{P}$ nach $F(a), F(b) \in \mathbb{C}$ so ab, daß bzgl. Addition „+“ und Multiplikation „*“ gilt: $F(a + b) = F(a) + F(b)$ und $F(a * b) = F(a) * F(b)$. Ein Aggregationsknoten kann damit seine Aggregationsfunktion, beispielsweise das Addieren von zwei Meßwerten, durchführen, ohne die Meßwerte zu kennen. Er operiert ausschließlich auf den beiden Chiffraten, die für die Aggregation nicht entschlüsselt werden müssen. Die Quellsensoren können auf diese Weise ihre Meßwerte durch paarweise Schlüssel mit der Senke chiffrieren – solch ein Datentransport zur Senke ist trotz Aggregation authentisch.

Obwohl es sich hierbei eigentlich um einen sehr eleganten Ansatz zur Lösung des Problems handelt, eignen sich PHs aus den folgenden Gründen dennoch nicht für authentischen Datentransport:

- Die Technologie hinter PHs basiert auf Public-Key ähnlicher Mathematik. Es müssen zum Beispiel teure Modulo-Exponentiationen berechnet werden, auf der eingeschränkt leistungsfähigen Sensorhardware ein problematischer Ansatz [123]. Wird

die Größe der Exponenten auf ein in Sensornetzen realistisches Maß verringert, leidet die Sicherheit, wie in Girao et al. [95], Westhoff et al. [241], erheblich. Die prinzipielle Einsatzfähigkeit von PHs auf limitierter Sensor-Hardware ist damit noch völlig unklar.

- Zudem vervielfacht sich die Länge der Chiffrate je nach Homomorphismus um einen bestimmten, ganzzahligen Faktor $d \geq 2$ im Vergleich zum Klartext [95, 176, 241]. Durch die längeren Chiffrate erhöht sich deren Speicherbedarf, es können pro Knoten weniger Chiffrate gespeichert werden und der Energiebedarf für das drahtlose Übertragen von Chiffraten erhöht sich.

- Die zum Einsatz kommenden PHs sind anfällig gegen jede Form von sogenannten Known-Plaintext-Angriffen [80] so wie auch Chosen-Plaintext-Angriffen [233] und damit unsicher – solche Angriffe sind in Sensornetzen allerdings sehr realistisch. Ein Beispiel: Installiert der Angreifer heimlich neben einen Temperatursensor im Zimmer des Patienten einen eigenen zweiten Temperatursensor, kennt er mit hoher Wahrscheinlichkeit die vom originalen Sensor gemessenen Temperaturwerte. Damit ist ein Known-Plaintext-Angriff möglich.

- Ein gravierender Nachteil: Die möglichen Aggregationsfunktionen sind rein auf Kombinationen von Addition und Multiplikation beschränkt. Jede etwas komplexere Aggregationsfunktion muß sich aus einer Menge von Additionen und Multiplikationen zusammensetzen. Das heißt, daß zur Auswertung der Aggregationsfunktion dann äußerst viele Additionen und Multiplikationen notwendig sind – was wiederum in enormem Zeit- und Energieaufwand mündet. Keinerlei *nicht-lineare* Funktionen lassen sich mit diesem Ansatz umsetzen. (Ein Beispiel für eine einfache nichtlineare Funktion ist die, die beim Überschreiten einer bestimmten Herzfrequenz oder eines Blutdruckwertes einen Alarm auslöst.)

- Rivest hat in seiner ursprünglichen Veröffentlichung in Rivest et al. [189] bereits bewiesen, daß manche Typen von Operationen, wie aggregierte Maxima- und Minima-Berechnung, mit PHs unmöglich beziehungsweise mit hohem Speicheraufwand [2] verbunden sind. In Sensornetzen kann man sich aber auch diese Operationen als Teil einer komplexen Aggregationsfunktion vorstellen.

- Auf eine weitere Problematik weist Castelluccia et al. [49] hin. Einige Funktionen, wie die Berechnung eines aggregierten Durchschnittswertes, benötigen bei PHs auf dem Aggregationsknoten Informationen, die ihm unter Umständen häufig gar nicht zur Verfügung stehen, so beispielsweise die aktuelle Gesamtanzahl aller Knoten im Netz.

- Ein weiteres Problem von PHs liegt in der Natur der Sache: Aggregationsknoten kennen den Inhalt der empfangenen Meßwerte nicht mehr, sie operieren rein auf den Chiffraten. Dies mag zwar in einigen Szenarien von Vorteil sein, in anderen jedoch inakzeptabel. Gerade dann, wenn einer der Aggregationsknoten auf dem Pfad zur Senke als Aktor die Meßwerte kennen *muß*, eignen sich PHs nicht. Als Beispiel sei hier ein Piepser am Körper des Patienten aus Abschnitt 2.2.4, S. 21 genannt, der unabhängig vom Display an der Senke auf einen kritischen Herzzustand hinweist.

- Ein Angreifer kann bei den PHs der bisherigen Arbeiten außerdem erfolgreich Replay-Attacken durchführen. Alte, zu einem früheren Zeitpunkt bereits verschickte Chiffrate können von einem korrumpierten Aggregationsknoten wiederholt in die

Aggregatbildung mit einbezogen werden – unbemerkt von anderen Knoten wie der Senke. Damit ist die Authentizität von Aggregaten unsicher.

Obwohl PHs auf den ersten Blick als elegante Lösung des Authentizitätsproblems erscheinen, verstoßen sich doch gegen die Entwurfsziele *Effizienz*, *Funktionalität* und *Sicherheit*. Neben den völlig unklaren Sicherheits- und Leistungsaussagen lassen insbesondere die Nachteile, keine komplexeren Aggregationsfunktionen und eine beliebige Verarbeitung von Meßwerten auf Aggregationsknoten zu unterstützen, ihren Einsatz in Sensornetzen scheitern.

4.2.2 Einschränkung der Aggregationsfunktion

Durch eine starke Einschränkung der möglichen Aggregationsfunktionen erreichen Arbeiten wie Jadia und Mathuria [118], Önen und Molva [167], Raina et al. [182] authentischen Datentransport.

Ähnlich zu Privacy Homomorphismen erlauben Jadia und Mathuria [118], Önen und Molva [167] nur Ganzzahladdition als Aggregationsfunktion. Zwei Sensorknoten a und b besitzen genau wie ihr Aggregationsknoten c paarweise verschiedene Schlüssel mit der Senke d: $K_{a,d}$, $K_{b,d}$ und $K_{c,d}$. Knoten a verschlüsselt seinen Meßwert x mit $C_1 = (x + K_{a,d})$, b chiffriert y zu $C_2 = (y + K_{b,d})$. Der Aggregationsknoten c kann die Chiffrate nicht entschlüsseln, lediglich auf den Chiffraten die Aggregation ausführen: $C_3 = C_1 + C_2 + K_{c,d}$. Die Basisstation d entschlüsselt schließlich C_3 zu $x + y = C_3 - K_{a,d} - K_{b,d} - K_{c,d}$. Auch dieser Ansatz erlaubt nur eine bestimmte Form der Aggregation, nämlich Addition. Aggregationsknoten sind nicht in der Lage, von Sensoren empfangene Meßwerte in irgendeiner Form zu interpretieren. Damit erfüllen Jadia und Mathuria [118], Önen und Molva [167] nicht die geforderte Eigenschaft der *Funktionalität*. Außerdem ist dieser Ansatz *unsicher*. Falls der Angreifer einen Meßwert x kennt oder zufällig rät, kann er den Schlüssel $K_{a,d}$ berechnen: $K_{a,d} = C_1 - x$.

Raina et al. [182] erlauben nur sogenannte *quasi-kommutative* Aggregationsfunktionen f, für die $f(x_1, x_2, \ldots, x_k) = f(x_1, f(x_2, x_3, \ldots), \ldots, x_k)$ gilt, siehe auch Benaloh und de Mare [21]. Beispiele solcher Funktionen sind Maximal-, Minimalwertberechnung, der Median usw. In einem relativ aufwendigen Verfahren schickt das vorgeschlagene Protokoll zunächst Hash-Werte von Meßwerten an Aggregationsknoten, die im zweiten Schritt schließlich überprüft werden. Dabei kommt allerdings mehrfache Modulo-Exponentiation mit großen Operanden zum Einsatz. Deren grundsätzlich fragwürdige Einsetzbarkeit auf Sensorknoten bleibt in der Arbeit unbeantwortet. Das Protokoll transportiert die Daten darüberhinaus von Knoten zu Knoten im Klartext, Verschlüsselung findet nicht statt. Falls es einem Angreifer gelingt, mehr als einen der am Protokoll beteiligten Knoten zu korrumpieren, kann er die Sicherheitsmechanismen umgehen. Auch Raina et al. [182] widerspricht damit den geforderten Entwurfsziele *Effizienz, Funktionalität* und *Sicherheit*.

Das Protokoll SIA aus Przydatek et al. [180] kommt einem Sicherheit–Energie-Tradeoff nahe: Die Senke kann das komplette Aggregat übeprüfen, in dem sie einzelne Sensoren und Aggregationsknoten um die gemessenen beziehungsweise teilweise aggregierten Werte bittet. Für bestimmte mathematische Funktionen, wie zum Beispiel den Mittelwert oder den Median, kann die Senke aus den ihr nun teilweise vorliegenden Werten die Korrektheit des Gesamtaggregats rekonstruieren – mit einem bestimmten Fehler. Je *mehr Stichproben* die Senke auswählt, desto *sicherer*, das heißt *korrekter* oder *authentischer* wird das Aggregat. Allerdings beschränkt sich SIA auch auf wenige mathematische

Funktionen. Bestimmte Vergleiche von Meßwerten können gar nicht behandelt werden. Dadurch wird das Entwurfsziel *Funktionalität* nicht erfüllt. Außerdem basiert SIA auf μTesla, siehe Perrig et al. [172, 173, 174], einem infrastruktur-basierten Schlüsselaustausch, ein Widerspruch zur *Dezentralität*.

SIA wird von Chan et al. [59] modifiziert, in dem einzelne Knoten vor dem Versenden ihrer tatsächlichen Meßwerte beziehungsweise Aggregate ein *Commitment*, den Hash des Meßwertes oder Aggregates in Richtung Senke versenden. Dadurch bildet sich eine Merkle-Hash-Baum [157]. Nachdem die Senke die Wurzel dieses Hash-Baums empfängt, flutet sie die Wurzel zurück ins Netz. Damit können einzelne Knoten im Sensornetz überprüfen, ob ihre Daten bzgl. einfacher Aggregationsfunktionen wie *Summe*, *Median* oder *Mittelwert* korrekt aggregiert worden sind. Auch dieses Verfahren benötigt dabei das Protokoll μTesla und erfüllt das Entwurfsziel *Dezentralität* nicht. Durch die starke Einschränkung der Aggregationsfunktionen erfüllt es weiterhin das Entwurfsziel *Funktionalität* nicht, und das aufwendige Fluten der Wurzel des Hash-Baumes in das gesamte Sensornetz erfüllt *Effizienz* nicht.

4.2.3 Verfahren zur Klassifizierung von Daten

Andere Veröffentlichungen basieren auf der Idee, Meßwerte in Kategorien beziehungsweise Klassen einzuteilen. Ein Aggregationsknoten entscheidet dann anhand der Klassifizierung über die Aggregation des Meßwertes.

Auf diese Weise findet in Wu et al. [246] eine vereinfachte Form von Datenaggregation statt: Weiterleitung von Meßwerten oder Verwerfen von Meßwerten. Ein Sensor kann seinen Meßwert für die Senke verschlüsseln und nach fest vorgegebenen Stichwörtern *klassifizieren*. Die Klassifizierung schickt der Sensor zusammen mit dem Chiffrat des Meßwertes an den Aggregationsknoten. Anhand der Klassifizierung entscheiden Aggregationsknoten, ob ein Meßwert weiter in Richtung Datensenke transportiert wird oder nicht. Ähnlich zu PHs muß der Meßwert dabei nicht entschlüsselt werden. Für diesen Ansatz gelten die gleichen Probleme wie für PHs, komplexere Aggregation ist damit nicht möglich, sondern nur Weiterleitung oder Verwerfung. Der Vorschlag scheitert außerdem an einem gravierenden Sicherheitsproblem: Ein Angreifer kann die unverschlüsselte Klassifizierunginformation eines Meßwertes beliebig abändern. Damit widerspricht dieser Ansatz den Entwurfsziele *Funktionalität* und *Sicherheit*

Ähnlich klassifiziert das Protokoll ESDPA aus Çam et al. [50, 51] alle möglichen Meßwerte von Sensoren in Intervalle, Pattern (Muster) genannt. Die Sensoren übertragen anstatt der Meßwerte nur die Nummer des zugehörigen Patterns an den Aggregationsknoten. Dieser streicht aus den empfangenen Mustern diejenigen mit redundanter Information heraus und erlaubt dann den Sensoren, die Meßwerte hohen Informationsgehalts gemessen haben, ihr Muster chiffriert an die Senke zu schicken. Obwohl die Autoren auch hier von einer Aggregation sprechen, unterscheidet sich diese Form der Klassifizierung doch sehr von den angenommenen Aggregationsfunktionen aus Kapitel 2. Für diese beliebig berechenbaren Funktionen eignet sich der Vorschlag nicht, ein Widerspruch zum Entwurfsziel *Funktionalität*. Selbiges gilt auch für Varianten von ESDPA, SRDA und SDDA, siehe Çam et al. [52], Sanli et al. [200], bei denen von den Sensorknoten anstatt der Muster nur die Differenzen neuer Meßwerte zu vorher vom Aggregationsknoten festgelegten Referenzwerten übertragen werden. Jedoch sind auch hier komplexere Aggregationen nach Kapitel 2 gar nicht möglich.

In dem Verfahren aus Zhu et al. [259] müssen *mehrere* Knoten denselben Meßwert eines Knotens a bezeugen und an den Aggregationsknoten schicken. Erst wenn die Mehrzahl der Zeugen oder eine bestimmte minimale Anzahl aller Knoten einen Meßwert bezeugt, schickt der Aggregationsknoten diesen Meßwert weiter in Richtung Senke. Komplexere Aggregation, das Vorverarbeiten von Daten wie in *Funktionalität* gefordert, findet hier demnach nicht statt.

4.2.4 Weitere Arbeiten

Nun folgen noch weitere Veröffentlichungen, die von ihrer Idee eher „Mischformen“ sind und sich daher in kein bestimmtes Schema kategorisieren lassen. Das soll jedoch nichts über ihre Relevanz aussagen.

Reine Hop-zu-Hop- oder Nachrichten-Authentizität

In einigen Arbeiten, zum Beispiel Bohge und Trappe [38], Deng et al. [71], Dimitriou und Foteinakis [77], Zhu et al. [257, 258], verschlüsseln messende Sensoren a, b jeweils ihre Werte und schicken sie chiffriert an den Aggregationsknoten c. Damit sind die Meßwerte vertraulich, integer und auch authentisch – allerdings auch nur ausschließlich zwischen a beziehungsweise b und c.

Das reicht jedoch nicht aus: Ein korrumpierter Aggregationsknoten c kann auf diese Weise unbemerkt falsche Aggregate bilden. Authentizität über Aggregationsknoten hinweg ist nicht möglich. Genau hier liegt aber die Herausforderung aggregierenden Datentransports. Diese Arbeiten widersprechen demnach *Funktionalität* beziehungsweise *Sicherheit*, sichere Aggregation ist mit ihnen nicht möglich.

In Vogt [227] sichern die Autoren nur die Authentizität von Nachrichten ab, ohne Aggregation zu betrachten. Knoten a will eine Nachricht authentisch an Knoten b senden. Es existiert dabei jedoch kein gemeinsamer Schlüssel zwischen beiden und die Nachricht wird über mehrere Hops $S_1, S_2, \ldots, S_n$ von a nach b geleitet. Falls a Schlüssel mit S_1 und S_2 besitzt, dann kann a die Nachricht jeweils für S_1 und S_2 verschlüsseln und an S_1 und S_2 schicken. Knoten S_1 entschlüsselt die Nachricht und sendet sie verschlüsselt an S_2, S_3 usw. Durch diese Art „Multi-Path“-Kommunikation wird sichergestellt, daß in Gegenwart eines korrumpierten Knotens die Authentizität der Nachricht beim Empfänger b garantiert ist. Das Verfahren läßt sich allerdings nicht für *aggregierenden* Datentransport übernehmen, bei dem Daten nicht nur zur Senke weitergeleitet, sondern darüberhinaus auch noch vorverarbeitet werden. Weiterhin ist dieses Verfahren *unsicher* gegen mehrere korrumpierte Knoten.

Secure Aggregation for Wireless Networks

Das Protokoll SAWN, siehe Hu und Evans [107], schützt die Datenaggregation in zwei Schritten. Im ersten Schritt senden zwei Sensorknoten a und b ihre Meßwerte x und y zusammen mit MACs an ihren Aggregationsknoten c. Die MACs berechnen sie dabei aus temporären Schlüsseln $K^t_{a,B}$ und $K^t_{b,B}$, die a beziehungsweise b paarweise mit der Basisstation B gemein haben. Die Knoten a und b schicken $(x, \mathrm{MAC}(K^t_{a,B}, x))$ und $(y, \mathrm{MAC}(K^t_{b,B}, y))$ an c. Obwohl Aggregationsknoten c die Authentizität von x und y nicht überprüfen kann, aggregiert er x und y mit einer beliebigen Aggregationsfunktion zu $F(x, y)$ und sendet

$$F(x,y), \mathrm{MAC}(K^t_{c,B}, F(x,y)), x, \mathrm{MAC}(K^t_{a,B}, x), y, \mathrm{MAC}(K^t_{b,B}, y)$$

an den nächst höheren Aggregationsknoten. Der erste der beiden Schritte von SAWN ist damit beendet. Weder der Aggregationsknoten c noch dessen Vorgänger können die Authentizität von x und y und auch nicht die Korrektheit der Aggregation F überprüfen.

Dies geschieht erst im zweiten Schritt des Verfahrens. Die Basisstation flutet per zuverlässigem Broadcast alle bisher geheimen temporären Schlüssel in das Netz, so wie beispielsweise $K^t_{a,B}$ oder $K^t_{b,B}$. Zuverlässiger Broadcast bedeutet hier, daß die Sendereichweite der Senke so hoch ist, daß alle Knoten im Netz die temporären Schlüssel empfangen können. Damit können jetzt sowohl c als auch sein Vorgänger die Authentizität von x und y überprüfen. Der Vorgänger von c ist außerdem in der Lage, die Korrektheit von F zu verifizieren, da er x und y kennt. Abschließend werden die Schlüssel K^t von der Basisstation noch mit Hilfe des Protokolls μTesla [172–174] durch neue temporäre Schlüssel $K^{t'}$ ersetzt. Dadurch, daß diese Arbeit auf μTESLA basiert, einem Protokoll mit häufiger Verwendung einer zentralen Basisstation, widerspricht sie also der geforderten *Dezentralität*.

Der gravierende Nachteil dieser Arbeit liegt in der Art und Weise, in der Hu und Evans [107] authentisch aggregiert, nämlich verzögert. Knoten können empfangene Aggregate immer erst im zweiten Schritt des Protokolls, zeitlich verzögert zum Empfang, überprüfen und sind dabei auf das Fluten der K^t-Schlüssel mittels zuverlässigem Broadcast von der Basisstation angewiesen. Einem Angreifer nach Abschnitt 2.4, S. 24 kann es durch Abschirmen von Knoten oder Funkstörungen gelingen, das Ausliefern der Aggregate und MACs im ersten Schritt des Protokolls zu verhindern oder zu verändern. Sobald die Basisstation dann die Schlüssel K^t offenlegt, ist der Angreifer in der Lage, an „falsche“ Meßwerte x', y' angepaßte MACs selbst zu erstellen, die geänderten Meßwerte und MACs an die zuvor abgeschirmten Knoten zu versenden und schließlich auch die von der Basisstation empfangenen K^t's weiterzugeben. Das Protokoll kann danach wie gewohnt weiterlaufen, der Betrug fällt nicht auf.

Dieses Protokoll außerdem nur in Gegenwart genau eines korrumpierten Knoten authentisch aggregieren. Im Sensornetz sind durchaus mehr als nur ein korrumpierter Knoten denkbar, zum Beispiel könnten in einem Protokolldurchlauf durchaus $k > 1$ korrumpierte Knoten teilnehmen. Sobald zwei im Aggregationsbaum aufeinanderfolgende Knoten korrumpiert sind, fällt ein modifiziertes Aggregat nicht mehr auf. Der Versand einzelner Meßwerte oder Aggregate erfolgt nicht chiffriert und kann von einem Angreifer abgehört werden. Damit erfüllt Hu und Evans [107] nicht das Entwurfsziel *Sicherheit*.

A Witness-Based Approach For Data Fusion Assurance In Wireless Sensor Networks

Das Protokoll aus Du et al. [82] kann auch mit mehreren korrumpierten Knoten umgehen. Für jeden Aggregationsknoten c existieren insgesamt m Knoten als Zeugen. Alle Zeugen besitzen jeweils paarweise verschiedene Schlüssel K_{m_i} mit der Senke. Jeder Sensorknoten a schickt nun eine Kopie seiner Meßwerte nicht nur an den Aggregationsknoten, sondern auch an alle m Zeugen. Diese Zeugen führen genauso wie c die Aggregationsfunktion aus und berechnen ein Aggregat A_{m_i}. Von diesem Aggregat bilden sie schließlich einen MAC, $\mathrm{MAC}_{m_i} = MAC(K_{m_i}, A_{m_i})$ und senden ihn an den Aggregationsknoten c. Knoten c sammelt die MACs sämtlicher Zeugen und schickt nun $(A_c, \mathrm{MAC}_c, \mathrm{MAC}_{m_1}, \mathrm{MAC}_{m_2}, \ldots)$ an die Datensenke. Dabei bezeichnet A_c das von c berechnete Aggregat und MAC_c den dazugehörigen MAC. Die Senke kann das Aggregat A_c schließlich überprüfen, in dem sie mit Hilfe von A_c und der einzelnen K_{m_i} die MAC_{m_i} generiert. Stimmen mindestens $(m - k)$ von m MACs mit MAC_c überein, geht

die Senke davon aus, daß A_c von c korrekt aggregiert worden ist, selbst in Gegenwart von k-korrumpierten Knoten.

Das Weiterleiten sämtlicher MACs bis an die Basisstation bedingt einen deutlichen zusätzlichen Overhead durch die Anzahl der dafür notwendigen Nachrichten und reduziert in jedem Fall die durch die Aggregation entstehenden Energieeinsparungen. Die Autoren untersuchen weiterhin nicht, welche Auswirkungen k auf die Energieeffizienz des gesamten Protokolls im Vergleich zu sicherem Datentransport ohne Aggregation hat. Dadurch, daß sämtliche der erzeugten MACs Multi-Hop durch das komplette Netz bis an die Basisstation gesendet werden müssen, wird dieses Protokoll selbst für kleine k energetisch teurer als sicherer Datentransport ohne Aggregation. Solch ein Mehraufwand steht im Gegensatz zur *Effizienz*.

Nur die Basisstation überprüft Nachrichten, was inhärent Unsicherheit bei kaskadierender Aggregation zur Folge hat: Keiner der Aggregationsknoten auf dem Pfad zur Senke hin kann die Aggregate verifizieren. Überdies ist es der Senke ausschließlich möglich, die Korrektheit der von c durchgeführten Aggregation zu überprüfen, die Authentizität der verursachenden Meßwerte beziehungsweise Quellsensoren kann nicht getestet werden. Das Protokoll widerspricht damit *Funktionalität*.

Auch Du et al. [82] verschlüsselt Meßwerte und Aggregate nicht. Ein Angreifer kommt so in die Lage, vertrauliche Information mitzulesen, ein Widerspruch zu *Sicherheit*.

Eine weitere wichtige Fragestellung, die Du et al. [82] allerdings offenhält, ist die nach der Auswahl von m Zeugen pro Aggregation. Auch dies ist im Sensornetz problematisch, da Aggregationen sich ändern können, Knoten können ausfallen (*Dynamik*). Die Auswahl solcher Zeugen muß darüberhinaus auch wieder sicher sein.

Resilient Data Aggregation

Wagner [234] untersucht theoretisch, welche Aggregationsfunktionen sich gegenüber korrumpierten Knoten, die falsche Meßwerte beitragen, als widerstandfähig erweisen. Widerstandsfähig bedeutet, daß die Ergebnisse der Aggregationsfunktion relativ invariant von den Meßwerten der korrumpierten Knoten sind. Die gefälschten Werte korrumpierter Knoten beeinflussen das gesamte Aggregat nicht sehr stark, das heißt nur im Rahmen eines vom Benutzer akzeptierten Fehlers.

In Buttyán et al. [44] wird ein Algorithmus vorgeschlagen, in dem gefälschte Werte aus der Aggregatbildung ausgefiltert werden können. Die Filterung basiert dabei auf zufälligem Überprüfen von Meßwerten aus Fischer und Bolles [93]. Die Arbeit Roy et al. [194] basiert auf der Idee, daß die Basisstation weiß, in welchem Wertebereich sich ein Aggregat befinden muß und daher extreme, möglicherweise von einem korrumpierten Knoten zur Aggregatbildung beigetragene Werte identifiziert.

Es interessiert an dieser Stelle nicht nur die Integrität der Daten, sondern vor allem ihre Authentizität über Aggregationsknoten hinweg. Das betrachten Buttyán et al. [44], Roy et al. [194], Wagner [234] nicht. Die in den Arbeiten untersuchten Aggregationsfunktionen beschränken sich außerdem nur auf einfache Funktionen wie zum Beispiel den Mittelwert, Median etc. und erfüllen damit nicht das Entwurfsziel *Funktionalität*.

Sicherheit durch Vertrauen

Ein gänzlich anderer Ansatz zur Authentizität liegt im Aufbauen von *Vertrauen* zu Sensoren. Die Arbeit Zhang et al. [252] basiert dabei auf der Idee, daß Aggregationsknoten die

von Sensorknoten empfangenen Meßwerte bewerten. Wenn mehrere Sensoren das gleiche Phänomen beobachten und an den Aggregationsknoten weiterleiten, bildet dieser einen Mittelwert davon. Je geringer der Unterschied der Meßwerte zu diesem Mittelwert desto mehr vertraut der Aggregationsknoten den Meßwerten und den dazugehörigen Knoten. Je mehr ein Aggregationsknoten a einem anderen Knoten x vertraut, desto „authentischer" stuft a den Meßwert von x ein und bezieht dies in seine Aggregatbildung mit ein. Aggregationsknoten a filtert Meßwerte von wenig authentischen Knoten aus. Dieser Grundsatz, Knoten über längere Zeiträume zu bewerten und darauf aufbauend Vertrauen und Sicherheit zu entwickelt basiert auf Ideen aus unter anderem Beth et al. [23], Covington et al. [64], Kamvar et al. [121], Reiter und Stubblebine [184].

Dieser Ansatz gelingt nur dann, wenn viele nicht-korrumpierte Knoten über einen langen Zeitraum hinweg und redundant das selbe Phänomen messen. Meldet ein einzelner Knoten einen Alarm, zum Beispiel der Blutdrucksensor einen kritischen Blutdruck, dann liegt dieser kritische Blutdruck unter Umständen soweit vom Mittelwert entfernt, daß er ignoriert wird. Außerdem lassen sich damit nur Phänomene aggregieren, über die man einen Mittelwert bilden kann. Soll der Gesundheitszustand eines Patienten sich aus mehreren unterschiedlichen Kategorien zusammensetzen, kann aus diesen Kategorien kein gemeinsamer Mittelwert gebildet werden. Ebenso unterstützt der vorgestellt Ansatz keine kaskadierende Aggregation. Dies entspricht nicht den geforderten, sehr allgemeinen Formen der Aggregation im Entwurfsziel *Funktionalität*. Falls weiterhin korrumpierte Knoten über einen sehr langen Zeitraum hinweg Vertrauen aufbauen, gelingt es Ihnen, die Mittelwerte vom tatsächlichen Mittelwert zu verschieben. Korrumpierte Knoten beeinflussen damit die Bildung eines Aggregats, was dem geforderten Entwurfsziel *Sicherheit* widerspricht.

Secure Hop-by-Hop Data Aggregation Protocol

In Yang et al. [250] schlagen die Autoren das Protokoll SDAP vor. Die grundsätzliche Idee von SDAP basiert auf der Fähigkeit der Basisstation, Anomalien in empfangenen Aggregaten zu entdecken. Sobald die Basisstation eine Anomalie entdeckt und bestimmte Sensoren oder Mengen von Sensoren als böse verdächtigt, müssen diese Knoten ihre zuvor gemessenen Daten und Aggregate direkt an die Basistation senden und damit die Aggregation nachträglich quasi aufheben.

SDAP garantiert damit keine Authentizität für jedes empfangene Aggregat, sondern Authentizität nur bei Aggregaten, die im Nachhinein von der Senke als „normal" klassifiziert werden. Dies widerspricht dem Entwurfsziel *Funktionalität*, das bei jedem empfangenen Aggregat dessen Authentizität sofort überprüfbar machen soll. Fälschungen können erst bei Vorliegen eines bestimmten Verdachtes überprüft werden. Inwieweit ein Verdacht, eine Anomalie, etwas „nicht-normales" in der Menge der empfangenen Aggregate, überhaupt erkannt werden kann, bleibt ein offenes Problem. Für alle Szenarien, in denen jedes Aggregat über eine hohe Entropie verfügt, beispielsweise, wenn der komplette Wertebereich eines empfangenen Datums aus Applikationssicht „Sinn macht", eignet sich SDAP damit nicht. In diesen Szenarien müßte jedes Aggregat nachträglich überprüft werden. Letzteres widerspricht der *Effizienz*.

4.2.5 Zusammenfassung

Tabelle 4.1 faßt den Stand der Forschung im Bereich authentischer Aggregation zusammen. Falls ein Verfahren eines der geforderten Entwurfsziele *nicht* erfüllt, so ist dies mit

Ansätze	Entwurfsziele: Funktionalität	Sicherheit	Dezentralität	Effizienz
Authentizität durch Homomorphismen [2, 49, 95, 162, 241, 242]	◯	◯	√	◯
Einschränkung der Aggregationsfunktion				
[118, 167]	◯	◯	√	√
[182]	◯	◯	√	◯
[180]	◯	√	◯	√
[59]	◯	√	◯	◯
Verfahren zur Klassifizierung von Daten				
[246]	◯	◯	√	√
[50–52, 200, 259]	◯	√	√	√
Diverse				
[38, 71, 77, 227, 257, 258]	◯	◯	√	√
[107]	√	◯	◯	√
[82]	◯	◯	√	◯
[44, 194, 234]	◯	√	√	√
[252]	◯	◯	√	√
[250]	◯	√	√	◯

Tabelle 4.1 Stand der Forschung *authentische Aggregation*

einem ◯ in der entsprechenden Zeile/Spalte vermerkt, im anderen Fall mit einem √. Die Tabelle läßt erkennen, daß viele der Arbeiten die Entwurfsziele *Funktionalität* und *Sicherheit* nicht erfüllen. Viele Arbeiten verstehen Aggregation weniger allgemein als dies in dieser Arbeit angenommen wird und liefern nur Lösungen für Spezialfälle von Aggregationen wie zum Beispiel Mittelwertbildung. Weiterhin können die meisten Arbeiten nicht mit dem angenommen Modell eines starken Angreifers, der Knoten korrumpiert, umgehen.

Es existiert bisher noch kein Ansatz für eine authentische Aggregation, welche die allgemeine, mathematische, kaskadierende Form von Aggregationsfunktionen unterstützt – in Gegenwart von korrumpierten Knoten. Dies leistet das im folgenden präsentierte Protokoll ESAWN.

4.3 Das Protokoll ESAWN

Wie eingangs bereits erwähnt, kann es effiziente Authentizität in aggregierenden Sensornetzen zusammen mit einem Sicherheit–Energie-Tradeoff geben. Die grundsätzliche Idee eines solchen Tradeoffs ist: Je „authentischer“ Daten im Netz aggregiert werden sollen, desto mehr Energie durch zusätzliche (Nachrichten-)Übertragungen muß dafür bezahlt werden. Umgekehrt, je effizienter Daten aggregiert werden sollen, desto unsicherer werden sie übertragen. Dieser Tradeoff zwischen Sicherheit und Energie kann zum Beispiel

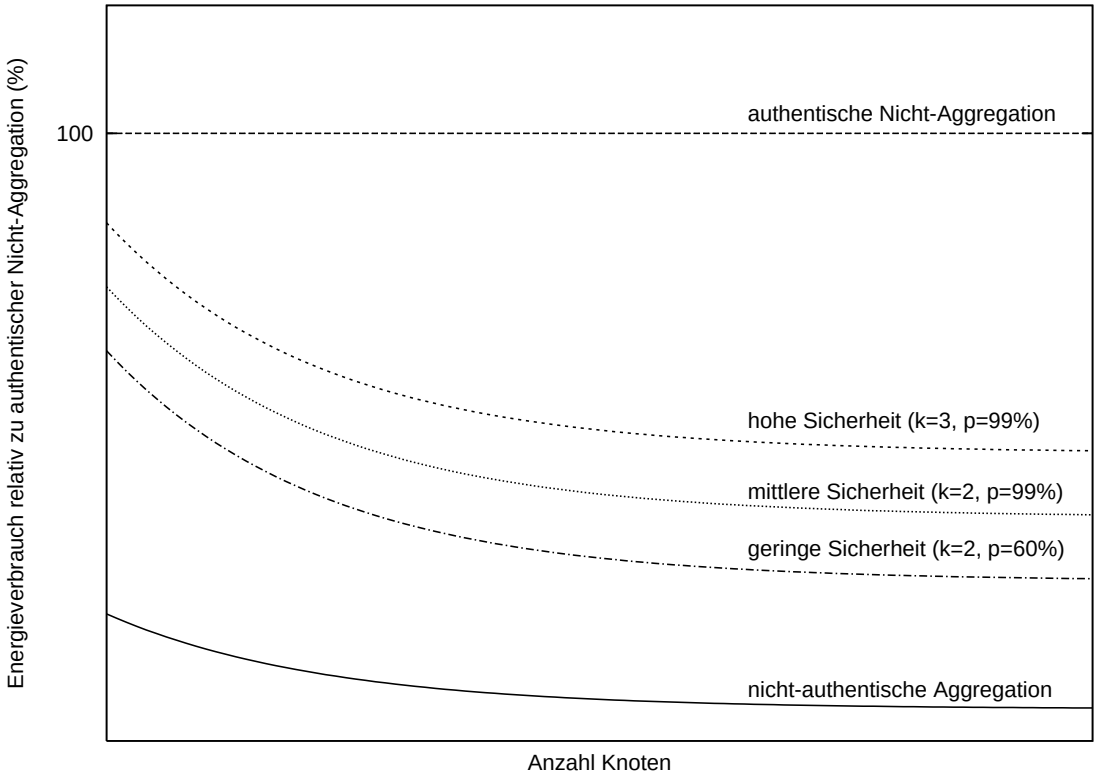


Abbildung 4.2 Graduelle Authentizität würde variablen Energieverbrauch bedeuten

vom Benutzer *parametrisiert* werden. Der Benutzer gibt vor, mit welcher Authentizität er wünscht, daß Daten im Sensornetz übertragen werden – im Bewußtsein, daß mehr Authentizität auch mehr Energie kostet.

Einen solchen Tradeoff skizziert nur zu Verständniszwecken Abbildung 4.2. Der Graph zeigt insgesamt drei Kurven, die den relativen Energieverbrauch eines authentisch aggregierenden Protokolls mit *authentischer Nicht-Aggregation* vergleichen. Authentische Nicht-Aggregation bezeichnet die Form des Datentransports, bei dem alle Sensoren ihre verschlüsselten Meßwerte durch das Netz an die Senke schicken. Es findet dabei keinerlei Aggregation auf Knoten statt. Da sämtliche Knoten, zum Beispiel mit Hilfe des Schlüsselaustauschprotokolls SKEY aus Kapitel 3, auch paarweise verschiedene Schlüssel mit der Senke besitzen, ist solch ein nicht-aggregierender Datentransport zwischen Knoten und Senke authentisch. Sein Energieverbrauch dient in Abbildung 4.2 als Referenzkurve: die oberste Kurve, waagerecht, bei 100% Energieverbrauch. Diese Referenzkurve gibt eine obere Grenze für den Energieverbrauch eines neuen sicheren aber aggregierenden Protokolls an: Überschreitet ein authentisch aggregierendes Protokoll diese Schwelle, lohnt sich sein Einsatz nicht mehr. Effizienter wäre dann der Einsatz authentischer Nicht-Aggregation.

Alle anderen Kurven im Graphen stehen in prozentualer Relation zum Energieverbrauch authentischer Nicht-Aggregation. Die unterste Kurve, bezeichnet mit *nicht-authentischer Aggregation*, soll die nicht-authentische und damit unsichere Aggregation darstellen. Es findet Aggregation im Netz statt, allerdings ohne irgendwelche Sicherheitsprotokolle. Bei dieser Form des Datentransports handelt es sich um ein Minium, das von keinem Sicherheitsprotokoll unterschritten werden kann: Kein authentisch aggregierendes Protokoll

kann *günstiger* Daten transportieren als ein unsicher aggregierendes. Sicherheit bedeutet immer eine Form von Mehraufwand im Netz.

Ein in Bezug auf Sicherheit-versus-Energie parametrisierbares Protokoll sollte aus Energiesicht zwischen den beiden abgebildeten Schwellenwerten liegen: So wie zum Beispiel die drei mittleren Kurven in Abbildung 4.2, bei denen hier nur der Verlauf zueinander von Bedeutung ist. Die drei Kurven unterscheiden sich in ihrer Sicherheit sowie damit auch dem Energieverbrauch. Die obere der drei Kurven in Abbildung 4.2 könnte beispielsweise Authentizität mit einer Wahrscheinlichkeit von $p = 99\%$ bei $k = 3$ korrumpierten, am Protokoll teilnehmenden Knoten zusichern. Eine Wahrscheinlichkeit von $p = 99\%$ Authentizität bedeutet beispielsweise, daß 99% aller Aggregate im Durchschnitt authentisch aggregiert werden, selbst wenn 3 korrumpierte Knoten unmittelbar an der Aggregation beteiligt sind. Die Konfiguration $k = 3, p = 99\%$ der oberen Kurve aggregiert Daten sicherer als die der mittleren, zweiten Kurve, bei der nur gegen $k = 2$ korrumpierte Knoten abgesichert werden kann – dafür ist diese Konfiguration aber aus Energiesicht auch deutlich günstiger. Noch „günstiger“ aggregiert die untere Kurve, ermöglicht aber auch nur Authentizität in $p = 60\%$ aller Fälle bei 2 korrumpierten Knoten.

Dieses Beispiel verdeutlicht, worauf das im Rahmen dieser Arbeit entwickelte Protokoll ESAWN, *Extended Secure Aggregation for Wireless sensor Networks*, siehe Blaß et al. [31], zielt: Da Authentizität im Sensornetz widersprüchlich zur Aggregation steht und damit einiges an Energiemehraufwand bedeutet, ermöglicht ESAWN dem Benutzer, bei unterschiedlichen Applikationen, Sicherheitsanforderungen, zur Verfügung stehenden Hardware-Ressourcen, Batteriekapazität und korrumpierten Knoten *graduell abstufbar* Sicherheit und Energie zu parametrisieren.

Zusammengefaßt zeichnet sich ESAWN durch die folgenden Eigenschaften aus:

- ESAWN erfüllt sämtliche der in Abschnitt 4.1.2 aufgestellen Entwurfsziele.
- ESAWN ist sicher gegen eine vom Benutzer parametrisierbare Anzahl byzantinischer Knoten. Es sichert Authentizität mit einer ebenso vom Benutzer bestimmbaren Wahrscheinlichkeit zu. Auf diese Weise läßt sich ein Tradeoff zwischen Sicherheit auf der einen und Energieverbrauch auf der anderen Seite finden.
- der Energieverbrauch von ESAWN ist dabei pro Knoten unabhängig von der Gesamtanzahl n aller Knoten im Sensornetz. Er skaliert mit $O(1)$. Ebenso skaliert der Speicheraufwand von ESAWN pro Knoten mit $O(1)$.

Weiterer Aufbau dieses Kapitels

Die folgenden Abschnitte erklären die Funktionsweise von ESAWN. Zunächst präsentiert Abschnitt 4.3.1 grob das zugrundeliegende Konzept hinter dem Verfahren. Abschnitt 4.3.3 beschreibt und erklärt dann das Protokoll sehr ausführlich: Abschnitt 4.3.3.1 erklärt dabei die Auswahl von k sogenannten *Zeugen*, Abschnitt 4.3.3.2 beschreibt darauf aufbauend den Ablauf eines Verifikationsschrittes im Detail und Abschnitt 4.3.3.4 stellt die probabilistische Verifikation zum Energieeinsparen vor. An den entsprechenden Stellen wird außerdem auf den Pseudocode von ESAWN eingegangen.

In Abschnitt 4.3.4 wird abschließend diskutiert, warum überhaupt die vorgestellte Lösung ESAWN selbst gegen k Angreifer sicher arbeitet. Die Erklärungen zu ESAWNs Sicherheit bleiben an dieser Stelle jedoch eher rudimentär beziehungsweise ungenau und dienen in erster Linie nur dem grundsätzlichen Verständnis der Sicherheit. Der etwas komplexere *formale Beweis* der Sicherheit folgt erst in Abschnitt 4.3.8.

4.3.1 Protokollidee

Als grundsätzliches Ziel soll ESAWN verhindern, daß korrumpierte Knoten unbemerkt Aggregate oder die Authentizität der zu den Aggregaten beitragenden, messenden Knoten fälschen können. Fälschen korrumpierte Knoten Aggregate, so sollen die nicht-korrumpierten („legitimen“) Knoten diesen „Betrug“ feststellen. Genauer: Kein legitimer Knoten soll ein gefälschtes Aggregat weiterverarbeiten, ohne die Fälschung zu bemerken. Könnten korrumpierte Knoten mindestens einem legitimen Knoten, wie beispielsweise der Senke, unbemerkt gefälschte Aggregate zusenden, würden die legitimen Knoten ihrerseits falsche Schlüsse daraus ziehen und falsch reagieren. Aktoren würden auf gefälschten Meßwerten basierend falsche, unter Umständen gefährliche Aktionen auf ihre Umgebung ausführen.

ESAWN setzt voraus, daß alle Knoten mit allen Knoten ihres Aggregationspfades über paarweise verschiedene Schlüssel verfügen, effizient ausgetauscht zum Beispiel mit Hilfe des Protokolls SKEY aus Kapitel 3. Durch den Besitz dieser Schlüssel können jeweils zwei Knoten sicher, das heißt vertraulich, integer und authentisch Daten austauschen. (Im folgenden seien dann auch alle Datenübertragungen zwischen zwei Knoten implizit mit deren gemeinsamen Schlüssel chiffriert, auch wenn das in den einzelnen Protokollbeschreibungen der Übersichtlichkeit halber nicht immer explizit genannt wird.)

Die grundsätzliche Idee hinter ESAWN liegt in der Überprüfung sämtlicher Aggregationen im Netz durch jeweils k verschiedene Knoten, den sogenannten *Zeugen*. Jede Aggregation wird unabhängig von jeder anderen geprüft. Damit soll es dem Angreifer selbst bei jeweils k an einer Verifikation teilnehmenden, korrumpierten Knoten nicht gelingen, eine Aggregation unbemerkt vom Rest des Netzes zu fälschen. Nach Abschnitt 2.4, S. 24 gelingt dem Angreifer damit eine Fälschung selbst bei insgesamt $\beta\%$ korrumpierten Knoten im Netz nicht. Durch eine geschickte Wahl der Zeugen sowie probabilistisch relaxierten Überprüfungen kann der Energieverbrauch von ESAWN dabei vom Benutzer auf ein gewünschtes Maß gedrosselt werden.

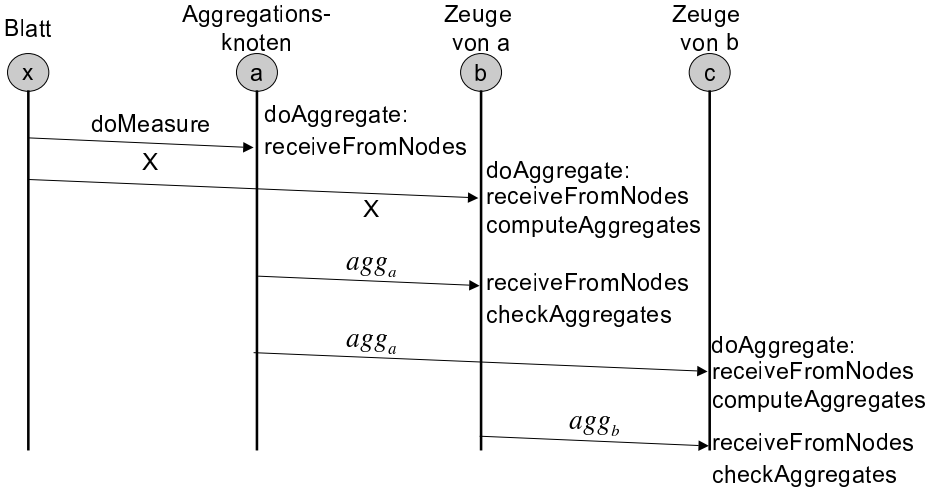


Abbildung 4.3 Veranschaulichung des Protokollablaufs

Den groben Protokollablauf von ESAWN skizziert Abbildung 4.3 für $k = 1$. Blatt x mißt ein Datum und schickt dieses an Aggregationsknoten a. Knoten a aggregiert dieses Datum und schickt das Aggregat an a's Vorgänger im Aggregationsbaum, das ist der Knoten b. Knoten b ist ein *Zeuge* von a's Aggregation und überprüft die Authentizität des empfangenen Aggregats.

Alle Blätter im Aggregationsbaum führen die Funktion $\mathrm{doMeasure}$ aus. Alle anderen Knoten, die Aggregationsknoten, führen die Funktion $\mathrm{doAggregate}$ aus. Die einzelnen Schritte von ESAWN sind:

1. Im Rahmen von $\mathrm{doMeasure}$ mißt Blatt x ein Datum X und schickt X sowohl an a als auch an den Zeugen von a's Aggregation, den Knoten b.

2. Aggregationsknoten a führt $\mathrm{doAggregate}$ aus.

   Im Rahmen von $\mathrm{doAggregate}$ empfängt a zunächst das Datum X seines direkten Nachfolgers im Aggregationsbaum (x), das er aggregieren soll. Zum Empfangeden von X dient die Funktion $\mathrm{receiveFromNodes}$.

3. Knoten a aggregiert X zu seinem Aggregat agg_a. Er schickt agg_a an seinen direkten Vorgänger im Aggregationsbaum b sowie an den Zeugen von b's Aggregation, den Knoten c.

4. Auch Aggregationsknoten b führt $\mathrm{doAggregate}$ aus. Im Gegensatz zu a ist b aber nicht nur Aggregationsknoten, sondern auch noch Zeuge einer Aggregation. Knoten b soll die Aggregation von a überprüfen.

   Knoten b erhält von x das gemessene Datum X mit $\mathrm{receiveFromNodes}$. Mit Hilfe von $\mathrm{computeAggregates}$ berechnet b aus X ein Aggregat agg'_a.

5. Nachdem b von a das berechnete Aggregat agg_a mit $\mathrm{receiveFromNodes}$ empfängt, verifiziert er, ob a korrekt in Bezug auf X aggregiert hat. Das heißt, er überprüft, ob $\mathrm{agg}_a \stackrel{?}{=} \mathrm{agg}'_a$ gilt. Dazu dient $\mathrm{checkAggregates}$.

   Damit ist die Verifikation der Authentizität abgeschlossen. Falls a korrumpiert ist und falsch aggregiert, kann b dies feststellen.

6. Das Protokoll ESAWN würde jetzt fortfahren mit der Verifikation der Aggregation des Knotens b.

   Knoten b berechnet sein eigenes Aggregat agg_b und schickt dieses an seinen direkten Vorgänger im Aggregationsbaum, den Knoten c. Dieser kann agg_b mit Hilfe des von a empfangenen agg_a verifizieren usw.

Anmerkung zur Notation

In Analogie zu Kapitel 3 bezeichnet k auch hier die Anzahl der korrumpierten Knoten hintereinander, die an einem Durchlauf des Protokolls teilnehmen. Bei SKEY waren das die korrumpierten Knoten bei einem Schlüsselaustausch, die hintereinander entlang eines Austauschpfades lagen. Bei ESAWN sind es die korrumpierten Knoten, die an einer *Verifikation* teilnehmen. Im weiteren Verlauf dieses Kapitels wird klar, daß bei ESAWN genau wie bei SKEY die Wahl von k direkt von der Höhe des Aggregationsbaums abhängt und auch auf ähnliche Weise bestimmt wird. Der Benutzer kann daher mit Hilfe des selben k sowohl SKEY als auch ESAWN parametrisieren – muß es aber nicht: Die Evaluierung wird zeigen, daß es sich aus Effizienzgründen durchaus lohnt, für den Schlüsselaustausch ein anderes, kleineres k zu bestimmen als für die anschließende Überprüfung von Aggregationen.

4.3.2 Erläuterungen zum Pseudocode

Da im folgenden Abschnitt ESAWN nicht nur erklärt, sondern auch auf dessen Pseudocode eingegangen wird, sind noch einige Hinweise zum Pseudocode notwendig.

Wie aus dem groben Ablauf des Protokolls ersichtlich, läßt sich der Pseudocode in zwei Algorithmen unterteilen. Algorithmus $\mathrm{doMeasure}$, siehe Algorithmus 11, ist der Algorithmus, den ein Blatt im Aggregationsbaum ausführt, wenn es einen Meßwert an seinen Aggregationsknoten sowie die Zeugen versendet. Algorithmus $\mathrm{doAggregate}$, siehe Algorithmus 12, ist der Algorithmus, den ein Aggregationsknoten ausführt. Der Algorithmus $\mathrm{doAggregate}$ unterscheidet sich in sofern von $\mathrm{doMeasure}$, da hier auch noch zusätzlich die Funktionalität zur Überprüfung von Aggregaten enthalten ist, für die ein Aggregationsknoten als Zeuge ausgewählt worden ist.

Im Pseudocode der Algorithmen bezeichnet:

- die Schreibweise $a \rightarrow b : C$, daß der Knoten mit der ID a eine Nachricht an den Knoten mit der ID b *sendet*. Inhalt dieser Nachricht ist C.
- die Schreibweise $a \leftarrow b : C$, daß der Knoten mit der ID a eine Nachricht vom Knoten mit der ID b *empfängt*. Inhalt dieser Nachricht ist C.

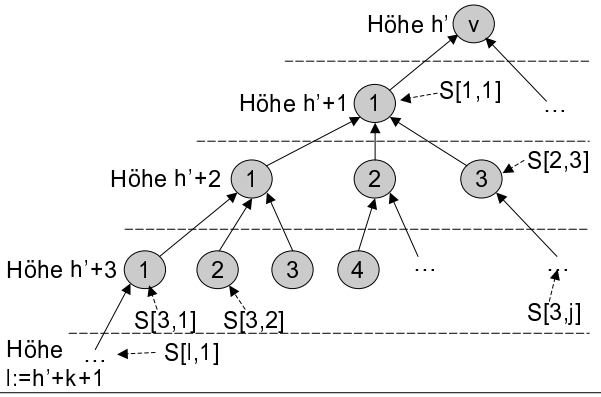


Abbildung 4.4 Nachfolger von v im Aggregationsbaum: $\mathbb{S}$ aus Algorithmus 12

- Π das Array mit den IDs der Vorgänger eines Knotens x im Aggregationsbaum. So bezeichnet zum Beispiel $\Pi[1]$ den direkten Vorgänger von x im Baum. Das Π eines Knotens x *im Pseudocode* entspricht damit trivialerweise nur den Vorgängerknoten aus dem im Grundlagenkapitel definierten *Aggregationspfad* $\mathbb{P}$, $\Pi \subseteq \mathbb{P}$.
- $\mathbb{S}$ das zwei-dimensionale Array eines Knotens v, in dem alle Knoten-IDs der Nachfolger, die *Successor*, von v im Aggregationsbaum enthalten sind. Dabei steht $\mathbb{S}[i][j]$ für den $j$ten-Nachfolger auf Höhe i unterhalb von v. Dies stellt Abbildung 4.4 graphisch dar. So steht die ID des 2ten Knotens, der sich 3 Ebenen unterhalb von v im Aggregationsbaum befindet, in $\mathbb{S}[3][2]$. Auch für $\mathbb{S}$ gilt $\mathbb{S} \subseteq \mathbb{P}$.
- SEEDS das Array mit den *Seeds* für die Überprüfungswahrscheinlichkeiten einzelner Knoten. So steht in $\mathrm{SEEDS}[v]$ zum Beispiel der Seed für Knoten v. Anhand dieser Seeds kann jeder Knoten mit Hilfe eines Pseudozufallszahlengenerators entscheiden, ob die aktuelle Aggregation eines anderen Knotens, für den er Zeuge ist, überprüft werden soll oder nicht.

  Einzelne Seeds repräsentieren den internen Zustand von Pseudozufallszahlengeneratoren, siehe Kelsey et al. [125].

- $\mathrm{updateSeeds(SEEDS)}$ die Funktion, mit der die einzelnen Seeds nach dem Ziehen der Pseudozufallszahlen verändert werden. Wie diese Funktion genau implementiert wird, ist nicht Teil dieser Arbeit. Im Rahmen der Evaluierung mit Hilfe eines diskreten Ereignissimulators wurde der Einfachheit halber die standard C-Implementierung eines Pseudozufallszahlengenerators verwendet. Für Anforderungen an die $\mathrm{updateSeeds(SEEDS)}$ Funktion siehe aber zum Beispiel Kelsey et al. [124], NIST [166].

- p die vom Benutzer gewählte, durchschnittliche Überprüfungswahrscheinlichkeit, mit der einzelne Aggregationen jeweils verifiziert werden.

- k die maximale Anzahl der korrumpierten Knoten, die auf einem Aggregationspfad hintereinanderliegen.

- $\mathrm{sourceNodesOf(x)}$ eine Funktion, welche die „Eingabeknoten“ eines Knotens x ausgibt. Die „Eingabeknoten“ von x sind seine direkten Nachfolger im Aggregationsbaum, deren Meßwerte beziehungsweise Aggregate zur Bildung von x's Aggregat verwendet werden. Im Beispiel aus Abbildung 4.4 sind $\mathrm{sourceNodesOf}(\mathbb{S}[2][1])$ die Knoten $\{\mathbb{S}[3][1], \mathbb{S}[3][2], \mathbb{S}[3][3]\}$. Die Funktion gibt dabei allerdings nicht die Knoten IDs aus, sondern die kleinste und größte laufende Nummer der Eingabeknoten auf ihrer Höhe im Baum. Im Beispiel würde $\mathrm{sourceNodesOf}(\mathbb{S}[2][1])$ die Zahlen $\{1, 3\}$ ausgeben, da der ersten bis zum dritten Knoten auf Ebene 3 die Eingabeknoten von $\mathbb{S}[2][1]$ sind.

- $\mathrm{predecessorOf(x)}$ die ID des Vorgängers von Knoten x im Aggregationsbaum – das ist der Aggregationsknoten von x.

- $\mathrm{timeout}$ eine Bedingung, die wahr wird, falls beim Warten auf den Empfang von Nachrichten innerhalb einer vorgegebenen Zeitspanne keine Nachricht empfangen wird. Das könnte darauf hinweisen, daß der Angreifer versucht hat, bestimmte Nachrichten zu blockieren.

- error beziehungsweise **STOP_ALL_WORK** die Fehlerfunkion oder den Fehlerzustand, der erreicht wird, wenn ein Knoten einen Betrug eines korrumpierten Knoten erkannt hat oder eine ihn Nachricht, die er erwartet, nicht erreicht. Dies ist bereits im letzten Kapitel diskutiert worden und wird am Ende dieses Kapitels nocheinmal aufgegriffen.

4.3.3 Protokollbeschreibung

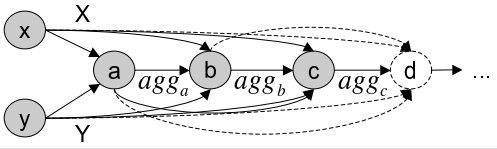


Abbildung 4.5 Idee der Aggregationsüberprüfung durch Zeugen

Ein einfaches Beispiel zum Prinzip von ESAWN zeigt Abbildung 4.5. Knoten a aggregiert die Daten X und Y der Knoten x und y. Diese Aggregation soll bei $k = 1, 2, 3, \ldots$ am Protokoll teilnehmenden, korrumpierten Knoten überprüft werden.

Dafür sind die Knoten $b, c, d, \ldots$ als *Zeugen* ausgewählt worden. ESAWN wählt dabei immer die k Vorgänger Π von a im Aggregationsbaum als Zeugen aus. Die Knoten x

Eingabe: int k, int SEEDS[], int p, ID Π[]
1: **Declaration:** Data myData, int p', ID predecessor
2:
3: **while true do**
4: {v beobachtet oder mißt seine Umgebung:}
5: myData := measure()
6:
7: {v schickt Meßwert an seinen direkten Vorgänger (= $\Pi[1]$):}
8: predecessor := $\Pi[1]$
9: $v \rightarrow$ predecessor : $E_{v,\text{predecessor}}(\text{myData})$
10:
11: {Falls das Aggregat von predecessor verifiziert werden soll, schickt v Kopien seines Meßwertes an die k Zeugen $\Pi[2 \ldots k+1]$:}
12: $p' := \text{generateRandomNumber}(1 \ldots 100, \text{SEEDS}[\text{predecessor}])$
13: **if** $p' \leq p$ **then**
14: **for** $i := 1$ to k **do**
15: $v \rightarrow \Pi[i+1] : E_{v,\Pi[i+1]}(\text{myData})$
16: **end for**
17: **end if**
18:
19: {v aktualisiert die Seeds:}
20: updateSeeds(SEEDS)
21: **end while**

Algorithmus 11 v.doMeasure(k, SEEDS, p, Π)

und y schicken jedes gemessene Datum nicht nur an a, siehe Algorithmus 11, Zeile 9, sondern auch an b, bei $k = 1$, und an c, d usw. bei $k = 2, 3, \ldots$. Analog versendet a sein gebildetes Aggregat agg_a nicht nur an b bei $k = 1$, sondern zusätzlich auch an c, d usw. bei $k = 2, 3, \ldots$. Dies zeigen die Zeilen 14ff. Die Zeugen sind dann in der Lage, genau wie a ein Aggregat aus den Quellwerten der Knoten x und y zu berechnen und mit dem Aggregat von a zu vergleichen, siehe doAggregate Algorithmus 12.

Falls ein Unterschied in der Aggregation besteht, ist falsch aggregiert worden, ein Betrug liegt vor, und die Zeugen leiten entsprechende Maßnahmen, wie zum Beispiel das *Alarmieren* ihrer Vorgängerknoten im Aggregationsbaum einschließlich der Senke, ein, siehe Algorithmus 14, Zeilen 8ff. Die Zeugen können weiterhin die Authentizität der Quellsensoren überprüfen, da sie laut Voraussetzung über paarweise Schlüssel mit ihnen Verfügen. Jedes empfangene Chiffrat kann damit auf Authentizität überprüft werden.

Kurz: Die Sensoren x und y schicken ihre Meßwerte nicht nur an a zur Aggregation, sondern zur Verifikation auch an die k Vorgänger von a im Aggregationsbaum. Zur Verifikation der Sicherheit wird damit die Aggregation auf k „Stufen“ im Aggregationsbaum aufgehoben: x und y schicken ihre Daten direkt an alle $k+1$ Vorgänger im Aggregationsbaum.

Selbst bei dieser sehr oberflächlichen ESAWN-Beschreibung wird klar, daß der dafür notwendige Energieaufwand durch das Verschicken von Nachrichten mit k steigt und damit das Verfahren bei großen k sehr teuer ist. Aus diesem Grund werden die Verifikationen stochastisch verteilt durchgeführt, so daß ein Angreifer Authentizität nur mit einer bestimmten Wahrscheinlichkeit erfolgreich fälschen kann.

Eingabe: int k, int SEEDS[], int p, ID Π[], ID $\mathbb{S}$[][]

Declaration: int p', Data agg_a, AggStore[][], AggStore$'$[][]
while true do
  {Knoten v ist Zeuge all seiner Nachfolger im Aggregationsbaum, die auf den Höhen $1 \ldots k$ *unter* ihm liegen. Er bekommt dazu von den Knoten auf den Höhen $2 \ldots k+1$ *unter* ihm Daten sowie die Daten von seinen direkten Nachfolgern zum Aggregieren, insgesamt von allen Knoten auf den Ebenen $k+1$ *unter* ihm (vergleiche Abbildung 4.4):}
  $l := k + 1$

  {AggStore und AggStore$'$ speichern die gebildeten Aggregate und die empfangenen Aggregate zum späteren Vergleich. AggStore$[i, j]$ speichert die gebildeten Aggregate des j-ten Nachfolgers auf Höhe i *unterhalb* von v:}
  AggStore := new Array[][]; AggStore$'$:= new Array[][]

  {v empfängt jetzt Daten von Knoten auf Höhen $i := l \ldots 1$ unterhalb von v:}
  AggStore$[l]$:= receiveFromNodes$(l, p, \Pi, \mathbb{S})${s. Algorithmus 13}
  for $i := l - 1$ to 1 **do**
    {v berechnet die sich aus den Daten ergebenden Aggregate: }
    AggStore$[i]$:= computeAggregates(AggStore$[i+1], p, i+1, \mathbb{S})${s. Alg. 16}
    {v empfängt Daten von allen Nachfolgern auf den Höhen $l - 1 \ldots 2$ *unter* v:}
    AggStore$'[i]$:= receiveFromNodes$(i, p, \Pi, \mathbb{S})$
    {v verifiziert die Aggregate}
    checkAggregates(AggStore$[i]$, AggStore$'[i], i,$ SEEDS$, \Pi, \mathbb{S})${Algorithmus 14}
  end for

  {v berechnet sein eigenes Aggregat:}
  $\text{agg}_v := f_v(\text{AggStore}[1,1], \text{AggStore}[1,2], \ldots, \text{AggStore}[1, |\text{AggStore}[1]|])$
  {v schickt Aggregat an seinen direkten Vorgänger:}
  $v \rightarrow \Pi[1] : E_{v,\Pi[1]}(\text{agg}_v)$

  {Falls v's Aggregat verifiziert werden soll, schickt v Kopien davon an die k Zeugen $\Pi[2 \ldots k+1]$:}
  $p' :=$ generateRandomNumber$(1 \ldots 100,$ SEEDS$[v])$
  if $p' \leq p$ **then**
    for $i := 1$ to k **do**
      $v \rightarrow \Pi[i+1] : E_{v,\Pi[i+1]}(\text{agg}_v)$
    end for
  end if

  {v aktualisiert die Seeds:}
  updateSeeds(SEEDS)
end while

Algorithmus 12 v.doAggregate$(k,$ SEEDS$, p, \Pi, \mathbb{S})$

Eingabe: int i, p, ID $\Pi[]$, ID $\mathbb{S}[][]$

Declaration: Data AggStoreOut, ID predecessor, int p'
AggStoreOut := new Array $[|\mathbb{S}[i]|]$

{v empfängt Daten von allen j Knoten auf Höhe i *unterhalb* von v:}
for $j := 1$ to $\mathbb{S}[i]$ **do**
  {predecessor bezeichnet den Vorgänger von Knoten $\mathbb{S}[i][j]$ im Aggregationsbaum:}
  predecessor := predecessorOf($\mathbb{S}[i][j]$)

  {Wenn predecessor's Aggregation übeprüft werden soll...}
  $p' :=$ generateRandomNumer$(1 \ldots 100, \text{SEEDS}[\text{predecessor}])$
  if $p' \leq p$ **then**
    {...erwartet v Daten von dessen Nachfolgern:}
    $v \leftarrow \mathbb{S}[i, j] : E_{v,\mathbb{S}[i,j]}(\text{agg}_{\mathbb{S}[i,j]})$

    {Falls Daten ausbleiben, liegt unter Umständen ein Betrugsversuch vor:}
    if timeout then
      error($\Pi, \mathbb{S}[i, j]$**)**
    else
      {v speichert den empfangenen Wert:}
      $\text{AggStoreOut}[j] := \text{agg}_{\mathbb{S}[i,j]}$
    end if
  end if
end for
return AggStoreOut

Algorithmus 13 v.receiveFromNodes$(i, p, \Pi, \mathbb{S})$

Eingabe: Data AggStore[], AggStore′[], int j, int SEEDS[], ID Π, ID $\mathbb{S}[][]$

Declaration: int p'
{v überprüft die selbst berechneten Aggregate aus AggStore mit den empfangenen aus AggStore':}
for $i := 1$ to $|\text{AggStore}|$ **do**
  {Überprüfe Knoten nur dann, wenn er überhaupt überprüft werden soll:}
  $p' :=$ generateRandomNumber$(1 \ldots 100, \text{SEEDS}[\mathbb{S}[j, i]])$
  if $p' \leq p$ **then**
    {Falls Unterschied erkannt, dann liegt ein Betrug vor:}
    if $\text{AggStore}[i] \neq \text{AggStore}'[i]$ **then**
      error($\Pi, \mathbb{S}[j, i]$**)**
    end if
  end if
end for

Algorithmus 14 v.checkAggregates$(\text{AggStore}, \text{AggStore}', j, \text{SEEDS}, \Pi, \mathbb{S})$

Eingabe: ID j
1: **for** $i := 1$ to $|\Pi|$ **do**
2: $v \rightarrow \Pi[i] : E_{K_{v,\Pi[i]}}(\text{Cheating}, j)$
3: **end for**
4: **STOP_ALL_WORK**

Algorithmus 15 Ein mögliches $v.\text{error}(\Pi, j)$

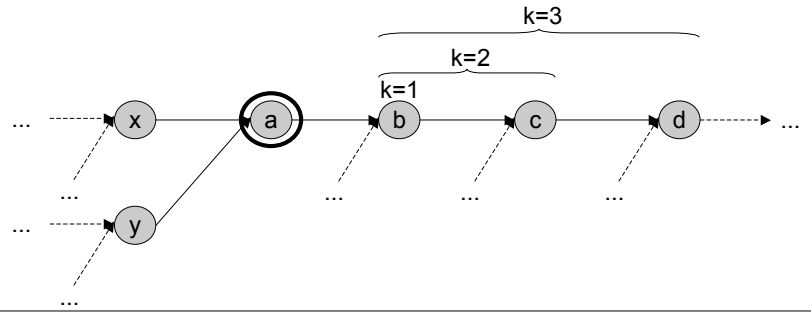


Abbildung 4.6 Auswahl von Zeugen in Abhängigkeit von k

4.3.3.1 Auswahl von k Zeugen

Bei Annahme von jeweils k an einer Überprüfung involvierten, korrumpierten Knoten müssen k Zeugen die Authentizität der Quellsensoren und Korrektheit der Aggregation bestätigen. In Abschnitt 2.4, S. 24 ist bereits erklärt worden, daß ein Angreifer genau wie der Benutzer über keinerlei Informationen bzgl. der konkreten Netzkonfiguration, des Netzaufbaus und der Aggregationsbeziehungen besitzt. Daher wird er nicht in der Lage sein, gezielt ganz bestimmte Knoten zu korrumpieren, sondern nur genau die, zu denen er zufällig zum Beispiel aufgrund ihrer physikalischen Lage einfachen Zugang hat. Bei angenommener Gleichverteilung aller korrumpierten Knoten sind auch die an einem Protokolldurchlauf teilnehmenden k korrumpierten Knoten immer gleichverteilt. Die Auswahl der k Zeugen kann damit aus Sicherheitssicht beliebig festgelegt werden. ESAWN wählt daher als k Zeugen einer Aggregation immer die k Vorgänger $\Pi[1 \ldots k]$ des Aggregationsknotens aus. Falls zur Überprüfung einer Aggregation nur $k' < k$ mögliche Knoten als Zeugen zur Verfügung stehen, zum Beispiel wenn ESAWN eine Aggregation sehr weit „oben“ im Aggregationsbaum überprüft, wählt ESAWN auch nur k' viele Zeugen aus. Dies hat keinerlei negative Konsequenzen für die Sicherheit, da der k'-te Zeuge immer durch die nicht-korrumpierte Senke repräsentiert wird – genauere Details zur Sicherheit folgen später.

Abbildung 4.6 zeigt den Ausschnitt eines Aggregationspfades. Die Aggregation kaskadiert hier von links nach rechts. Das bedeutet: Das Aggregat, das a mit Hilfe irgendeiner Aggregationsfunktion f_a aus den Meßwerten der Knoten x und y bildet, dient b als Teil einer Eingabe für seine Aggregatberechnung. Bei ESAWN dienen die k direkten Vorgänger von a im Aggregationsbaum als Zeugen. Bei $k = 1$ ist ausschließlich b Zeuge der Aggregation, bei $k = 2$ sind b und c Zeugen, bei $k = 3$ schließlich b, c und d usw.

Die Tatsache, daß die Knoten $b, c, \ldots$ als Zeugen der Aggregation des Knotens a ausgewählt werden, muß sowohl x, y und a als auch $b, c, \ldots$ zunächst einmal auf sichere Art und Weise mitgeteilt werden – eine Herausforderung aus Sicherheitssicht. Da sich die Aggregationsbeziehungen untereinander von Zeit zu Zeit ändern können, neue Knoten dem Netz beitreten und alte das Netz verlassen, unterliegt auch die Zeugenzuteilung einer Dynamik. Allerdings gibt es eine relativ einfache Lösung des Problem: Die Zuordnung und

Konfiguration des Aggregationsbaumes wird von ESAWN aus als von einer bestimmten Software-Komponente als gegeben betrachtet. Diese Komponente teilt dynamisch allen Knoten bei Änderungen an Aggregationsbeziehungen ihre Vorgänger und Nachfolger im Aggregationsbaum mit. Ein Beispiel für eine solche Software-Komponente könnte das in Abschnitt 3.3.7 entworfene Verfahren, ein Teil von SKEY, sein: Sobald ein neuer Knoten das Netz betritt, konstruiert er sich mit Hilfe seiner initialen Zufallsknoten seinen künftigen Aggregationspfad $\mathbb{P}$ im Netz auf sichere Weise. Damit stehen dann auch automatisch die Zeugen fest. Umgekehrt merken alle im Netz befindlichen Knoten, daß ein neuer Knoten nun Vorgänger auf ihrem Pfad geworden ist, und sie wissen, daß sie damit Teil der k Zeugen für ihn geworden sind. Auch dynamische Änderungen an Aggregationsbeziehungen werden dadurch unterstützt.

Beispiel

Es sei angenommen, in Abbildung 4.6 wäre Knoten x dem Netz gerade beigetreten. Laut Voraussetzung hat er damit, zum Beispiel durch das Protokolls SKEY,

1. zunächst seinen Aggregationspfad $\mathbb{P}$ auf sichere Weise erhalten.
2. mit allen Knoten auf seinem Aggregationspfad auf sichere Weise Schlüssel ausgetauscht.

Dadurch weiß x, daß a sein Aggregationsknoten ist, an den er seine Meßdaten zu schicken hat. Folglich muß x an auch die k Vorgänger von a, mit denen x bereits Schlüssel ausgetauscht hat, Kopien seiner Meßwerte schicken.

4.3.3.2 Ablauf der Verifikation

Vorab soll gelten: Grundsätzlich verschlüsselt jeder Knoten seine Daten, die er an einen anderen Knoten versendet, mit dem dazu passenden paarweisen Schlüssel. So chiffriert Knoten x seinen Meßwert X zu $E_{K_{x,a}}(X)$ und sendet dies an a, Algorithmus 11, Zeile 9. Analog chiffriert y seinen Meßwert Y zu $E_{K_{y,a}}(Y)$. Nach dem Empfang um dem Bilden der Aggregation $f_a(X, Y) = \mathrm{agg}_a$ mit Hilfe der Aggregationsfunktion f_a, siehe Algorithmus 12, Zeile 21, sendet a das Aggregat $E_{K_{a,b}}(\mathrm{agg}_a)$ an b, Zeile 23. Hierbei bezeichnet die Notation $K_{i,j}$ die paarweisen Schlüssel zwischen zwei Knoten i und j in Analogie zu Kapitel 3. Schon das Chiffrieren sorgt für paarweise Sicherheit zwischen den beteiligten Knoten wie x und a, y und a sowie a und b.

Die eigentlich Verifikation der Korrektheit und Authentizität von a's Aggregation wird nun im folgenden überprüft. ESAWN funktioniert dabei grundsätzlich induktiv über die Höhe des Aggregationsbaumes: Es sollen sukzessive alle Aggregationen, angefangen von den Aggregationen der Meßwerte, der „Blattknoten" im Baum, baumaufwärts bis hin zur Senke verifiziert werden. Im Beispiel aus Abbildung 4.6 steht dementsprechend zunächst a's Aggregation zur Überprüfung, dann die von b, c, d usw. Nach der sukzessiven Verifikation all dieser Aggregationen auf Korrektheit und Authentizität des Ursprungs in Gegenwart von jeweils k korrumpierten Knoten kann die Senke davon ausgehen, daß für das letzte, von ihr empfangene Aggregat Korrektheit und Authentizität bezüglich *aller* beteiligter Quellsensoren und Aggregationsknoten gilt – in Gegenwart von den in Abschnitt 2.4, S. 24 definierten $\beta\%$ korrumpierten Knoten. Selbiges dürfen auch sämtliche

(Aggregations-)Knoten zwischen Blättern und Senke annehmen. Unter der Voraussetzung, daß ihre sämtlichen Vorgänger im Baum jeweils Aggregationen erfolgreich überprüft haben, gelten die empfangenen Aggregate als authentisch und korrekt bzgl. aller Sensoren einschließlich der Blätter.

Die nun anschließende vollständige Induktion soll die Vorgehensweise der ESAWN-Verifikation grob beschreiben und in Abschnitt 4.3.4 die daraus resultierende Sicherheit diskutieren. Ein formaler Beweis der Korrektheit von ESAWN folgt in Abschnitt 4.3.8.

Induktionsvoraussetzung

Zu einem bestimmten Protokollschritt beziehungsweise Zeitpunkt t soll nun die Korrektheit der Aggregation von a und die dazugehörige Authentizität der Knoten x und y überprüft werden, wie in Abbildung 4.6 dargestellt. Die Knoten x und y senden dabei die Aggregate X und Y an a, siehe dazu Algorithmus 12, Zeile 23. Die *Induktionsvoraussetzung* sei, daß im Schritt zuvor, $t-1$, sowohl die Korrektheit der Aggregate X und Y als auch die Authentizität der dafür verantwortlichen Quellsensoren, Vorgänger von x und y im Baum, erfolgreich in Gegenwart von k korrumpierten Knoten verifiziert wurden.

Induktionsschluß

Der *Induktionsschluß* überprüft jetzt die Korrektheit und Authentizität der Aggregation von a. Zunächst wird mit $k = 1$ ein korrumpierter Knoten betrachtet. Die einzige Möglichkeit, daß die Aggregatbildung bei $k = 1$ durch a verfälscht werden kann, besteht dann, wenn a selbst der korrumpierte Knoten ist. Die Knoten x und y können weder Aggregatbildung noch Authentizität gegenüber b verfälschen, da sie ja verantwortliche Quellsensoren für die Bildung des Aggregats sind. Falls es sich bei X oder Y bereits selbst um Aggregate handelt, wurde im Schritt zuvor laut Induktionsvoraussetzung ihre Korrektheit und dazugehörige Authentizität verifiziert. Mit $k = 1$ steht b als Zeuge für die Aggregation von a fest. Knoten x schickt

$$E_{K_{x,a}}(X)$$

an a sowie

$$E_{K_{x,b}}(X)$$

an b. Knoten y schickt analog

$$E_{K_{y,a}}(Y)$$

an a sowie

$$E_{K_{y,b}}(Y))$$

an b – vergleiche Algorithmus 12. Aus diesen Chiffraten kann a die Daten X und Y extrahieren und mit Hilfe von f_a das Aggregat agg_a bilden: $\mathrm{agg}_a = f_a(x, y)$.

Nun sendet a das Chiffrat

$$E_{K_{a,b}}(\mathrm{agg}_a)$$

an den Zeugen b. Da a dieses Chiffrat mit $K_{a,b}$ verschlüsselt, kann nur Knoten b entschlüsseln und den Inhalt als von Knoten a stammend authentifizieren. Damit erhält b das Datum agg_a. Weiterhin weiß Knoten b, daß sich die Knoten x und y an a's Aggregation beteiligen, weil sie Teil von b's Aggregationspfad sind. Daher erwartet b, daß die Chiffrate, die es von x und y empfängt, auch von x und y kommen müssen. So entschlüsselt

b mit Hilfe von $K_{x,b}$ und $K_{y,b}$ die Chiffrate von x und y zu X und Y. Da neben b nur x respektive y die Schlüssel $K_{x,b}$ beziehungsweise $K_{y,b}$ besitzen, geht b davon aus, daß X tatsächlich von Knoten x sowie Y von Knoten y stammt: Sie sind authentisch. (Techniken zur Authentizität durch symmetrische Schlüssel diskutiert Abschnitt 2.6.2.2, S. 42.) Schließlich bildet b das Aggregat $\mathrm{agg}'_a = f_a(x, y)$ selbst nach, siehe computeAggregates, Algorithmus 16.

Eingabe: Data AggStoreIn[], int p, i, ID $\mathbb{S}$[][]
1: **Declaration:** int r, s
2: Data AggStoreOut:= new Array [|$\mathbb{S}[i-1]$|]
3:
4: {v aggregiert aus den Daten der Knoten $\mathbb{S}[i]$ die Aggregate der Knoten $\mathbb{S}[i-1]$:}
5: **for** $j := 1$ to $|\mathbb{S}[i-1]|$ **do**
6: ID predecessor := $\mathbb{S}[i-1][j]$
7:
8: {v berechnet Aggregat nur dann, falls es überhaupt verifiziert werden soll:}
9: int $p' :=$ generateRandomNumer$(1 \ldots 100, \mathrm{SEEDS}[\mathrm{predecessor}])$
10: **if** $p' \leq p$ **then**
11: {r, s geben die Grenzen der Knoten auf Höhe i unterhalb von v an, die Quellen für die Aggregation von predecessor sind:}
12: $(r, s) :=$ sourceNodesOf(predecessor)
13:
14: {v berechnet Aggregat:}
15: AggStoreOut$[j] := f_x(\mathrm{AggStoreIn}[r], \mathrm{AggStoreIn}[r+1], \ldots, \mathrm{AggStoreIn}[s])$
16: **end if**
17: **end for**
18: **return** AggStoreOut

Algorithmus 16 v.computeAggregates(AggStoreIn, $p, i, \mathbb{S}$)

Falls $\mathrm{agg}'_a = \mathrm{agg}_a$, dann glaubt b, daß Knoten a korrekt aggregiert hat. Neben der Authentizität der Werte X und Y steht damit auch die Korrektheit von a's Aggregation fest, siehe Funktion checkAggregates in Algorithmus 14. Falls jedoch $\mathrm{agg}'_a \neq \mathrm{agg}_a$, dann hat es einen Betrugsversuch gegeben, und b kann diesen Betrug der Senke S und auch b's Vorgängerknoten im Aggregationsbaum sicher melden, zum Beispiel über eine Nachricht $E_{K_{b,S}}(\mathrm{Betrug}, a)$. Die Überprüfung „stoppt“ in diesem Fehlerfalle und b stellt seine Arbeit bis auf weiteres ein, siehe Algorithmus 14, Zeile 9 und Algorithmus 15.

Das Weitermelden eines Betruges, wie bereits angesprochen, oder das Einstellen der Arbeit ist in diesem Fall von besonderer Bedeutung, da b's Vorgänger unbedingt darüber informiert werden müssen, daß ein Betrug stattgefunden hat. Würde b seine Arbeit nicht bis aus weiteres einstellen, könnte der Angreifer die „Alarm“-Nachrichten von b einfach abfangen und verwerfen – sein Betrug würde damit nicht auffallen. Falls b sich aber bis auf weiteres weigert, am normalen Protokollablauf teilzunehmen, merken dies seine Vorgänger im Aggregationsbaum, da sie Nachrichten von b erwarten, die nun ausbleiben. Durch das Ablaufen von Timern, beispielsweise, kann eine solche Technik relativ einfach implementiert werden, siehe dazu auch Algorithmus 13, Zeilen 15ff.

Damit endet der Induktionsschluß. Zum Zeitpunkt t kann der Angreifer die Aggregatbildung nicht unbemerkt verfälschen.

Es sei an dieser Stelle noch bemerkt, daß ESAWN die induktive Überprüfung der Aggregationen nicht nur für *einen* Aggregationspfad, wie in Abbildung 4.6 gezeigt, sondern schrittweise für *alle* Aggregationen und damit alle Aggregationspfade durchführt. ESAWN beginnt dabei bei der Überprüfung der Aggregationsknoten der Blätter und arbeitet sich dann sukzessive durch den Baum durch in Richtung Senke.

Denial-of-Service Angriffe durch ESAWN

Es ist durchaus möglich, daß bei Einsatz von ESAWN korrumpierte Knoten eine neue Art von Denial-of-Servive Angriff starten. Korrumpierte Knoten könnten grundlos Alarm-Nachrichten erzeugen und verschicken, um das Netz so ständig in STOP_ALL_WORK-Zustände zu versetzen. Obwohl bereits im Grundlagenkapitel Denial-of-Service-Angriffe ausgeschlossen worden sind, sind solche *false-positives* interessant: Aus diesem Grund diskutiert Abschnitt 4.3.7 false-positives ausführlicher.

$k > 1$ Zeugen

Analog definiert sich der Induktionsschluß in ESAWN bei mehr als einem, nämlich k an einem Protokollschritt involvierten, korrumpierten Knoten. Der Angreifer kann das Aggregat ausschließlich dann erfolgreich fälschen, wenn er neben a auch noch k Zeugen, insgesamt $k + 1$ Knoten, korrumpiert. Bei weniger oder genau k korrumpierten Knoten hintereinander fällt jeder Betrugsversuch auf. Auf Details dazu geht die Sicherheitbetrachtung in Abschnitt 4.3.4 ein. Zur Überprüfung von a's Aggregation wählt ESAWN deshalb auch entsprechend $k > 1$ Zeugen aus, die k Vorgänger von a. Das sind im Beispiel aus Abbildung 4.6 die Knoten $b, c, d, \ldots$. Die Knoten x und y schicken Kopien ihrer Meßwerte nicht nur an b, sondern auch an $c, d, \ldots$. Knoten x schickt

$$
\begin{array}{c}
E_{K_{x,a}}(X) \text{ an } a, \\
E_{K_{x,b}}(X) \text{ an } b, \\
E_{K_{x,c}}(X) \text{ an } c, \\
E_{K_{x,d}}(X) \text{ an } d, \\
\ldots
\end{array}
$$

Analog schickt y

$$
\begin{array}{c}
E_{K_{y,a}}(Y) \text{ an } a, \\
E_{K_{y,b}}(Y) \text{ an } b, \\
E_{K_{y,c}}(Y) \text{ an } c, \\
E_{K_{y,d}}(Y) \text{ an } d, \\
\ldots
\end{array}
$$

Wie im Abschnitt zuvor berechnet a das Aggregat agg_a durch $\mathrm{agg}_a = f_a(X, Y)$. Um allen k Zeugen gerecht zu werden, versendet a nun

$$
\begin{array}{c}
E_{K_{a,b}}(\mathrm{agg}_a) \text{ an } b, \\
E_{K_{a,c}}(\mathrm{agg}_a) \text{ an } c, \\
E_{K_{a,d}}(\mathrm{agg}_a) \text{ an } d, \\
\ldots
\end{array}
$$

Knoten b überprüft mit X und Y, ob agg_a korrekt durch $f_a(X, Y)$ berechnet worden ist. Falls nicht, dann kann b wiederum eine Fehlernachricht $E_{K_{b,S}}(\mathrm{Betrug}, a)$ an die Senke schicken.

Damit sind auch die Knoten $c, d, \ldots$ in der Lage, a's Aggregation zu verifizieren: Sie kennen sowohl X und Y als auch agg_a.

Bemerkung: Aus Effizienzgründen lohnt es sich für x, seine Chiffrate *nicht* einzeln an a, b, c, d usw. zu schicken. Eine Nachricht, welche x an zum Beispiel d schickt, läuft aus Routing-Sicht über a, b und c, siehe hierzu die Diskussionen in Abschnitt 2.2.4, S. 16. Das bedeutet, x versendet eine Nachricht an a mit der Bitte um Weiterleitung an d, a „routet" diese Nachricht an b usw. Um diesen Aufwand an Nachrichten zu verringern, schickt x sämtliche Chiffrate $\{E_{K_{x,a}}(X), E_{K_{x,b}}(X), E_{K_{x,c}}(X), E_{K_{x,d}}(X), \ldots\}$ an a und versucht dabei, so viele wie möglich zusammen in eine Nachricht zu „packen". Knoten x packt seine Chiffrate zusammen in ein oder mehrere Nachrichten. Da die Länge der Chiffrate häufig deutlich kleiner als die Nachrichtlänge ist, kann so die Gesamtanzahl der zu übertragenden Nachrichten erheblich verringert werden. Die Sicherheit von ESAWN reduziert dies nicht, weil der Angreifer auch weiterhin ohne Kenntnis eines der Schlüssel $K_{x,a}, K_{x,b}, \ldots$ keines der Chiffrate verändern kann. Nur der Übersichtlichkeit halber, zum einfacheren Verständnis, steht in den Funktionen $\mathrm{doMeasure}$ und $\mathrm{doAggregate}$ des Pseudocodes die Variante, in der ein Knoten jedes Chiffrat einzeln an die Zeugen schickt.

Induktionsanfang

Schließlich bleibt noch der Induktionsanfang zu zeigen. Der allererste Protokollschritt in ESAWN, entsprechend dem Zeitpunkt $t = 0$, charakterisiert die Überprüfung der Blätter. Unter der Voraussetzung aus Abschnitt 2.4, S. 24, daß die Blätter vom Angreifer niemals kompromittiert werden können, gestaltet sich der Induktionsanfang relativ einfach: Die von den Blättern an die ersten Aggregationsknoten verschickten Daten sind damit authentisch und korrekt „aggregiert".

4.3.3.3 Kenntnis von $\mathbb{P}$

ESAWN setzt voraus, daß jeder Knoten x gesichert Kenntnis über seinen Aggregationspfad $\mathbb{P}_x$ besitzt. Dies ist nicht selbstverständlich, denn $\mathbb{P}_x$ kann sich im Laufe der Zeit ändern, beispielsweise wenn neue Knoten hinzukommen oder sich ganz allgemein Aggregationsbeziehungen von Knoten untereinander ändern.

Ein Vergleich von ESAWN mit anderen Arbeiten ist dennoch fair: Solche Veränderungen im Aggregationsbaum haben bei ESAWN nur *lokal begrenzte* Auswirkungen. Verändern sich Aggregationspfade oder Routing-Informationen in einem Teilbaum G'_i des Aggregationsbaums G, so hat dies keine Konsequenzen für alle anderen Teilbäume G'_j von G, solange $G'_j \not\subset G'_i$ und $G'_i \not\subset G'_j$. Jeder Knoten x benötigt dementsprechend durch $\mathbb{P}_x$ aktuell nur lokales Wissen über den Zustand des Netzes in seiner näheren Umgebung und keine globale Sicht auf das gesamte Sensornetz.

Darüberhinaus nehmen auch einige verwandte Arbeiten Kenntnis über den Aggregationspfad oder Knoten in näherer Umgebung von x an, darunter zum Beispiel Du et al. [82], Hu und Evans [107], Vogt [227]. Bereits in Kapitel 3 ist zudem gezeigt worden, daß der Energieaufwand, der sich aus einer möglichen *Veränderung* im Aggregationsbaum ergibt, nur logarithmisch mit der Gesamtzahl der Knoten n wächst, $O(\log n)$, und allgemein durch Kenntnis von $\mathbb{P}$ der Speicherverbrauch insgesamt nur mit $O(\log n)$ pro Knoten wächst.

4.3.3.4 Probabilistische Verifikation

Der offensichtliche Nachteil ESAWNs liegt in der durch k Zeugen implizierten, deutlich gestiegenen Anzahl von Nachrichten. Durch das mehrfache Versenden von Meßwerten und Aggregaten an Zeugen nivelliert ESAWN die Energieeinsparungen durch Aggregation.

Um daher den Energieverbrauch von ESAWN zu senken, kann, auf Kosten der Sicherheit, die Prüfwahrscheinlichkeit einzelner Aggregationen von $p = 100\%$ auf $p < 100\%$ reduziert werden. Weniger häufig durchgeführte Aggregationen bedeuten weniger Nachrichten und damit weniger Energieverbrauch durch die Funkübertragungen. Auf der anderen Seite wirken sich weniger häufig und nur stichprobenartig durchgeführte Überprüfungen negativ auf die Sicherheit ESAWNs aus.

Das Prinzip der probabilistischen Verifikation läßt sich wie folgt beschreiben: Der Benutzer bestimmt a priori die Wahrscheinlichkeit $p \leq 100\%$, mit der ESAWN Aggregationen überprüft. In jedem Aggregations- beziehungsweise Protokollschritt entscheidet ESAWN für jede mögliche Aggregation, ob die Aggregation mit den dafür vorgesehenen Zeugen überprüft werden muß oder nicht. Die Wahrscheinlichkeit der Überprüfung bleibt dabei von Aggregation zu Aggregation untereinander unabhängig und beträgt jeweils $p\%$.

Ein Zeuge empfängt probabilistisch Kopien von Meßwerten beziehungsweise Aggregaten mit der Funktion $\mathrm{receiveFromNodes}$ aus Algorithmus 13, Zeilen 10ff. Ebenso probabilistisch berechnet er daraufhin selbst die Aggregate in $\mathrm{computeAggregates}$ aus Algorithmus 16, Zeilen 9ff. Schließlich überprüft er empfangene Aggregate mit seinen eigenen Berechnungen in der Funktion $\mathrm{checkAggregates}$ aus Algorithmus 14, Zeilen 5ff.

Analog *versendet* ein Meßknoten beziehungsweise Aggregationsknoten seine Meßwerte oder Aggregate auch nur in $p\%$ aller Fälle an die Zeugen. Dies zeigt Algorithmus 11, in den Zeilen 12ff sowie Algorithmus 12 in den Zeilen 26ff.

Genauere Untersuchungen sowohl zum Energieaufwand als auch zur Sicherheit durch ein probabilistisch relaxiertes ESAWN im Vergleich zu einfacher, jedoch unsicherer Aggregation betrachtet Abschnitt 4.4.

4.3.3.5 Verteilung des Seeds

Die Verifikation gelingt nur dann, wenn alle an einer Aggregation beziehungsweise Verifikation beteiligten Knoten, das sind die Zeugen des Aggregationsknotens und seine direkten Vorgänger im Aggregationsbaum, gleichzeitig wissen, ob eine Verifikation durchgeführt wird oder ob nicht.

Vorgehen

In ESAWN entscheidet auf all den zu einer Verifikation gehörenden Knoten ein beliebiger Pseudozufallszahlengenerator [156] in Abhängigkeit von p, ob die aktuelle Aggregation für einen bestimmten Knoten durchzuführen ist. Damit sämtliche an der Verifikation beteiligten Knoten zum selben Zeitpunkt synchronisiert über die Verifikation entscheiden können, initialisiert ESAWN den initialen Startwert des Generators (engl. *Seed*) auf all diesen Knoten gleich. Dieser Seed kann zum Beispiel bei der Festlegung der Aggregationsbeziehungen und damit verbunden der Einteilung der Zeugen den jeweiligen Knoten mitgeteilt werden. Da alle Zeugen und Quellsensoren voneinander Kenntnis haben und

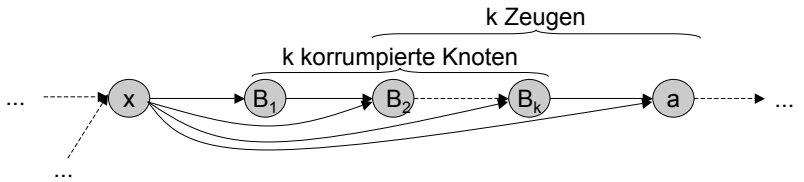


Abbildung 4.7 Auswahl von k Zeugen, mindestens ein Zeuge nicht kompromittiert

über paarweise Schlüssel verfügen, wäre auch ein gemeinsames Wahlverfahren denkbar. Die einfachste Lösung für einen Seed, den alle an der Aggregation beteiligten Knoten kennen, ist jedoch, die ID des Aggregationsknotens dafür zu nehmen. So könnte zum Beispiel der Seed zur Überprüfung der Aggregation von Knoten a ganz einfach $H(a)$ sein. Da alle Knoten die ID a kennen, wissen sie damit auch automatisch den Seed.

An dieser Stelle sei einfach angenommen, daß die Zeugen, Quellsensoren und Aggregationsknoten ein und den selben Seed kennen. Damit ist eine synchrone Überprüfung möglich. Jeder Knoten besitzt von allen Knoten, an deren Verifikation er teilnimmt den entsprechenden Seed. Das Seed-*Array* wird im Pseudocode mit SEEDS bezeichnet, vergleiche zum Beispiel Algorithmus 11 Zeile 12.

In diesem Kontext spielt es weiterhin *keine* Rolle, ob auch korrumpierten Knoten den Seed kennen: Da sich mit einer bestimmten Wahrscheinlichkeit unter allen beteiligten Knoten auch korrumpierte Knoten befinden, kennen diese den Seed. Mit Hilfe des Seeds wissen sie durch Berechnen der nächsten Pseudozufallszahlen im voraus, ob ein Aggregat geprüft wird oder ob nicht. Darauf können sich die korrumpierten Knoten einstellen und nur dann Aggregate fälschen, wenn keine Überprüfung durchgeführt werden wird. Das erhöht jedoch *nicht* die Chancen für die korrumpierten Knoten, ein Aggregat unbemerkt zu fälschen, denn eine einzelne Aggregation a kann trotzdem nur in $(1-p)$ Prozent der Fällen von a erfolgreich gefälscht werden, falls weniger als $(k-1)$ korrumpierte Knoten auf dem Aggregationspfad von a hintereinanderliegen.

4.3.4 ESAWNs Sicherheit

Das Ziel von ESAWN liegt darin zu verhindern, daß legitime Knoten *unbemerkt* falsche beziehungsweise gefälschte Aggregate empfangen. Die legitimen Knoten würden daraufhin ebenso falsche Schlüsse ziehen, falsch weiteraggregieren und unter Umständen als Aktoren die falschen Aktionen durchführen. Grundsätzlich verhindert dies die Auswahl von k-Zeugen, wodurch immer mindestens ein Zeuge legitimer Knoten bleibt.

Zum einfacheren Verständnis sei nun angenommen, daß k Angreifer direkt hintereinander auf einem Aggregationspfad liegen. Aus Sicherheitssicht stellt dies wieder eine Art schlechtester Fall dar, weil hintereinanderliegende, korrumpierte Knoten zusammenarbeiten und an der Überprüfung der *gleichen* Aggregation als Zeugen teilnehmen. Jede andere Verteilung der k korrumpierten Knoten, bei der mindestens ein nicht-korrumpierter Knoten oberhalb von $k_1 < k$ Knoten im Aggregationsbaum liegt und unterhalb von $(k-k_1)$ Knoten, läßt sich wie $k = k_1$ Knoten behandeln: ESAWN muß garantieren, daß der legitime Knoten trotz Zusammenarbeit der k_1 korrumpierten Knoten kein gefälschtes Aggregat akzeptiert.

Den Sachverhalt k hintereinander-korrumpierter Knoten zeigt Abbildung 4.7 in Anlehnung an Abbildung 4.6. Auch hier sei x Quellsensor beziehungsweise der vorhergehende Aggregationsknoten im Baum, dessen Meßwert oder Aggregat in die Aggregatbildung des

zu überprüfenden Aggregationsknoten B_1 einfließt. Knoten x seit laut Induktionsvoraussetzung bereits erfolgreich verifiziert: Er liefert ein korrektes, authentisches Aggregat an B_1. Weiterhin seien die Knoten B_i als korrumpiert zu betrachten. Nicht nur den zu überprüfenden Knoten B_1, sondern auch die Knoten $B_2, \ldots, B_k$ hat der Angreifer erfolgreich korrumpiert. Der erste nicht korrumpierte Knoten nach den B_i's heißt a. Als k Zeugen der Aggregation von B_1 gelten in ESAWN sowohl $B_2, \ldots, B_k$ als auch a. Knoten x schickt seinen Meßwert damit an B_1 und die k Zeugen $B_2, \ldots, B_k, a$. Damit erhält mindestens ein nicht-korrumpierter Knoten, hier a, einen Meßwert von x und bekommt die Möglichkeit, B_1's Aggregation diesbezüglich zu überprüfen. Bei k Angreifern und k Zeugen wird klar, daß jeder Knoten insgesamt k Aggregationen auf seinem Aggregationspfad bezeugen muß – seine k Nachfolger im Aggregationsbaum. Knoten a überprüft demnach neben B_1 auch noch die $k-1$ folgenden Aggregationen $B_2, \ldots, B_k$ – das sind Nachfolgerknoten im Aggregationsbaum. Dazu empfängt a Daten der $l := k+1$ Knoten $x, B_1, \ldots, B_k$. Dies zeigen auch die Zeilen 9ff in Algorithmus 12.

Damit ist a in der Lage, das von B_k empfangene Aggregat auf Korrekt und Authentizität in Bezug auf x und möglicherweise k korrumpierten Knoten $B_1, \ldots, B_k$ zu untersuchen. Damit kann a nur ein korrektes und authentisches Aggregat von B_k empfangen.

ESAWNs Sicherheit soll am Beispiel mit $k = 2$ korrumpierten Knoten und Zeugen verdeutlicht werden.

4.3.4.1 Beispiel mit k=2

Im folgenden seien die Knoten B_1 und B_2 korrumpiert sowie wie a ein legitimer Knoten, der den Betrug von B_1 und B_2 bemerken muß. Jede Kommunikation zwischen zwei Knoten sei dabei im folgenden immer mit den paarweisen Schlüsseln der beteiligten Knoten chiffriert.

Knoten x sendet im ersten Schritt seinen Meßwert beziehungsweise sein Aggregat X sowohl an B_1 als auch an die beiden Zeugen dieser Aggregation: B_2 und a. Welche der gerade an der ESAWN-Verifikation beteiligten Knoten B_1, B_2, a über welche Information verfügt, zeigt Tabelle 4.2.

Knoten	Daten auf Knoten
B_1	Schritt 1: X, $\mathrm{agg}_{\mathrm{B}_1} := f_{B_1}(X)$
B_2	Schritt 1: X, $\mathrm{agg}_{\mathrm{B}_1} := f_{B_1}(X)$, $\mathrm{agg}_{\mathrm{B}_2} := f_{B_2}(\mathrm{agg}_{\mathrm{B}_1})$
	Schritt 2: $\mathrm{agg}'_{\mathrm{B}_1}$
a	Schritt 1: X, $\mathrm{agg}_{\mathrm{B}_1} := f_{B_1}(X)$, $\mathrm{agg}_{\mathrm{B}_2} := f_{B_2}(\mathrm{agg}_{\mathrm{B}_1})$
	Schritt 2: $\mathrm{agg}'_{\mathrm{B}_1}$, $\mathrm{agg}_{\mathrm{B}_1} \stackrel{?}{=} \mathrm{agg}'_{\mathrm{B}_1}$
	Schritt 3: $\mathrm{agg}'_{\mathrm{B}_2}$, $\mathrm{agg}_{\mathrm{B}_2} \stackrel{?}{=} \mathrm{agg}'_{\mathrm{B}_2}$

Tabelle 4.2 Zum Beispiel mit $k = 2$, Daten auf den einzelnen Knoten.

Nachdem x sein Datum versendet hat, aggregiert B_1 dieses mit Hilfe seiner Aggregationsfunktion f_{B_1} zu $f_{B_1}(X) = \mathrm{agg}_{\mathrm{B}_1}$. Dieses Aggregat wird von B_1 in Schritt 2 an B_2, a und an a's Vorgänger auf dem Aggregationspfad gesendet. Auch dies zeigt Tabelle 4.2. Da es sich bei B_1 laut Annahme um einen korrumpierten Knoten handelt, besteht die Gefahr, daß B_1 sein Aggregat $\mathrm{agg}_{\mathrm{B}_1}$ möglicherweise falsch aggregiert hat. Anstatt aus X das Aggregat $\mathrm{agg}_{\mathrm{B}_1}$ korrekt zu aggregieren, könnte B_1 unter Umständen auch ein gefälschtes $\mathrm{agg}'_{\mathrm{B}_1} \neq \mathrm{agg}_{\mathrm{B}_1}$ an B_2 und a versenden. Für Knoten B_2 spielt das weiter keine Rolle,

da dieser ja laut Annahme ebenso wie B_1 vom Angreifer im Vorfeld korrumpiert worden ist und mit B_1 zusammenarbeitet. Knoten a jedoch kann überprüfen, ob das von B_1 empfangene $\mathrm{agg}'_{\mathrm{B}_1}$ dem korrekten $\mathrm{agg}_{\mathrm{B}_1}$ entspricht: Zunächst berechnet er aus X selbst $f_{B_1}(X) = \mathrm{agg}_{\mathrm{B}_1}$ und vergleicht dann ganz einfach $\mathrm{agg}_{\mathrm{B}_1} \stackrel{?}{=} \mathrm{agg}'_{\mathrm{B}_1}$. Falls die Überprüfung fehlschlägt, meldet a der Basisstation eine Fälschung. Knoten B_2 aggregiert mit seiner Aggregationsfunktion f_{B_2} zu $\mathrm{agg}'_{\mathrm{B}_2} = f_{B_2}(\mathrm{agg}'_{\mathrm{B}_1})$ und schickt in Schritt 3 $\mathrm{agg}'_{\mathrm{B}_2}$ an a. Auch dies kann a wie oben überprüfen.

Jetzt überprüft a schließlich das von B_2 empfangene und möglicherweise falsch berechnete Aggregat $\mathrm{agg}_{\mathrm{B}_2}{}'$. Aus dem selbst berechneten $\mathrm{agg}_{\mathrm{B}_1}$ bildet a erst $f_{B_2}(\mathrm{agg}_{\mathrm{B}_1}) = \mathrm{agg}_{\mathrm{B}_2}$ und überprüft dann $\mathrm{agg}_{\mathrm{B}_2} \stackrel{?}{=} \mathrm{agg}'_{\mathrm{B}_2}$. Bei Übereinstimmung weiß a, daß nicht nur B_1, sondern auch B_2 korrekt aggregiert haben. Er kann jetzt selbst seine Aggregation durchführen und an seinen Vorgänger weiterschicken. Schlägt die Überprüfung $\mathrm{agg}_{\mathrm{B}_2} \stackrel{?}{=} \mathrm{agg}'_{\mathrm{B}_2}$ fehl, so schickt a wieder eine Fehlermeldung an die Senke. Dieses schrittweise Empfangen von Aggregaten und Vergleichen mit vorher selbst berechneten Aggregaten zeigt Algorithmus 12 in den Zeilen 11ff.

Damit ist der Induktionsschluß fertig und damit für die Aggregation B_1 gezeigt, daß selbst in Gegenwart von $k = 2$ korrumpierten Knoten, kein nicht-korrumpierten Knoten unbemerkt ein gefälschtes Aggregat empfängt. Auch die Authentizität der Aggregation ist gesichert, dadurch daß x alle Kopien seines Wertes X jeweils authentisch mit Hilfe paarweiser Verschlüsselung an die Zeugen schickt. Die Zeugen wissen dann, daß X tatsächlich von x stammt.

An dieser Stelle sei noch erwähnt, daß ESAWN Sicherheit auch gegen mehr als insgesamt k korrumpierte Knoten pro Protokollschritt bietet, sobald diese nicht unmittelbar hintereinander auf einem Aggregationspfad liegen. Die Zahl k gibt eine Art untere Schranke für die korrumpierten Knoten pro Pfad an, für den Fall, daß diese direkt hintereinanderliegen: Kein Teilpfad von korrumpierten Knoten darf länger als k sein.

4.3.4.2 error-Nachricht, Fehlerzustand und Blockieren von Nachrichten

In dem Fall, in dem ein Knoten a als Zeuge einer Aggregation ein gefälschtes Aggregat entdeckt, informiert er all seine Vorgänger im Aggregationsbaum incl. der Senke über diese Tatsache. Das leistet die Funktion error in Algorithmus 15. Knoten a schickt einfach Nachrichten an seine Vorgänger, in denen steht, daß er einen Betrug entdeckt hat. Die Vorgänger können sich dann dementsprechend darauf einstellen und den Teilbaum von a nicht weiter für ihre Aggregation benutzen oder ähnliches.

Ein Angreifer kann diese Nachrichten von a oder grundsätzlich jede beliebige andere Nachricht, beispielsweise die Kopien von Aggregaten an Zeugen, blockieren. Um dem entgegenzuwirken, stellt Knoten a seine normale Tätigkeit als Aggregationsknoten bis auf weiteres ein, siehe Zeile 4. Er verschickt selbst keine Aggregate mehr und keine Kopien davon an Zeugen. Mit Hilfe von ablaufenden Timeouts in den receiveFromNodes Funktionen der Aggregationsknoten oberhalb von a im Aggregationsbaum merken dann allerdings auch diese Knoten, daß Nachrichten an sie ausbleiben und in ihrem Teilbaum „etwas nicht stimmt“. Das Blockieren von Nachrichten im Sensornetz ist demnach kein effektiver Angriff.

Wie bereits oben erwähnt lassen sich so vom Angreifer *false-positive* Alarmmeldungen erzeugen: ein DoS-Angriff. Dies diskutiert Abschnitt 4.3.7.

4.3.5 Auswirkung probabilistischer Überprüfung

Die Überprüfung der Aggregationen gegen k-korrumpierte Knoten kostet durch das mehrfache Versenden von Nachrichten viel zusätzliche Energie. Details zum tatsächlichen Energieverbrauch präsentiert die Evaluation in Abschnitt 4.4 später. Zunächst werden nun noch die *Sicherheitsauswirkungen* der Energieoptimierung durch das probabilistische Verifizieren von Aggregationen diskutiert.

Wenn eine Aggregation, um Energie zu sparen, nur mit einer Wahrscheinlichkeit von $p \leq 100\%$ überprüft wird, so kann dessen Sicherheit auch nur mit $p\%$ garantiert werden. Ein korrumpierter Aggregationsknoten hätte in $(1-p)\%$ der Fälle die Chance, seine Aggregation unbemerkt zu fälschen und weiterzusenden.

Es kommt hinzu, daß die Aggregationen voneinander unabhängig jeweils mit $p\%$ geprüft werden und sich die Wahrscheinlichkeiten für einen möglichen Betrug dementsprechend kumulieren. Gelingt es mindestens einem der Aggregationsknoten auf dem kompletten Aggregationspfad, sein Aggregat unbemerkt zu fälschen, entweder durch eine ausgelassene Verifikation oder weil mehr als k Zeugen auf einem Aggregationspfad hintereinander korrumpiert sind, so sind alle daraus berechneten Aggregate auch gefälscht. Die „gesamte Aggregation“ ist damit erfolgreich gefälscht. Die *Wahrscheinlichkeit für eine Korrekte Aggregation* (WKA) bezeichnet die Wahrscheinlichkeit, daß kein Betrug beim Aggregieren im Netz unentdeckt bleibt. Das heißt, daß entweder jeder Betrug auffällt oder kein Betrug stattfindet. Im folgenden sei angenommen, daß korrumpierte Knoten immer versuchen zu betrügen. Hierbei handelt es sich um keine Beschränkung der Allgemeinheit: Jeder korrumpierte Knoten, der nicht betrügt, handelt wie ein legitimer Knoten. Sobald nur $\beta' < \beta$ Prozent der Knoten betrügen, gelten die folgenden Aussagen eben für β'.

Herleitung WKA

Gesucht ist die folgende Gegenwahrscheinlichkeit:

> Wie hoch ist die Wahrscheinlichkeit, daß kein Aggregationsknoten korrumpiert ist und nicht überprüft wird?

Wenn $|\Lambda|$ die Anzahl der Aggregationsknoten im Netz bezeichnet, läßt sich obige Gegenwahrscheinlichkeit formal schreiben als

$$\begin{aligned}\mathrm{WKA} &= (1-(\beta\cdot(1-p)))^{|\Lambda|}\\ \mathrm{WKA} &= (1-(\beta-\beta\cdot p))^{|\Lambda|}\\ \mathrm{WKA} &= (1-\beta+\beta\cdot p)^{|\Lambda|}\end{aligned}$$

Damit bezeichnet WKA die Wahrscheinlichkeit, daß ein Knoten nicht korrumpiert ist oder, falls er korrumpiert ist, überprüft wird – für alle Aggregationsknoten $|\Lambda|$. In einem Aggregationsbaum mit Aggregationsgrad δ befinden sich auf Höhe h' durchschnittlichen $\delta^{h'}$ Aggregationsknoten. Im gesamten Baum mit n Knoten und Höhe $h \approx \log_\delta(1+(\delta-1)n)-1$ befindet sich demnach im *Durchschnitt* die folgende Anzahl Aggregationsknoten:

$$|\Lambda| = \sum_{i=1}^{h-1}\delta^i = \frac{n-1}{\delta} - 1.$$

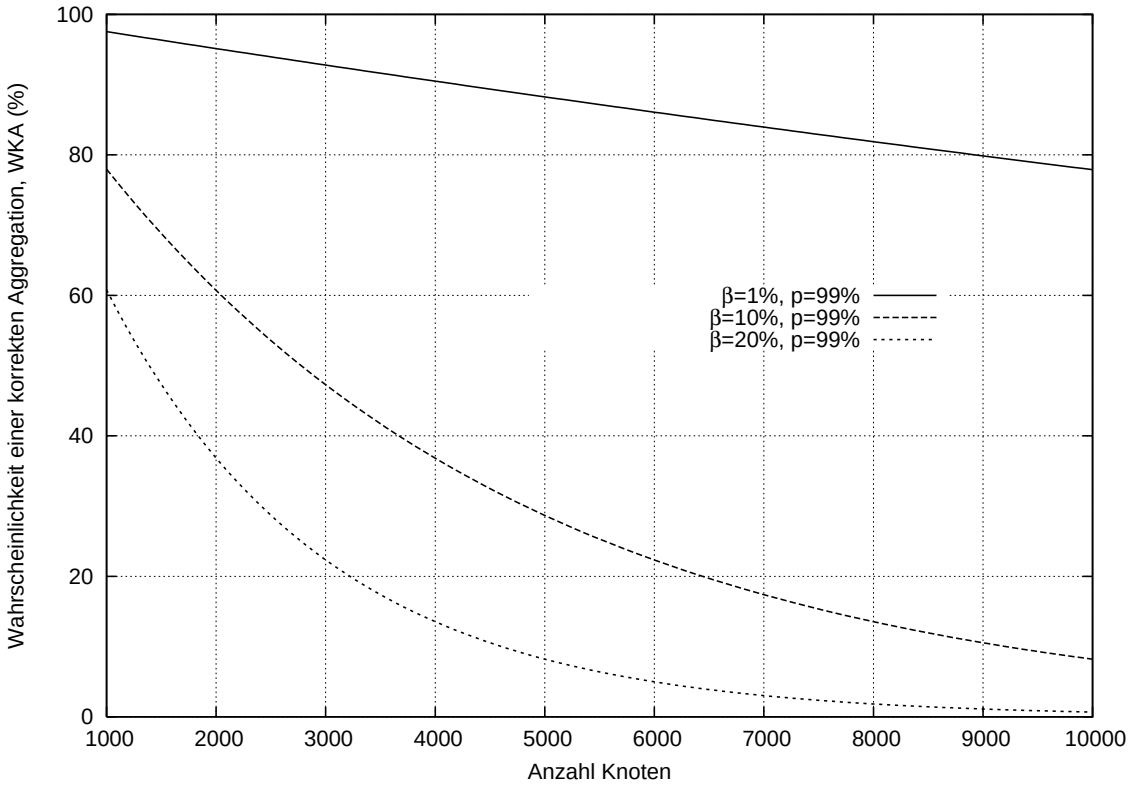


Abbildung 4.8 Theoretischer Verlauf der WKA, $\delta = 4$

Daraus läßt sich schließlich die WKA berechnen zu

$$\text{WKA} \approx (1 - \beta + \beta \cdot p)^{\frac{n-1}{\delta}-1}.$$

Bei $p < 100\%$ sinkt die Wahrscheinlichkeit WKA damit relativ schnell. Dies zeigt beispielhaft auch der Verlauf der beiden Kurven in Abbildung 4.8, einmal mit $\beta = 10\%$, $p = 99\%$ und einmal mit $\beta = 20\%$, $p = 99\%$. Diese theoretische Betrachtung von WKA reicht allerdings nicht aus, weil sie davon ausgeht, daß $k' \leq k$ korrumpierte Knoten auf allen Aggregationspfaden hintereinander liegen. Dies gilt wiederrum nur mit einer bestimmten Wahrscheinlichkeit. Details dazu untersucht der nächste Abschnitt sowie die Evaluierung in Abschnitt 4.4.

Authentizität von weniger als 100% mag auf der einen Seite enttäuschend klingen, bringt aber, wie später noch zu sehen sein wird, deutliche Energieeinsparungen. In vielen Szenarien reicht eine Aussage über einen bestimmten Prozentsatz sicher transportierter Daten außerdem aus. Beispiel: Wenn 90% der aggregierten Gesundheitsdaten korrekt und authentisch transportiert werden, erlaubt dies eben einen medizinischen Notfall mit „sehr hoher Wahrscheinlichkeit“ zu erkennen. In anderen Szenarien sind sicherlich auch noch deutlich schwächere Anforderungen bzgl. probabilistisch relaxierter Sicherheit (zum Beispiel 80%, 70%, usw.) denkbar.

Zusammengefaßt: Falls alle $\beta\%$ Knoten betrügen, dann kann die Senke im Durchschnitt in WKA-Prozent der Fälle einen Betrug erfolgreich feststellen. Wenn nur ein Teil aller $\beta\%$ korrumpierten Sensorknoten betrügen, steigt im Durchschnitt die Wahrscheinlichkeit für eine korrekte Aggregation höher als WKA. Empfängt die Senke weiterhin ein Aggregat, und es ist kein Betrugsversuch gemeldet worden, dann ist dieses in WKA-Prozent

der Fälle auch korrekt beziehungsweise authentisch aggregiert worden – falls *alle* korrumpierten Knoten versuchen zu betrügen. Andernfalls, falls weniger als sämtliche $\beta\%$ der korrumpierten Knoten überhaupt versuchen zu betrügen, gilt wie oben, daß die Wahrscheinlichkeit für eine korrekte Aggregation höher als WKA liegt. Damit gibt WKA eine untere Schranke für die durchschnittliche Wahrscheinlichkeit für korrekte Aggregation an – für den laut Angreifermodell anzunehmenden Fall, daß alle korrumpierten Knoten gleichzeitig versuchen zu betrügen.

4.3.6 Sicherheitsbetrachtung: Wahl von k und p

Bei Einsatz der probabilistisch relaxierten Sicherheit von ESAWN fällt die Wahrscheinlichkeit für eine korrekte Aggregation $\mathrm{WKA} \approx (1-\beta+\beta\cdot p)^{\frac{n-1}{\delta}-1}$, das heißt die Authentizität aller Aggregation von den Blättern bis zur Senke, relativ schnell. Falls korrumpierte Knoten bei mindestens einer Aggregation, die zufällig nicht überprüft wird, ein Aggregat fälschen können, gilt das gesamte Aggregat, welches die Senke schließlich erreicht, als „gefälscht“.

Auf der anderen Seite will der Benutzer eine gewisse „Ende-zu-Ende“-Authentizität zwischen Knoten und Senke mit einer bestimmten Wahrscheinlichkeit WKA im Sensornetz garantieren und braucht dafür die entsprechenden Parameter k und p, mit denen er sein Sensornetz konfiguriert. Dies ist dann möglich, wenn der Benutzer die wahrscheinliche Gesamtknotenzahl n sowie den voraussichtlichen durchschnittlichen Aggregationsgrad δ abschätzen kann und daraus h berechnet. Wenn er zudem ungefähr beurteilen kann, wieviele Knoten $\mathcal{B}$ der Angreifer voraussichtlich korrumpiert beziehungsweise gegen wieviele korrumpierte Knoten er sein Netz absichern und verteidigen möchte, berechnet er β. Der Prozentsatz korrumpierter Knoten β berechnet sich aus $\beta = \frac{\mathcal{B}}{n}$. Aus $\mathrm{WKA} \approx (1-\beta+\beta\cdot p)^{\frac{n-1}{\delta}-1}$ ergibt sich durchschnittlich

$$p \approx 1 - \frac{1 - \sqrt[(n-\delta-1)]{\mathrm{WKA}^{\delta}}}{\beta}.$$

Analog bestimmt der Benutzer auch die Anzahl der k notwendigen Zeugen, wenn er erwartet, daß ein Angreifer eine bestimmte Anzahl Knoten $\mathcal{B}$ im Netz korrumpiert. In einem Aggregationsbaum der Höhe h liegen im Durchschnitt $\beta\cdot(h-1)$ korrumpierte Knoten auf jedem Pfad zwischen Blättern und Wurzel. Im „schlimmsten Fall“ liegen alle diese korrumpierten Knoten direkt hintereinander auf einem Aggregationspfad. Bei der Verifikation einer Aggregation muß ESAWN damit im schlimmsten Fall gegen bis zu $\beta\cdot(h-1)$ solcher Knoten noch sicher arbeiten. Der Benutzer könnte k folglich zunächst wählen zu $k = \lceil\beta\cdot(h-1)\rceil$. (Hier wird k deshalb aufgerundet, weil k nur ganzzahlige Werte, die Zeugen, annehmen kann). Im *Durchschnitt* würde der Benutzer sich auf diese Weise gegen den angenommenen Angreifer verteidigen können.

Die Wahl von k ist allerdings wichtig: Sobald auf irgendeinem Aggregationspfad im Baum tatsächlich mehr als k korrumpierte Knoten hintereinanderliegen, können diese Knoten *jedes* Aggregat unbemerkt fälschen. Da mit einer bestimmten Wahrscheinlichkeit in einem zufälligen Aggregationsbaum auch mehr als $k = \lceil\beta\cdot(h-1)\rceil$ korrumpierte Knoten auf einem Aggregationspfad hintereinanderliegen können, reicht diese Wahl von k nicht *immer* aus. Je größer $\mathcal{B}$ und damit β, desto wahrscheinlich wird es, daß auch $k' > k$ Knoten hintereinander auf einem Aggregationspfad liegen. Um sicherzustellen, daß auf jedem

Aggregationspfad „genügend“ Zeugen liegen, könnte er $k = \lceil h - 1 \rceil$ setzen, was entsprechend mehr Energie kostet. Dabei ist allerdings auch noch zu bedenken, daß h aus einer Abschätzung von n und δ entsteht, dementsprechend also auch nur eine *durchschnittliche Höhe* über alle möglichen Aggregationsbäume angibt. Er kann die wirkliche Höhe seines aktuellen Aggregationsbaumes – laut Annahme – niemals bestimmen: Der Aggregationsbaum ist dynamisch, Knoten verlassen das Netz temporär usw. Wüßte der Benutzer die wirkliche Höhe, dann könnte er k entsprechend sicher wählen. Da die aktuelle Höhe aber unbekannt ist, kann in diesem Fall die Erhöhung von k um einen „Sicherheitspuffer“ Abhilfe bringen. Der Benutzer setzt k nicht auf $k = \lceil h - 1 \rceil$, sondern auf $k = \lceil h \rceil$, $k = \lceil h + 1 \rceil$ usw. Damit wird die Gefahr nochmals minimiert, daß $k' > k$ korrumpierte Knoten im Baum hintereinanderliegen.

Wie man sieht, gestaltet sich die Wahl von k allgemein als äußerst schwierig. Die Evaluierung in Abschnitt 4.4 untersucht daher die Zusammenhänge von k, p und WKA noch experimentell weiter.

Dynamisches Anpassen von k und p

Genau wie SKEY in Abschnitt 3.3.8.2 unterstützt auch ESAWN prinzipiell ein sich zur Laufzeit des Sensornetzes variierendes k sowie auch ein variierendes p. Falls der Benutzer entscheidet, daß vorab bestimmte k, p nicht ausreichen, a prio gewählte k, p zu hoch für den aktuellen Angreifer gewählt worden sind oder er die WKA des Netzes ändern möchte, kann er die bisherigen k, p durch $k' \neq k, p' \neq p$ austauschen.

Dazu werden alle Knoten im Sensornetz über die neuen k', p' auf sichere Weise informiert. Die Senke w als Wurzel eines Aggregationsbaums schickt stellvertretend für den Benutzer $w \rightarrow i : E_{K_{w,i}}(k', p')$ an alle Knoten i im Sensornetz. Daraufhin benutzen alle Knoten im Sensornetz ESAWN mit den neuen k', p'. Auf diesen Aspekt des dynamischen Anpassens von k und p während des Sensornetz-Betriebs ist allerdings in dieser Arbeit nicht weiter eingegangen worden.

4.3.7 Diskussion über die Vollständigkeit von ESAWN

ESAWN betrachtet genau wie bisherige Arbeiten immer nur die Problematik, daß ein versuchter Betrug beziehungsweise die Fälschung eines Aggregats im Sensornetz entdeckt werden soll. ESAWN garantiert, daß legitime Knoten gefälschte Aggregate entdecken und daraufhin „Alarm“ melden. Damit ist ESAWN, um es mit der Begrifflichkeit eines Logik-Kalküls formal auszudrücken, *vollständig*: Jeder versuchte Betrug, jede versuchte Fälschung, die auch tatsächlich stattfindet, wird von ESAWN entdeckt. Allerdings ist ESAWN nach *dieser* Terminologie nicht korrekt: Falls ESAWN einen Betrug feststellt, ein „Alarm“ von einem Knoten gemeldet worden ist, dann muß nicht zwangsläufig auch ein Betrug stattgefunden haben (false-positive). Beispielsweise könnte ein korrumpierter Knoten einfach grundlos eine Alarm-Nachricht versenden, obwohl im Netz sämtliche Aggregation authentisch aggregiert worden sind.

Diese Überprüfung, ob ein gemeldeter Betrug *tatsächlich*, das heißt in der Wirklichkeit, stattgefunden hat, gestaltet sich als äußerst schwierig. Bei der zum Einsatz kommenden rein symmetrischen Kryptographie scheitert derjenige, der diese Überprüfung durchführt, zum Beispiel der Benutzer, an einem Zuordnungsproblem. Ein legitimer Zeuge meldet einen Betrug, wenn er von einem korrumpierten Aggregationsknoten ein Aggregat agg' empfängt, das nicht seinen eigenen Vorberechnungen agg entspricht. Um einen Betrug zu

begründen, müßte der Zeuge den Empfang von agg' vom Aggregationsknoten allerdings dem überprüfenden Benutzer beweisen können. Das allerdings geht bei symmetrischer Kryptographie nicht. Der korrumpierte Knoten würde dem Benutzer gegenüber behaupten, das korrekte agg an den Zeugen verschickt zu haben und unterstellt damit, daß der Zeuge grundlos die Alarm-Nachricht versendet hat. Dadurch kann der Benutzer letztlich nicht zuordnen, ob wirklich ein Betrug bei der Aggregation stattgefunden hat und welcher der beiden Knoten betrügt und welcher nicht.

Abhilfe könnte hier ein Mehrheitsentscheid, ein Wahlverfahren schaffen, in dem sämtliche an der Verifikation eines Aggregatsknoten beteiligte Zeugen darüber abstimmen, ob tatsächlich falsch aggregiert worden ist oder nicht. Genauer: Bei einem Mehrheitsentscheid würden sich die Knoten durch ein Wahlverfahren auf das korrekte Aggregat einigen, so daß Fälschungen damit gar nicht mehr möglich wären. Alle legitimen Knoten würden das korrekte Aggregat kennen.

Solche Wahlverfahren sind allerdings aus Energiesicht extrem teuer – die Arbeit Lamport et al. [136] gibt dafür einige Schranken an. Sei M die Menge aller an einer Aggregation und deren Verifikation beteiligter Knoten. Falls sich unter diesen Knoten in M insgesamt k korrumpierte Knoten befinden, dann muß $|M| > 3k$ gelten. Nur wenn diese Ungleichung erfüllt wird, dann können die $(|M|-k)$ legitimen Zeugen der Menge M tatsächlich feststellen, ob ein Betrug stattgefunden hat – und vor allem *ob nicht*. Die $(|M| - k)$ legitimen Knoten würden im Falle eines Betrugs einen Alarm an die Senke schicken. Damit gilt: Wenn die Senke weniger als $(|M| - k)$ Alarm-Nachrichten über den stattgefundenen Betrug eines einzelnen Aggregationsknoten erreichen, dann weiß die Senke, daß es *keinen* Betrug gegeben hat.

Es ist relativ offensichtlich, daß diese Form von Korrektheit, erheblich Energie kostet. Anstatt bei k korrumpierten Knoten wie bisher k Zeugen auszuwählen, müssen jetzt $k' = 3k$ Zeugen für ESAWN ausgewählt werden. Ein Erhöhen von k in ESAWN erhöht aber die Anzahl der zu versendenden Chiffrate und damit die Nachrichten erheblich – das werden die Simulationen noch verdeutlichen. Erschwerend kommt hinzu, daß der Vorgang eines einzelnen solchen Mehrheitsentscheids selbst auch extrem viel Energie kostet. Dafür müssen

$$\prod_{i=1}^{k'+1} (|M| - i) = (|M| - 1)(|M| - 2) \cdots (|M| - (k' + 1))$$

bestimmte „Informationen“ mit Hilfe von Nachrichten zwischen den Knoten in M ausgetauscht werden wobei die Größe dieser Informationen auch noch mit $O(k')$ skaliert. Der Aufwand an zusätzlich zu versendenden Nachrichten steigt damit erheblich, näheres dazu findet sich in Lamport et al. [136]. Unter der Annahme, daß Knoten sichere Broadcasts durchführen können, verringert sich der Aufwand auf $O((k')^2)$ Nachrichten der Größe $O(|M|k' + (k')^3 \log |M|)$ Bits, vergleiche Srikanth und Toueg [214]. Dieser Aufwand ist immer noch sehr hoch, und sichere Broadcast werden in dieser Arbeit als nicht gegeben angenommen.

Daß solche Wahlverfahren zur Sicherung von Aggregation in Sensornetzen ungeeignet sind, folgern Achtzehn et al. [3]. Die Autoren untersuchen und implementieren eine abgeschwächte Form eines Mehrheitsentscheids für Sensornetze, bei dem in einem Sensornetz mit $n = 7$ Knoten genau 2 korrumpiert sind. Die Abschwächung liegt in den Annahmen bezüglich der beiden korrumpierten Knoten:

- Ein Knoten sendet Daten, die erkennbar außerhalb einer Wertedomäne liegen, zum Beispiel $-500°C$ Temperatur, und
- der andere korrumpierte Knoten schickt gefälschte Werte symmetrisch an alle anderen Knoten. Symmetrisch bedeutet, er schickt allen anderen Knoten den gleichen Wert. Andere Betrugsformen sind nicht erlaubt.

In der Implementierung auf einer der MICA-Plattform ähnlichen Hardware [201] vergrößert sich selbst bei diesem sehr einfachen Protokoll die Größe der Nachrichten von ursprünglich 2 Byte Nutzdaten (Temperaturwert) pro Nachricht auf 20 Byte Nutzdaten pro Nachricht. Der Mehrheitsentscheid selbst benötigt dabei mit $n = 7$ Knoten insgesamt 3 s Zeit. In einem Netz mit vielen tausend Aggregationsknoten, deren Aggregation *jeweils* mit Hilfe eines solchen Mehrheitsentscheids überprüft wird, ist selbst diese schwache Form aus Energiesicht viel zu teuer und viel zu langwierig.

In vielen Szenarien reicht aber alleine die Tatsache aus, daß ein Alarm gemeldet wurde, um der Datensenke und dem Benutzer zu zeigen, daß es in der Tat ein Fehlverhalten in einem bestimmten Bereich des Sensornetzes gegeben hat. Der Benutzer weiß dann, daß in einem bestimmten Teilbaum des Aggregationsbaumes entweder tatsächlich ein Betrug stattgefunden oder aber ein korrumpierter Knoten einen Fehlalarm gemeldet hat. In jedem Fall liegt in diesem Teilbaum ein Fehlverhalten vor und der Benutzer kann dementsprechend darauf reagieren. Mögliche Reaktionen wären, daß dieser Teilbaum zum Beispiel zukünftig nicht mehr in die Aggregatberechnung mit einbezogen wird oder ähnliches.

Aufgrund des hohen Aufwands ist die obige Form möglichen Schutzes gegen false-positives nicht betrachtet worden.

4.3.8 Korrektheitsbeweis für ESAWN

An dieser Stelle folgt der formale Beweis der Korrektheit von ESAWN. Das Protokoll ESAWN soll verhindern, daß es bis zu k korrumpierten Knoten hintereinander auf einem Aggregationspfad, gelingt, ein Aggregat so zu fälschen und damit andere Knoten zu betrügen, daß dies von einem legitimen Knoten unbemerkt bleibt.

Zuerst der folgende Beweis für einen allgemeinen Baum B.

Definitionen

gegeben:	Parameter k
Definition:	**Baum** $B = (V, E)$, ein zyklenfreier, gerichteter Graph V Menge von Knoten E Menge gerichteter Kanten, so daß jeder Knoten genau einen **Vorgänger** mit seiner ausgehenden Kante bestimmt, außer der **Wurzel** w, die keinen Vorgänger besitzt. Knoten ohne eingehende Kanten heißen **Blätter**.
gegeben:	**Farbfunktion** $c : V \rightarrow \{\text{schwarz}, \text{weiss}\}$ Jeder Knoten aus V eines Graphen ist entweder schwarz oder weiß.

Definition: **Pfad** $\mathcal{P}$ in B
Eine Folge von Knoten $\mathcal{P} = v_1, v_2, \ldots, v_l$ mit $v_i \in B$ und v_i's ausgehende Kante zeigt auf Vorgänger v_{i+1}.
Die l Knoten $v_1, v_2, \ldots, v_l$ liegen **hintereinander** auf $\mathcal{P}$.

Definition: **Länge** eines Pfades $\mathcal{P} = v_1, v_2, \ldots, v_l$
Anzahl l der Knoten in $\mathcal{P}$

Definition: **Höhe** von Baum B
Die Länge des längsten Pfades in B

Definition **Entfernung** von zwei Knoten v_1, v_l
v_1 ist von v_l insgesamt $l - 1$ Knoten entfernt, falls ein Pfad $v_1, v_2, \ldots, v_l$ mit Länge l in B existiert.

Definition: **1-Vorgänger** eines Knotens v
Der Vorgänger von v

Definition: **r-Vorgänger**, $r > 1$
Vorgänger des $(r-1)$-Vorgängers

Definition: $(k+1)$**-Vorgänger-Abschluß** von B
Ein gerichteter Graph, der B enthält sowie zusätzliche Kanten von jedem Knoten zu seinem 1-Vorgänger, 2-Vorgänger,. . ., $(k+1)$-Vorgänger, falls es diese Vorgänger in B gibt.

Definition: **weißer Teilgraph** eines mit c gefärbten Graphen G
Der Teilgraph von G mit nur weißen Knoten sowie nur den Kanten zwischen den weißen Knoten.

Definition: **Grad** δ eines Baums B
Die maximale Anzahl eingehender Kanten jedes Knotens

Definition: Baum B ist **wurzelverbunden**,
falls für jeden Knoten aus B ein Pfad zur Wurzel w existiert.

Annahme A_k: Von beliebigen $(k+1)$ hintereinanderliegenden Knoten auf jedem Pfad ist mindestens einer weiß.

Behauptung: $A_k \Rightarrow$ für alle Bäume B gilt: Falls Wurzel w weiß, dann ist der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B wurzelverbunden.

Beweis: durch strukturelle Induktion über B

Induktionsanfang:
B besteht nur aus der Wurzel w

Beweis:
Falls Wurzel w weiß ist, dann ist der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B lediglich B selbst und damit verbunden.

Induktionsvoraussetzung:
Gegeben seien maximal δ Bäume $B_1, B_2, \ldots, B_\delta$, für die gilt:
$A_k \Rightarrow$ falls die Wurzel von Baum B_i weiß ist, dann ist der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B_i wurzelverbunden.

Induktionsschluß:
zu zeigen:
Für eine neue Wurzel w eines Baumes B mit den Teilbäumen $B_1, B_2, \ldots, B_\delta$ gilt: $A_k \Rightarrow$ falls Wurzel w weiß ist, dann ist der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B wurzelverbunden. Siehe hierzu auch Abbildung 4.9.

Betrachte: w ist weiß in B.

- Falls die Wurzel eines Teilbaums B_i weiß ist, dann gilt laut Induktionsvoraussetzung, daß alle weißen Teilgraphen des $(k+1)$-Vorgänger-Abschlusses von B_i selbst wurzelverbunden sind.
- Hilfslemma 1: Falls die Wurzel eines Teilbaums B_i schwarz ist, dann zerfällt der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B_i in jeweils wurzelverbundene Teilgraphen. $A_k \Rightarrow$ Die Wurzeln dieser weißen Teilgraphen sind in B_i maximal k Knoten von der Wurzel von B_i entfernt.
- (Hilfslemma 1 und A_k)$\Rightarrow$ der erste weiße Knoten in allen B_i ist höchstens $(k+1)$ Knoten entfernt von w.
- Der $(k+1)$-Vorgänger-Abschluß von B erzeugt eine Kante von diesem weißen Knoten zu w, vergleiche Abbildung 4.9.
- Der weiße Teilgraph des $(k+1)$-Vorgänger-Abschlusses von B bleibt verbunden.

□

Zusammenfassung

Dieser Beweis zeigt, wieso bei ESAWN kein Betrug von einem legitimen Knoten unentdeckt bleiben kann. Man betrachte irgendeinen Aggregationsbaum als den Baum B im Beweis. Die Sensorknoten im Aggregationsbaum werden durch die Knoten in B repräsentiert. Zwischen zwei Knoten in B existiert genau dann eine Kante, wenn im Aggregationsbaum eine Aggregationsbeziehung zwischen ihnen besteht.

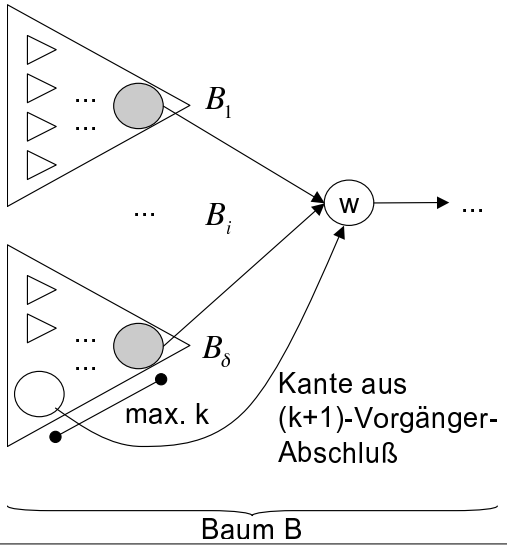


Abbildung 4.9 Zum Beweis über Teilbäume des gesamten Baums

Die weißen Knoten im Baum B stellen die legitimen Knoten im Aggregationsbaum dar. Die schwarzen Knoten in B repräsentieren die korrumpierten Knoten.

Annahme A_k ergibt sich aus der Annahme über das Angreifermodell, daß nicht mehr als k korrumpierte Knoten auf einem Aggregationspfad hintereinander korrumpiert sein dürfen.

Schließlich leistet das Protokoll ESAWN den im Beweis erwähnten $(k+1)$-Vorgänger-Abschluß in einem Aggregationsbaum.

Der Beweis zeigt demnach, daß jeder legitime Knoten im Aggregationsbaum bei der Verifikation des Aggregats eines Aggregationsknotens von mindestens einem zur Aggregation gehörenden, legitimen Quellsensor Daten erhält. Damit kann er einen möglichen Betrug des Aggregationsknotens feststellen.

4.4 Evaluierung

Genau wie bei SKEY soll auch die Skalierbarkeit und Einsatzfähigkeit von ESAWN bzgl. steigender Knotenzahlen n im Sensornetz untersucht werden. Nach einer zunächst theoretischen Analyse von ESAWNs Leistung bzgl. Energiebedarf und Sicherheit, folgt dazu in Abschnitt 4.4.3 die Auswertung verschiedener durchgeführter Simulationen.

4.4.1 Speicherverbrauch

Der von ESAWN benötigte Speicherverbrauch in Abhängigkeit der Gesamtknotenzahl n läßt sich relativ einfach ausrechnen. Grundsätzlich gilt: Sobald ein Aggregationsknoten x von seinen Quellsensoren $x_1, \ldots, x_\delta$ Daten $X_1, \ldots, X_\delta$ erhält und aus diesen ein Aggregat agg_x berechnet hat, kann er die X_i verwerfen und muß nur noch agg_x speichern.

Unter dieser Voraussetzung besteht bei ESAWN für einen Aggregationsknoten x auf Höhe h' im Aggregationsbaum in dem Moment der höchste Speicherbedarf, bei dem er

- $\delta - 1$ Aggregate seiner Nachfolgerknoten $x_1, \ldots, x_{\delta-1}$ auf Höhe $h' + 1$ gespeichert hat und nun dabei ist, das Aggregat für den Knoten x_δ zu berechnen. Knoten x hat im Moment demnach $\delta - 1$ Aggregate gespeichert.

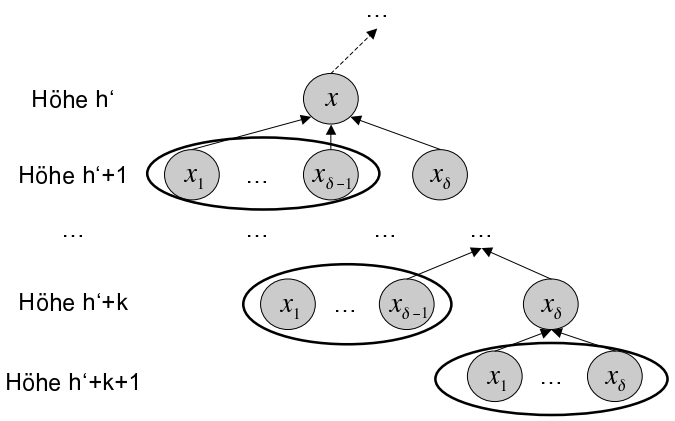


Abbildung 4.10 Moment des höchsten Speicherverbrauchs von ESAWN, Aggregate umrandeter Knoten befinden sich im Speicher von x, siehe Text

- Weiterhin hat x im Teilbaum von x_δ auf Höhe $h' + 2$, den direkten Nachfolgern von x_δ, ebenfalls alle bis auf ein Aggregat berechnet. Demzufolge speichert x weitere $\delta - 1$ Aggregate.
- Dies setzt sich fort bis zur Höhe $h' + k$ unterhalb von x. Für all diese k Höhen hat x jeweils $\delta - 1$ Aggregate gespeichert. Hinzukommt jetzt nur noch das letzte Aggregat auf Höhe $h' + k$, für das x nochmals δ-viele Aggregate der Nachfolger auf Höhe $h' + k + 1$ benötigt.

Diesen Moment des höchsten Speicherverbrauchs zeigt Abbildung 4.10. Die umrandeten Knoten deuten an, daß x deren Aggregate speichert. Insgesamt speichert x auf diese Weise maximal $k \cdot (\delta - 1) + \delta$ Aggregate.

Da k genau wie δ unabhängig von der Gesamtanzahl aller Knoten n konstant ist, wächst der Speicherverbrauch von ESAWN pro Knoten mit $O(1)$.

4.4.2 Energieverbrauch

Für ESAWN sind weiterhin die Energiekosten für zu verschickende Nachrichten und durchzuführende kryptographische Operationen von Interesse. Um den Energieverbrauch als wichtigstes Kostenkriterium von ESAWN besser einordnen zu können, soll er mit dem Energieverbrauch der eingangs erwähnten Datentransporte *authentische Nicht-Aggregation* und *nicht-authentische Aggregation* in Relation gesetzt werden.

Wie im Kapitel über Grundlagen beschrieben soll in dieser Arbeit von nicht-degnerierten Aggregationsbäumen, die normalverteilt um einen Erwartungswert δ viele Nachfolger besitzen, ausgegangen werden. Ein solcher Baum von insgesamt n Sensorknoten besitzt die Höhe $h \approx \log_\delta(1 + (\delta - 1)n) - 1$. Auf jeder Ebene h' dieses Baumes, wobei $0 \leq h' \leq h$, befinden sich durchschnittlich $\delta^{h'}$ Knoten.

4.4.2.1 Authentische Nicht-Aggregation

Ohne Datenaggregation läßt sich die notwendige Anzahl der zu versendenden Nachrichten für das komplette Netz wie folgt berechnen: Der Aggregationsbaum dient den einzelnen Knoten nur noch als Routing-Vorgabe in Richtung der Datensenke. Bei dieser Form des Datentransports werden die einzelnen Daten nicht aggregiert, sondern nur noch simpel weitergeleitet. Ein Aggregationsknoten funktioniert dabei nur als einfacher Routing-Knoten und leitet die von Kind-Knoten empfangenen Daten einfach an seinen Vaterknoten

weiter. Jedes Datum eines Blattes auf Ebene h muß über alle h Ebenen im Baum zur Senke transportiert werden. Bei δ^h Blättern im Baum fallen demnach insgesamt

$$h \cdot \delta^h \approx h \cdot \delta^{\log_\delta(1+(\delta-1)n)-1} \approx (\log_\delta(1+(\delta-1)n)-1)\frac{n(\delta-1)+1}{\delta} =: f(n,\delta)$$

Nachrichten für den gesamten Transport aller Daten zur Senke hin an. Da die Höhe des Baumes nur logarithmisch mit der Knotenzahl n steigt, skaliert die Anzahl der Nachrichten nicht exponentiell. Weil der folgende Grenzwert existiert:

$$\lim_{n\to\infty} \frac{f(n,\delta)}{n \log_\delta n} = \frac{\delta-1}{\delta} < \infty,$$

folgt, daß $f(n,\delta)$ höchstens so stark steigt wie $n \log_\delta n$, genauer:

$$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}, \forall n > n_0 : f(n,\delta) \leq c \cdot n \log_\delta n.$$

So steigt die Anzahl der gesamten Nachrichten $f(n,\delta)$ insgesamt mit $O(n \log n)$. Pro Knoten steigt sie folglich mit $O(\log n)$.

Ver- und Entschlüsselt wird dabei jeder Meßwert nur genau einmal: verschlüsselt beim Versenden am Sensorknoten sowie entschlüsselt beim Empfang durch die Senke. Im kompletten Netz kommt es daher zu δ^h Verschlüsselungen sowie δ^h Entschlüsselungen – pro Knoten $O(1)$ Operationen.

4.4.2.2 Nicht-Authentische Aggregation

Im Vergleich dazu fallen bei der unsicheren Aggregation ohne irgendwelche Authentizitäts- oder Korrektheitsverifikationen deutlich weniger Nachrichten an. Die Routing-Knoten funktionieren nun wieder als „richtige“ Aggregationsknoten. Bei Aggregation sendet jeder Knoten nur genau *einmal*. Genauer: Er versendet sein Aggregat. Insgesamt sind daher im Netz $(n-1) \in O(n)$ Nachrichten notwendig, beziehungsweise pro Knoten $O(1)$ Nachrichten. Dies ist deutlich weniger verglichen mit keiner Aggregation.

Ver- oder Entschlüsselungen fallen bei dieser Form des Datentransports nicht an.

4.4.2.3 Sichere Aggregation mit ESAWN

In Gegenwart von k korrumpierten Knoten und demnach k Zeugen bei einer Verifikation fallen wesentlich mehr Nachrichten als bei der nicht-authentischen Aggregation im Netz an. Zu bedenken ist dabei allerdings, daß ESAWN Chiffrate verschickt – und zwar mehrere Chiffrate in ein oder mehrerer Nachrichten.

Jeder Knoten x im Netz

- sendet sein Aggregat, beziehungsweise dessen Chiffrat an seinen direkten Vorgängerknoten im Aggregationsbaum,
- sendet k Chiffrate für die k Zeugen an seinen Vorgängerknoten zur Weiterleitung,
- leitet von seinen Nachfolgern auf den Höhen $1, \ldots, k$ unter ihm Chiffrate an seine Vorgänger im Baum weiter. Das heißt, Knoten x muß von den Knoten auf Höhe $i = 1, \ldots, k$ unterhalb von sich die Chiffrate der sich dort befindenden δ^i Knoten Chiffrate weiterleiten. Für jeden Knoten auf Höhe i unterhalb von x muß x jeweils $(k-i+1)$ Chiffrate weiterleiten.

Insgesamt versendet jeder Knoten bei ESAWN *maximal*

$$(k+1) + \sum_{i=1}^{k} \delta^i (k-i+1) = (k+1) + \frac{\delta^{k+2} - k\delta^2 + k\delta - \delta^2}{(\delta-1)^2}$$

Chiffrate. Dabei handelt es sich deshalb um die *maximale* Anzahl von Chiffraten, weil diese nur in dem Fall verschickt wird, falls sich Knoten x auf eine Höhe im Aggregationsbaum aufhält, so daß noch mindestens k Knoten oberhalb und unterhalb von ihm im Baum liegen. Liegt der Knoten relativ weit oben oder weit unten im Baum, muß k durch $k' < k$ ersetzt werden. So müssen beispielsweise die Blätter überhaupt keine Chiffrate anderer Knoten weiterleiten. Weiterhin werden mehrere Chiffrate in eine Nachricht zusammengepackt und gleichzeitig verschickt. Wieviele Nachrichten tatsächlich in ESAWN verschickt werden, und damit der Energieverbrauch, hängt folglich mit dem Format der Nachrichten, genauer mit der mögliche Menge an Chiffraten pro einzelner Nachricht zusammen. Hierauf gehen die Simulationen noch ein. Da aber k ausschließlich von den Fähigkeiten des Angreifers abhängt und δ konstant ist, kann man festhalten, daß die Anzahl der Nachrichten in ESAWN mit steigender Gesamtknotenzahl n im Netz mit $O(1)$ pro Knoten skaliert.

Alle Knoten, bis auf die Senke, verschlüsseln ihre eigenen Meßwerte beziehungsweise Aggregate, sowie entsprechend die k Kopien für die Zeugen. ESAWN benötigt insgesamt $(n-1)(k+1)$, das sind pro Knoten $k+1$ Verschlüsselungsoperationen.

Jeder Knoten x empfängt von all seinen Nachfolgern auf den Höhen $1, 2, \ldots, k+1$ *unterhalb* von x jeweils ein Chiffrat, das er entschlüsseln muß. Das sind insgesamt

$$\sum_{i=1}^{k+1} \delta^i = \frac{\delta^{k+2} - \delta}{\delta - 1}$$

Entschlüsselungen. Auch dieser Ausdruck ist unabhängig von n, damit skaliert die Anzahl der Entschlüsselungen mit $O(1)$.

ESAWNs Energieaufwand skaliert mit konstantem $O(1)$ pro Knoten.

Auswirkungen von p auf den Energieverbrauch

Falls jede Aggregation anstatt jedesmal nur mit einer bestimmten Wahrscheinlichkeit $p \leq 100\%$ überprüft wird, sinkt dementsprechend auch der Energieverbrauch von ESAWN um $(1-p)\%$. Gegenüber „normalem“ ESAWN mit $p = 100\%$ müssen $(1-p)\%$ weniger Aggregate ver- oder entschlüsselt und $(1-p)\%$ weniger Nachrichten verschickt werden.

Konfigurationen ergeben partielle Ordnung $\succcurlyeq_{\mathrm{E}}$

Für die Menge der möglichen Konfigurationen (k, p) von ESAWN, das heißt die jeweilige Wahl der Parameter k und p, existiert in Bezug auf deren resultierenden *Energieverbrauch* eine partielle Ordnung $\succcurlyeq_{\mathrm{E}}$.

Für den Energieverbrauch von Konfigurationen (k, p) von ESAWN gilt:

$$\begin{gathered} k \geq k' \\ \Rightarrow (k,p) \succcurlyeq_{\mathrm{E}} (k',p) \end{gathered}$$

$$\begin{gathered} p \geq p' \\ \Rightarrow (k,p) \succcurlyeq_{\mathrm{E}} (k,p') \end{gathered}$$

und

$$k \geq k', p \geq p'$$
$$\Rightarrow (k,p) \succcurlyeq_{\mathrm{E}} (k',p')$$

Der Benutzer kann damit zum Erhöhen der WKA entweder k oder p erhöhen, dadurch steigt der Energieverbrauch von ESAWN. Er kann allerdings noch nicht abschätzen, ob entweder das Erhöhen von k oder das von p aus Energiesicht günstiger wird. Dies zeigt erst Abschnitt 4.4.3.2. Damit wird $\succcurlyeq_{\mathrm{E}}$ in Abschnitt 4.4.3.2 zu einer totalen Ordnung.

4.4.3 Simulation

Zur weiteren Analyse seiner Leistung ist im Rahmen dieser Arbeit auch ESAWN implementiert worden [244]. Genau wie in Abschnitt 3.4.3.1 hat der diskrete Ereignissimulators GloMoSim Sensornetze verschiedener Größe und verschiedener Konfigurationen simuliert. Die Parameter der durchgeführten Simulationen, beispielsweise über Annahmen zur Aggregation, den Aggregationsgrad, Aggregationsbaum, das Routing usw., entsprechen denen aus Abschnitt 3.4.3.1, S. 93.

Untersucht worden ist dabei zunächst der *Energieverbrauch* von ESAWN sowie anschließend die *Sicherheit* ESAWNs, insbesondere in Bezug auf WKA und die Wahl von k und p.

4.4.3.1 Energieverbrauch

Simuliert und gemessen worden ist der Energieverbrauch so genannter *kompletter Aggregationen*: Eine komplette Aggregation beinhaltet den Datentransport sämtlicher Meßwerte von den Blätter durch das gesamte Sensornetz hindurch zur Senke.

Zur besseren Einschätzung der für ESAWNs Datentransport anfallenden Energie, ist diese in Relation gesetzt worden zu den beiden Datentransporten *authentische Nicht-Aggregation* sowie *nicht-authentische Aggregation*. Dafür hat die Simulation, genau wie bei SKEY, zufällig verschiedene Aggregationsbäume mit unterschiedlicher Gesamtknotenzahl n und Grad δ generiert. Auf jedem dieser Aggregationsbäume sind dann die drei Arten von Datentransporten simuliert und verglichen worden.

Um den Energieverbrauch zu messen, hat in den Simulationen jedes Blatt des Aggregationsbaums einen Meßwert zu seinem Aggregationsknoten geschickt. Die Aggregationsknoten haben dann entsprechend ihre Aggregationsfunktion abgearbeitet und die entstehenden Aggregate weitergeleitet. Da ausschließlich der Energieverbrauch gemessen werden sollte, den auf Protokollebene die drei Datentransporte implizieren, und um die Ergebnisse nicht durch Kollisionen auf MAC-Schicht zu verfälschen, haben zunächst die Blätter auf Höhe h hintereinander ihre Meßwerte an die Aggregationsknoten auf Höhe $(h-1)$ geschickt. Diese haben Aggregate gebildet, an die Aggregationsknoten auf Höhe $(h-2)$ geschickt usw. In der Simulation von *authentischer nicht-Aggregation* haben die (Aggregations-)Knoten jede empfangene Nachricht einfach nur weitergeschickt. Auf diese Weise ist der gesamte Energieverbrauch für einen kompletten Durchlauf aller Aggregationen im Netz gemessen worden. Die Gesamtenergie hat sich in den Simulationen zusammengesetzt aus der Energie zum Versenden von Daten sowie aus der durch die anfallenden kryptographische Operationen Ver- und Entschlüsselung.

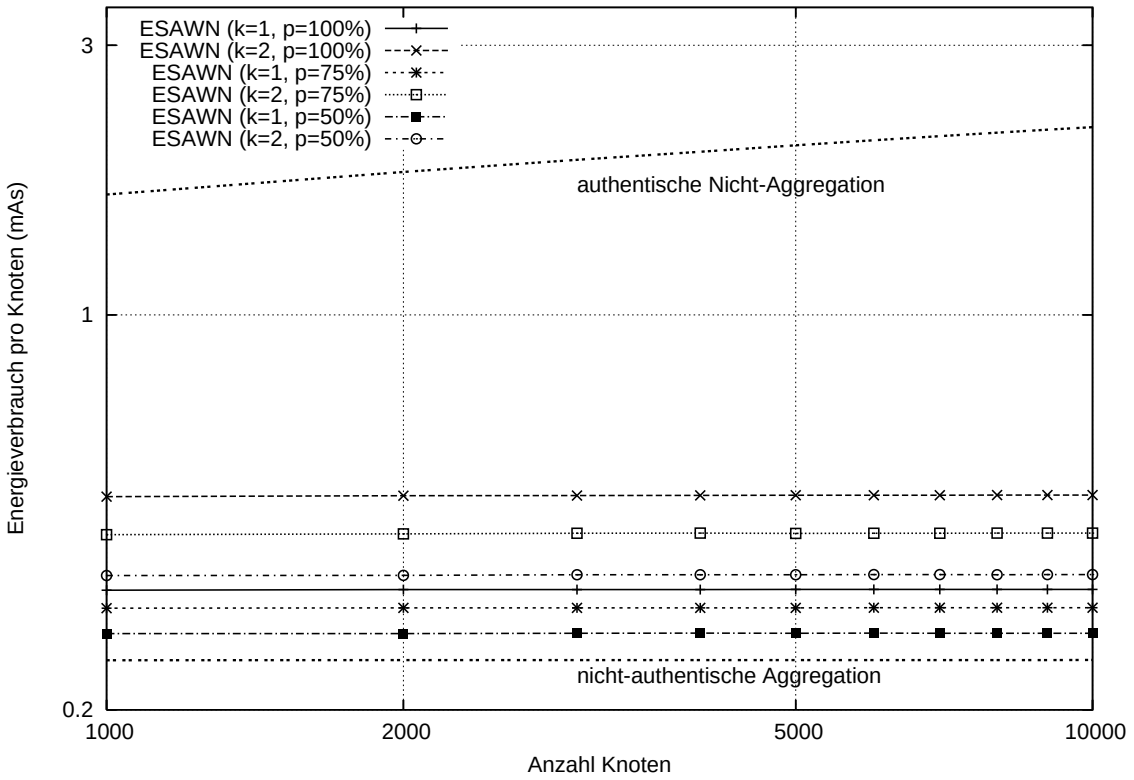


Abbildung 4.11 Energiekosten für komplette Aggregation, $\delta = 3$

Falls die Daten nicht wie in den Simulationen zeitlich hintereinander, sondern beliebig, zufällig versendet werden, kann dies folgende Auswirkungen haben: Da der Zugriff auf das Funkmedium zufällig geschieht, können einzelne Sensoren durch wiederholte Kollisionen „verhungern“ (engl. *Starvation*, eine Liveness-Eigenschaft eines Protokolls, s. Alpern und Schneider [7]). Das bedeutet, daß einzelne Sensoren ihre Nachricht theoretisch, wenn auch unwahrscheinlich, niemals übertragen können. Durch wiederholte Kollisionen auf dem Funkmedium kann sich der Transport der Daten zur Senke daher extrem, theoretisch unendlich, verzögern. Durch das ständige Versuchen, Nachrichten neu zu übertragen, steigt außerdem der Energieverbrauch zum erfolgreichen Übertragen von Nachrichten – theoretisch unendlich. Die tatsächlich notwendige Zeit zum Transport von Daten und auch der dafür notwendige Energieverbrauch hängen stark von der Art und Weise der Aggregation, den Zeitpunkten, wann einzelne Sensoren Nachrichten verschicken müssen, der Anwendung usw., ab. Daher sind in dieser Arbeit solche Aspekte nicht näher betrachtet und vom idealisierten, zeitlich verzögerten Versand von Nachrichten pro Höhe im Aggregationsbaum ausgegangen worden. Einige weitere Untersuchungen zum Verhalten von ESAWN in Bezug auf die 802.11 MAC-Schicht finden sich allerdings noch in Anhang C.

Simulationsergebnisse

Abbildung 4.11 zeigt den Gesamtbedarf an Energie *pro Knoten* für einen wie oben beschriebene *komplette Aggregation*. Die Abbildung zeigt dabei absolute Werte, das heißt in *mAs* Energieverbrauch. Für dieses Beispiel sind Aggregationsbäume mit variierendem n, verschiedenen k und einem erwarteten Aggregationsgrad von $\delta = 3$ simuliert worden. Sowohl die x- als auch die y-Achse skalieren logarithmisch in der Abbildung. Der Energieverbrauch für die obere Kurve *authentische Nicht-Aggregation* steigt wie erwartet

an, da mit zunehmender Anzahl Knoten jeder Knoten mehr Daten seiner Vorgänger im Aggregationsbaum in Richtung Senke weiterleiten muß. Hingegen bleibt der Verbrauch sowohl von *nicht-authentischer Aggregation* als auch von sämtlichen ESAWN-Konfigurationen pro Knoten invariant gegenüber steigender Anzahl Knoten n, wie auch theoretisch erwartet: Bei festem δ verrichtet ein Aggregationsknoten unabhängig von n immer die gleiche Arbeit. Wie zu erwarten kostet eine steigende Prüfwahrscheinlichkeit mehr Energie als eine geringere. Genauso erhöht sich mit einem mächtigeren Angreifer, der mehr Knoten korrumpiert, auch gleichzeitig k und damit der Energieverbrauch. Bei steigendem k müssen mehr Nachrichten verschickt und mehr Chiffrate ver- und entschlüsselt werden.

Die nun folgenden Diagramme zeigen ab jetzt nicht mehr den totalen Energieverbrauch, sondern nur noch den Energieverbrauch von ESAWN relativ zum Datentransport *authentische Nicht-Aggregation*. Aus Mangel an Vergleichsmöglichkeiten mit anderen, bisher veröffentlichten Arbeiten erscheint der *absolute* Energieverbrauch nicht so interessant wie der relative Energieverbrauch zu *authentische Nicht-Aggregation* bei gleicher Netzkonfiguration. Der Vergleich der *Einsparungen* von Aggregation beziehungsweise ESAWN durch verschiedene k und p können mit einer relativen Darstellung wesentlich prägnanter visualisiert werden: Diese ermöglicht sofort Aussagen wie „Die Konfiguration X kostet nur Y Prozent von *keiner Aggregation*.“ Weiterhin ist in den folgenden Graphen auch noch der Energieverbrauch von *nicht-authentischer Aggregation* eingezeichnet. Dadurch läßt sich die Leistung von ESAWN noch besser beurteilen – „wieviel mehr kostet ESAWN im Vergleich zu keiner Sicherheit?“

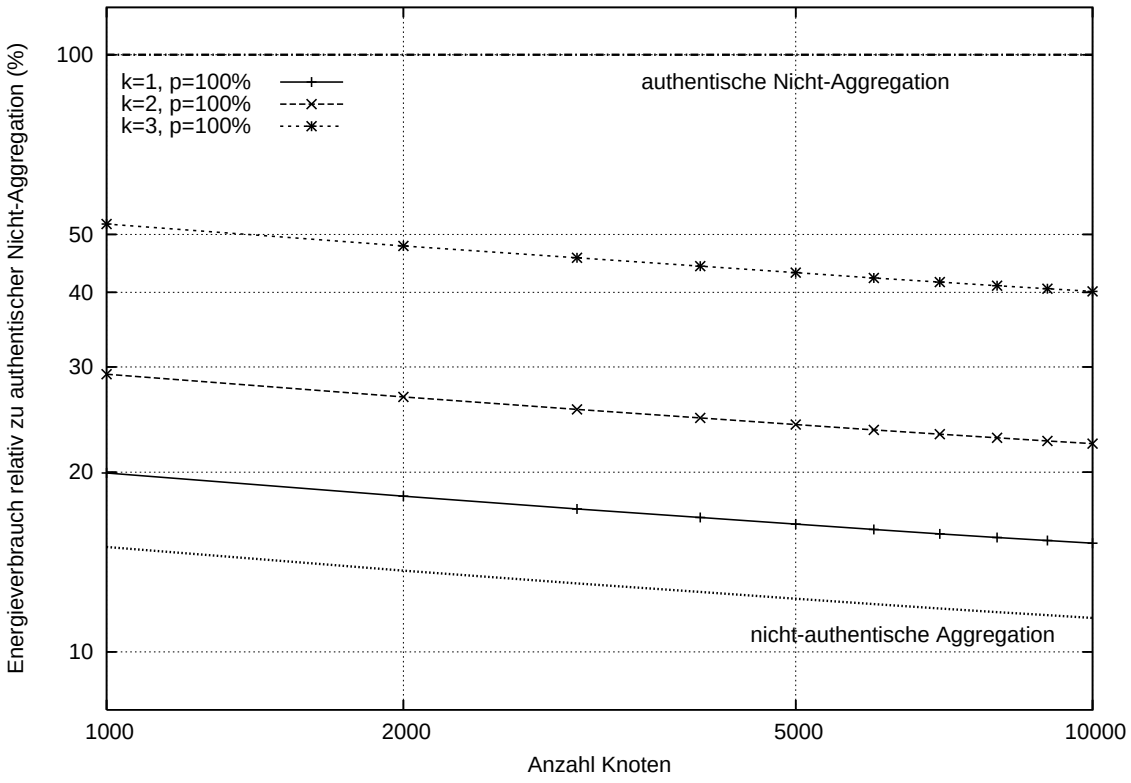
Variieren von k und p

In Abbildung 4.12 ist daher auch der Energiebedarf verschiedener ESAWN-Konfigurationen k und p bei steigender Gesamtknotenzahl n jeweils relativ zu nicht *authentischer Nicht-Aggregation* zu sehen. Grundsätzlich sinken alle Kurven des *relativen* Energieverbrauchs von ESAWN mit zunehmender Gesamtzahl n an Knoten, da bei steigendem n die Energieeinsparungen durch Aggregation gegenüber keiner Aggregation steigen. Bei gleichbleibender Konfiguration sinkt dementsprechend der *relative* Energieverbrauch.

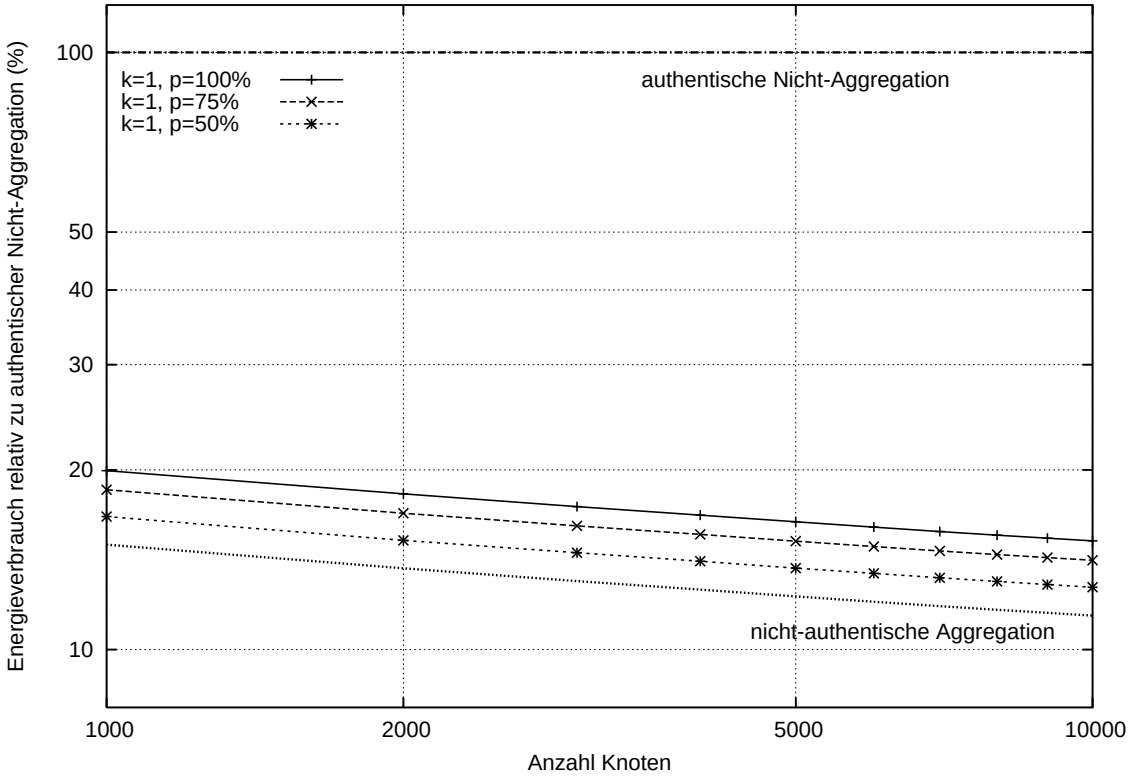
Der Energieverbrauch von ESAWN steigt in der Simulation weniger als proportional mit zunehmendem k, das heißt: $k = 2$ kostet weniger als doppelt so viel Energie wie $k = 1$ bei gleichem p. Dies zeigt Abbildung 4.12(a). Der Grund dafür ist, daß der Energieverbrauch von der verwendeten TinyOS-Nachrichtengröße abhängt: In jede TinyOS-Nachricht mit 29 Byte Nutzlast passen drei 64 Bit RC5 Chiffrate, siehe Abschnitt 3.4.3.1, S. 94, so daß TinyOS erst bei mehr als drei Chiffraten ein (oder mehrere) zusätzliche Nachrichten verschicken muß. Erst wenn eine TinyOS-Nachricht komplett mit Chiffraten „gefüllt“ ist, schickt TinyOS eine weitere Nachricht.

Auch kostet eine ESAWN-Konfiguration mit beispielsweise $p = 50\%$ nicht die Hälfte der Energie von $p = 100\%$, wie man vielleicht zunächst annimmt, sondern etwas mehr als die Hälfte – siehe Abbildung 4.12(a). Dies liegt ganz einfach daran, daß ESAWN unabhängig von jeder Prüfwahrscheinlichkeit auf jeden Fall zumindest die Meßwerte und Aggregate durch das Netz zur Senke transportiert. Der skizzierte Energieverbrauch zählt das selbstverständlich mit.

Wie man aus Abbildung 4.12(a) ablesen kann, ermöglicht ESAWN selbst bei 100% Überprüfung und $k = 3$ korrumpierten Knoten auf einem Aggregationspfad noch deutliche Energieeinsparungen, teilweise deutlich über 50%, gegenüber nicht-aggregierendem Datentransport. Obwohl $k = 3$ zunächst als sehr klein erscheint, bedeutet dies doch schon



(a) Variierendes k



(b) Variierendes p

Abbildung 4.12 Relative Energiekosten von ESAWN, verschiedene k und p, $\delta = 3$

einen großen Anteil korrumpierten Knoten auf dem Aggregationspfad. Beispiel: Selbst

bei einem Sensornetz mit $n = 10000$ Knoten verfügt ein durchschnittlicher Aggregationsbaum bei Grad $\delta = 3$ über eine Höhe von $h = 9$ Knoten. Von diesen 9 Knoten werden laut Annahme über den Angreifer die Blätter jedoch niemals bezüglich der Authentizität ihrer Meßwerte betrügen. Mit $k = 3$ Zeugen kann man sich demnach gegen 3 korrumpierten Knoten von insgesamt 8 möglichen, entsprechend $\beta = 37,5\%$(!), verteidigen. Auf dieses Verhältnis wird im nächsten Abschnitt noch im Detail eingegangen.

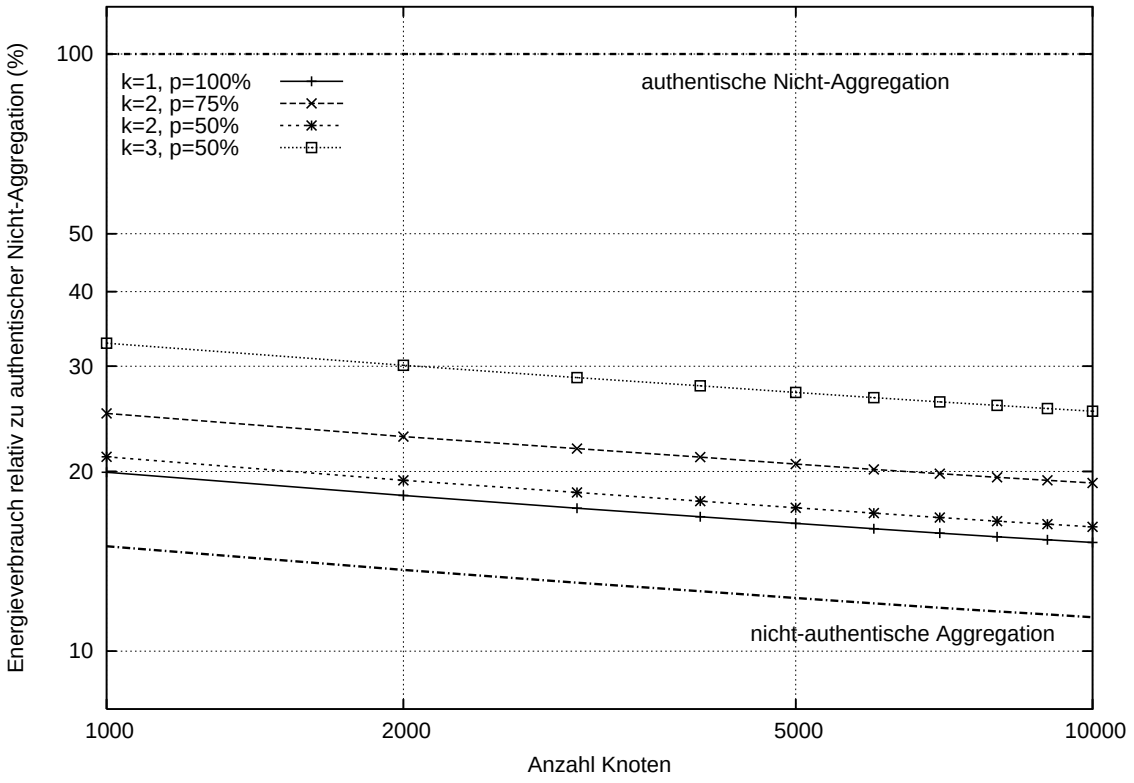


Abbildung 4.13 Relative Energiekosten, variierende k und p

Aus Abbildung 4.13 läßt sich ablesen, daß durch höheres k mehr Energie verbraucht wird, als durch geringeres p eingespart wird. Die Konfiguration mit $k = 2, p = 50\%$ kostet vergleichsweise mehr Energie als die mit $k = 1, p = 100\%$. Der Parameter p hat einen geringeren Einfluß auf den Energieverbrauch als k. Der Parameter p hat, wie am Anfang dieses Abschnitts theoretisch hergeleitet, nur einen linearen Einfluß auf die Anzahl der zu versendenden Chiffrate, der Parameter k einen exponentiellen.

Daher läßt sich die folgende Quintessenz zusammenfassen: Gegen weniger korrumpierte Knoten jeweils häufiger zu überprüfen ist energetisch günstiger, als gegen mehr korrumpierte Knoten weniger häufig zu überprüfen. Wenn der Benutzer seine ESAWN Konfiguration p und k festlegt, sollte er diese Tatsache mit in seine Überlegungen einbeziehen. Dies wird noch in Abschnitt 4.4.3.2 von Bedeutung sein.

Variieren von δ

Abbildung 4.14 vergleicht der Vollständigkeit halber die möglichen Auswirkungen verschiedener Aggregationsgrade $\delta = 4, 5, 6$ miteinander. In diesem Graph sind die Kurven für *nicht-authentische Aggregation* bewußt weggelassen, um die Skizze nicht zu überfüllen. Sonst wären für *jedes* δ noch *jeweils* eine weitere Kurve als untere Schranke notwendig gewesen. Die verschiedenen abgebildeten ESAWN-Energiekurven sind aber mit

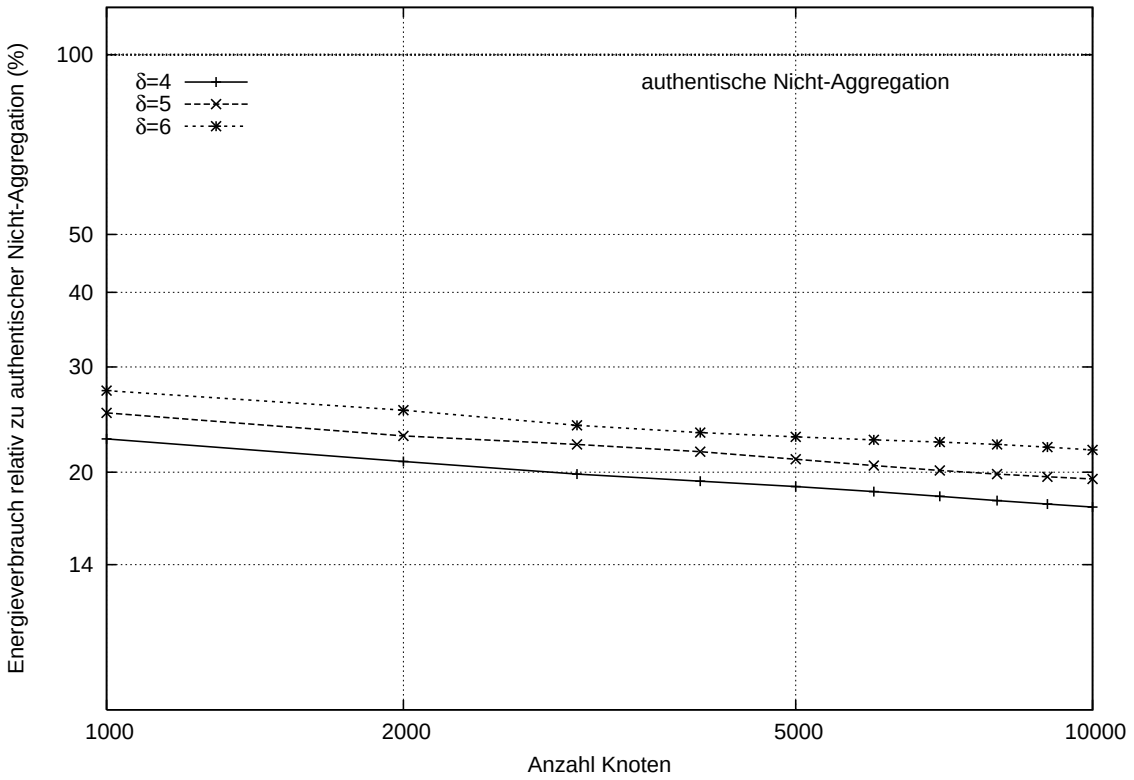


Abbildung 4.14 Relative Energiekosten, variierendes δ, $k = 1, p = 100\%$

ihren jeweiligen Energieverbräuchen zu *authentischer Nicht-Aggregation* in Relation gesetzt worden.

Hier zeigt sich zunächst, daß bei konstant gehaltener ESAWN-Konfiguration, das ist die Wahl der Parameter k und p, ein größeres δ auch einen höheren Energieverbrauch impliziert. Das liegt daran, daß bei größerem δ die relativen Einsparungen von Aggregation sich gegenüber nicht-Aggregation weniger stark auswirken. Die breiteren Aggregationsbäume fassen zwar mehr Daten zu einem Aggregat zusammen, allerdings sinkt auch die Baumhöhe durch die zunehmende Breite. Bei geringer Baumhöhe müssen dann die Knoten innerhalb des Baumes auch insgesamt weniger Daten Richtung Senke weiterleiten – ein größeres δ spart demnach bei nicht-Aggregation Energie ein. Der relative Gewinn von Aggregation gegenüber nicht-Aggregation wird dadurch geschmälert.

Der Unterschied im Energieverbrauch zwischen variierenden δ's stellt sich allerdings, verglichen mit verschiedenen k's, als relativ gering dar. Zwischen beispielsweise $\delta = 4$ und $\delta = 6$ liegen nur ein paar wenige Prozentpunkte. Außerdem handelt sich bei dem Aggregationsgrad wahrscheinlich um einen Parameter, den der Benutzer nur in seltenen Fällen beeinflussen kann, sondern den die Gegebenheiten seines speziellen Szenarios vordefinieren. Zur Erinnerung: Der durchschnittliche Aggregationsgrad δ gibt an, wieviele Knoten im Durchschnitt Quellsensoren eines Aggregationsknotens sind.

4.4.3.2 Sicherheit von ESAWN – WKA

Schließlich bleibt noch die Frage zu klären, welche Energiekosten bei einer vom Benutzer gewünschten, vorgegebenen *Wahrscheinlichkeit für eine Korrekte Aggregation* (WKA) durch ESAWN entstehen. Und genauso umgekehrt: Gibt der Benutzer maximal auszuge-

bende Energiekosten vor, stellt sich die Frage, welche WKA ESAWN damit letztendlich erreicht.

In beiden Fällen will der Benutzer neben den zu erwartenden Energiekosten beziehungsweise der WKA außerdem auch noch die dafür notwendigen Werte k und p bestimmen, mit denen er ESAWN parametrisieren muß. Auch dieses Problem, der Zusammenhang zwischen k, p, dem Energieverbrauch und der damit erzielten WKA, ist Gegenstand von Simulationen gewesen.

Die Abbildungen 4.15 und 4.16 zeigen die Simulationsergebnisse. Vor deren Interpretation folgt jedoch zunächst, was und wie simuliert worden ist: Bei einer vorgegebener Konfiguration, feststehenden Parametern n, β, und δ, hat die Simulation genau wie im vorigen Abschnitt zufällig mehrere Aggregationsbäume erzeugt und darauf eine komplette Aggregation durchgeführt: Alle Blätter haben ihre Meßwerte an ihre Aggregationsknoten geschickt, diese haben aggregiert und wieder eine Stufe weiter in Richtung Senke geschickt usw.

Am Ende eines solchen Simulationslaufes hat der Simulator entscheiden können, ob es *mindestens einem* korrumpierten Knoten erfolgreich gelungen ist, unbemerkt ein Aggregat zu fälschen, also ein legitimer Knoten ein gefälschtes Aggregat ohne es zu merken weiterverarbeitet hat. Solche erfolgreichen Betrugsversuche bleiben in Abhängigkeit von der konkreten Wahl der Sicherheitsparameter k und p *immer* zufällig möglich, da durch die zufällige Generierung des Aggregationsbaumes auch zufällig mehr als k korrumpierte Knoten auf einem Aggregationspfad hintereinanderliegen können. Die Frage ist jedoch, wie sich die Wahrscheinlichkeit mindestens eines erfolgreichen Betrugsversuchs zu k und p verhält. Die Simulation hat dementsprechend, bei mehreren Durchläufen pro Konfiguration k und p, die Wahrscheinlichkeit WKA' überprüft, bei der *kein* Betrug unentdeckt bleibt. Ziel der Simulationen ist gewesen, k und p so zu bestimmen, daß die sich daraus ergebene WKA' *mindestens* so groß wird wie die vorgegebene WKA.

Die Parameter k und p haben sich dabei für jede vorgegebene Netzkonfiguration (n, δ, β) und eine vom Benutzer gewünschte WKA nach Algorithmus 17 bestimmt. Dabei bevorzugt Algorithmus 17 die Inkrementierung von p vor der Inkrementierung von k, um höhere Sicherheit (WKA') zu erreichen. Dies hat zwei Ursachen:

1. Grundsätzlich kann WKA' auf zwei verschiedene Weisen erhöht werden: Durch Erhöhen von p verringert sich trivialerweise die Wahrscheinlichkeit, daß korrumpierte Knoten unbemerkt Aggregationen fälschen. Damit erhöht sich WKA'. Durch Erhöhen von k vergrößert sich die Wahrscheinlichkeit, daß in einem zufälligen Aggregationsbaum alle Aggregationspfade über eine „ausreichende“ Anzahl von Zeugen verfügen. Ausreichend bedeutet, daß in jedem Pfad, in dem k' korrumpierte Knoten hintereinander liegen auch $k > k'$ Zeugen zum Einsatz kommen. Damit erhöht sich auch WKA'.

   Durch unterschiedliche Wahl von k und p bei festen Netzkonfigurationen (n, δ, β) kann man demnach im Durchschnitt die selbe WKA' erreichen. Beispielsweise könnte die gleiche WKA' mit $k = 1$, $p = 100\%$ erreicht werden wie mit $k = 2$, $p = 50\%$.

2. Da aber der Energieverbrauch von $k = 1$, $p = 100\%$ deutlich unter dem von $k = 2$, $p = 50\%$ liegt, siehe letzter Abschnitt, ermöglicht die Konfiguration $k = 1$, $p = 100\%$ die gleiche Sicherheit aus Energiesicht allerdings deutlich günstiger.

Algorithmus 17 sucht dementsprechend die *kostengünstigste* Lösung für k und p. Dabei ist klar, daß größere $k' > k$ und $p' > p$ ebenso die gewünschte Sicherheit garantieren – aber eben teurer sind.

Eingabe: $n, \delta, \beta, \mathrm{WKA}$
1: $k := 0, p := 0$
2: **repeat**
3: **if** $p < 100\%$ **then**
4: $p := (p + 1)$
5: **else**
6: $p := 0, k := (k + 1)$
7: **end if**
8: $\mathrm{WKA}' := \mathrm{ESAWN}(n, \delta, \beta, k, p)$ {s. Text}
9: **until** $\mathrm{WKA}' \geq \mathrm{WKA}$
Ausgabe: k, p

Algorithmus 17 Pseudocode zum Berechnen von WKA's

Partielle Ordnung $\succcurlyeq_{\mathrm{E}}$ damit total

Die auf Seite 152 aufgestellte partielle Ordnung $\succcurlyeq_{\mathrm{E}}$, die verschiedene ESAWN-Konfigurationen (k, p) in Bezug auf ihren Energieverbrauch ordnet, wird folglich total. Bei *fester* WKA gilt für alle Konfigurationen $(k, p), (k', p')$, die diese WKA erreichen:

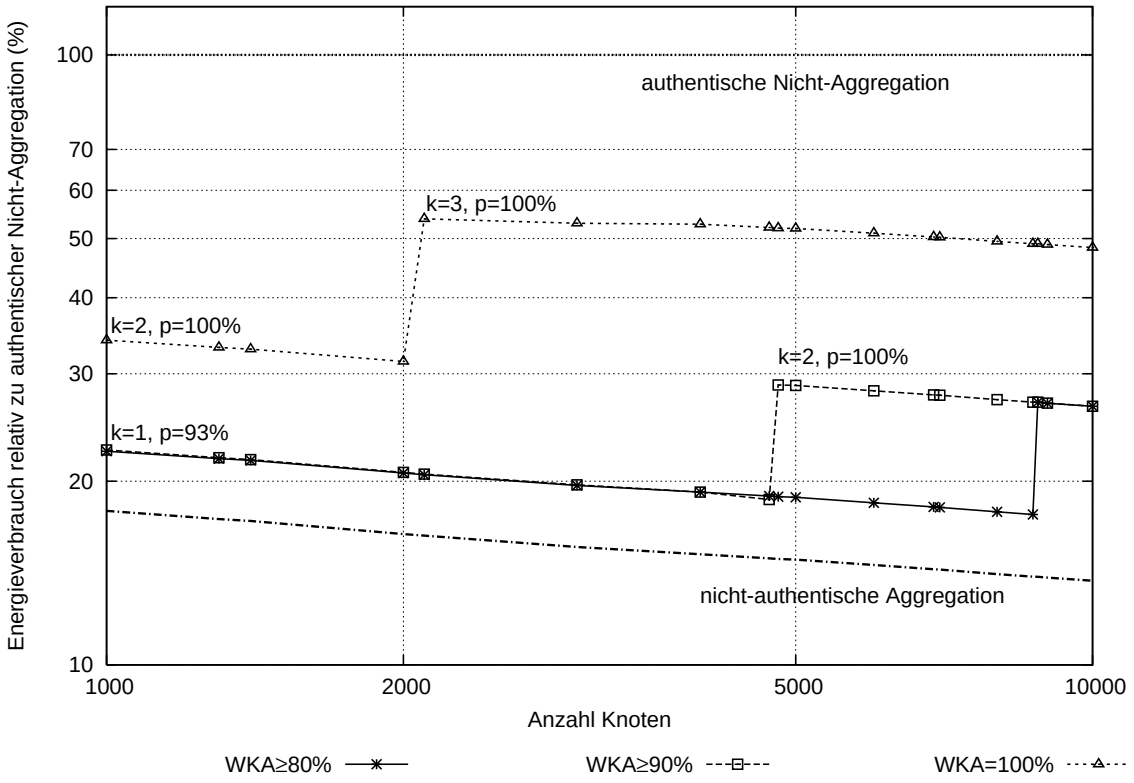
$$k \geq k' \Rightarrow \forall p, p' : (k, p) \succcurlyeq_{\mathrm{E}} (k', p').$$

Der Benutzer versucht demnach nur wie in Algorithmus 17, k so klein wie möglich zu halten.

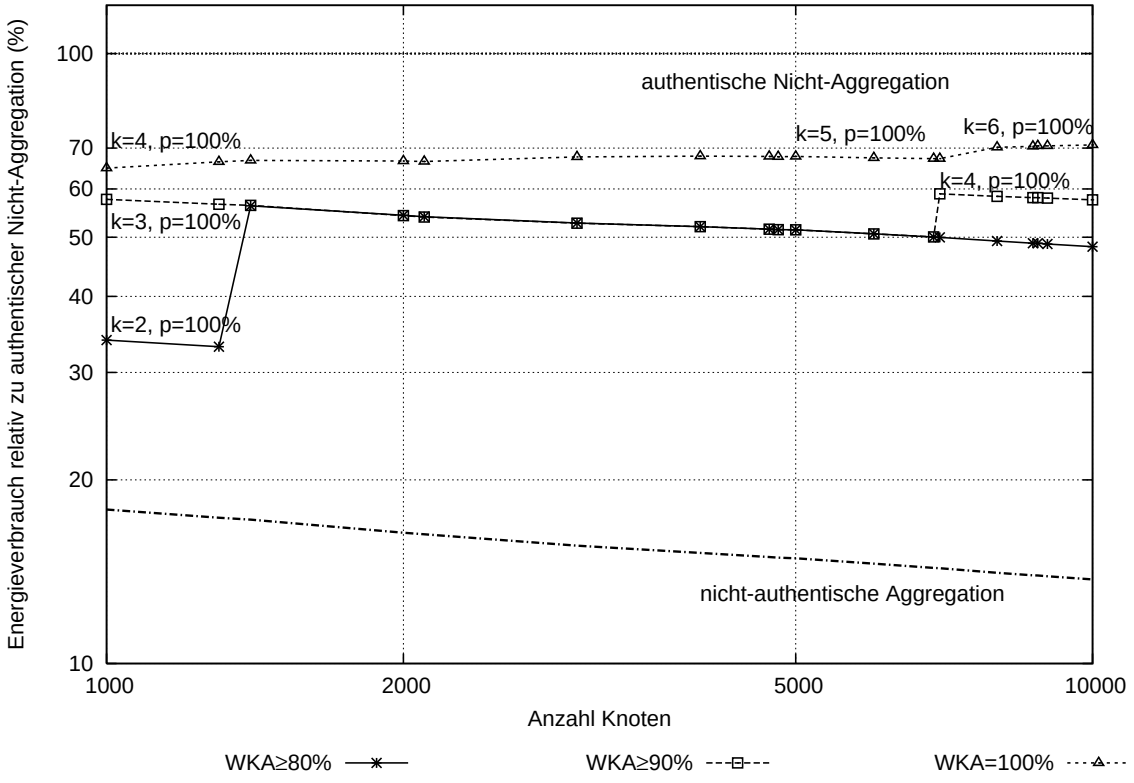
Die mit Algorithmus 17 gefundenen k und p erzielen im Durchschnitt *mindestens* die vom Benutzer gewünschte Sicherheit WKA. Die Betonung liegt hier auf dem Wort *mindestens*, da die sehr diskrete Wahl von k, beschränkt auf natürliche Zahlen, die WKA's nicht beliebig genau „treffen“ kann. Der Parameter k muß derart diskret gewählt werden, weil alles andere aus Protokollsicht keinen Sinn macht. Als Beispiel sei hier Abbildung 4.15(b) genannt, die zeigt, daß bei einem Netz mit mehr als $n \approx 1500$ Knoten und gewünschter WKA von 80% die Parameter $k = 2$, $p = 97\%$ nicht mehr ausreichen und $k = 3$, $p = 100\%$ notwendig werden. Dies ist die günstigste Wahl von p und k, um eine WKA von 80% zu erzielen – diese Konfiguration reicht jedoch dann sogar für eine WKA von 90% aus.

Zu erwähnen ist noch Zeile 8 von Algorithmus 17. Der Aufruf von ESAWN soll aussagen, daß in mehreren zufälligen, mit Hilfe der Parameter n, δ, β, k, p bestimmten Sensornetzen ESAWN simuliert und die sich ergebene durchschnittliche Wahrscheinlichkeit für eine korrekte Aggregation WKA' bestimmt wird.

Da sich das Energieverhalten bei unterschiedlichen durchschnittlichen Aggregationsgraden kaum verändert, ist in den Simulationen $\delta = 4$ festgehalten worden.



(a) $\beta = 1\%$ korrumpierte Knoten



(b) $\beta = 10\%$ korrumpierte Knoten

Abbildung 4.15 ESAWN Energieaufwand für Mindestwahrscheinlichkeiten korrekter Aggregation, $\beta = 1\%$ beziehungsweise $\beta = 10\%$ fest

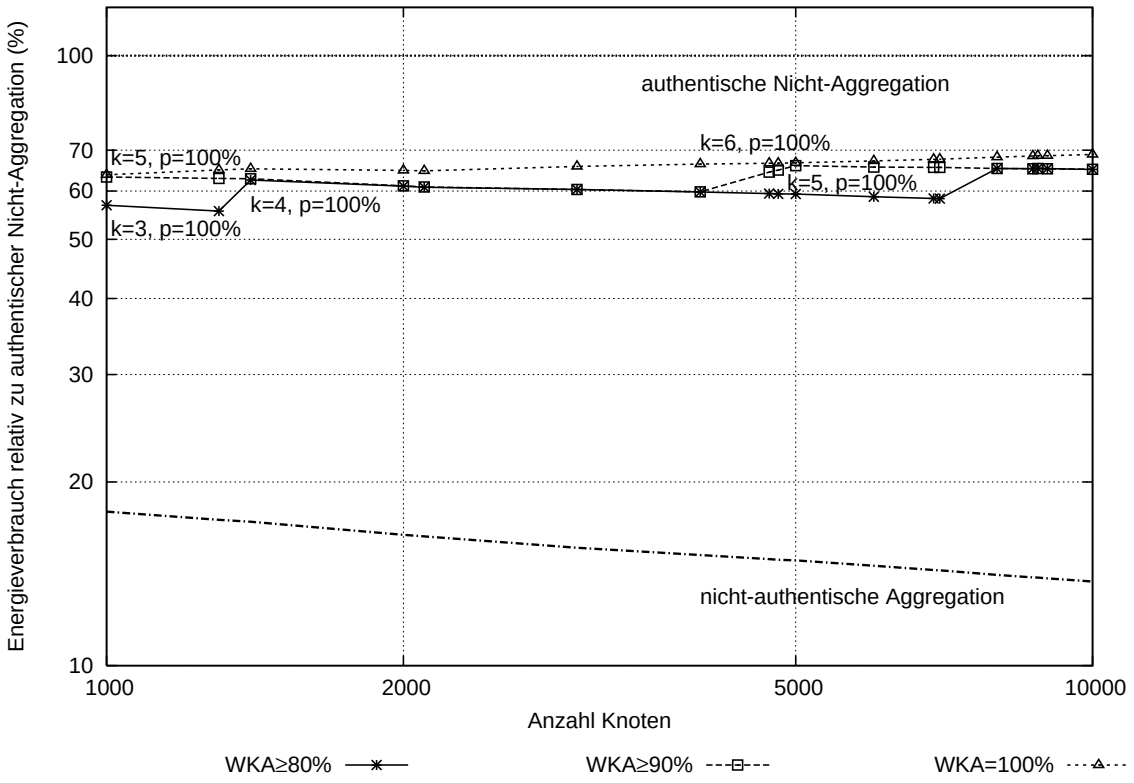


Abbildung 4.16 ESAWN Energieaufwand für Mindestwahrscheinlichkeiten korrekter Aggregation, $\beta = 20\%$ fest

Simulationsergebnisse

Zur Interpretation der Ergebnisse: Die Abbildungen 4.15 und 4.16 zeigen die anfallenden Energiekosten bei $n = 1000 \ldots 10000$ Sensorknoten. Abbildung 4.15(a) zeigt die Simulationsergebnisse bei angenommenem $\beta = 1\%$, Abbildung 4.15(b) bei $\beta = 10\%$ und Abbildung 4.16 schließlich bei $\beta = 20\%$ korrumpierten Knoten. Die einzelnen Kurven in den Abbildungen tragen ferner an einigen Stellen die Beschriftung der jeweils zugrundeliegende Konfiguration, das heißt die Wahl der Parameter k und p, die notwendig sind, um mindestens die gewünschte WKA zu erreichen. Der genaue Verlauf der einzelnen Kurven, eine komplette Übersicht über alle Simulationsergebnisse, findet sich in tabellarischer Form in Anhang D, S. 193.

Die Kurven in den Abbildungen „springen“, wenn durch die zunehmende Gesamtanzahl von Knoten im Sensornetz (X-Achse) auch die Gesamtanzahl korrumpierter Knoten zunimmt, das bisherige k dadurch nicht mehr ausreicht und ein größeres k notwendig wird. Da k nur diskret gewählt werden kann, springt der Energieverbrauch an dieser Stelle. Auch die Sprungstellen in den Abbildungen sind mit den ab dann geltenden Konfigurationen in den Abbildungen beschriftet. Als Beispiel sei wieder Abbildung 4.15(b) genannt, sowie die Kurve, welche den Energieverbrauch bei gewünschtem WKA von mindestens 80% zeigt. Bei $n \approx 1500$ Knoten springt die Kurve durch die diskrete Erhöhung von $k = 2$ auf $k = 3$ deutlich nach oben.

Die Gesamtzahl korrumpierter Knoten $\mathcal{B}$ und damit auch k muß in den Abbildungen deshalb zunehmen, weil β in den Simulationen festgehalten worden ist, n jedoch steigt.

ESAWN stehen demnach pro Verifikation einer Aggregation absolut gesehen mehr korrumpierte Knoten gegenüber.

Folglich steigt der Aufwand für ESAWN grundsätzlich mit beliebigen aber festen WKA und β bei steigender Gesamtknotenzahl n, da sich damit auch die Anzahl potentieller korrumpierter Knoten k erhöht. Um die gleiche Sicherheit bei steigenden Angreiferzahlen zu ermöglichen, muß ESAWN mehr Energie durch höhere k oder p ausgeben.

Zwischen zwei Sprungstellen, bei denen sich wie beschrieben k, p erhöht, sinkt der relative Energiebedarf allerdings leicht ab, siehe zum Beispiel Abbildung 4.15(b), Kurve für $\mathrm{WKA} \geq 80\%$ zwischen $n \approx 1500$ und $n \approx 6000$ Knoten. Da k und p nur diskret gewählt werden können, reicht die Wahl von k, p bei $n \approx 1500$ für das gesamte Intervall zwischen 1500 und 6000 Knoten aus, um eine $\mathrm{WKA} \geq 80\%$ zu erzielen. Da in diesem Intervall bei festem k, p der Energieverbrauch ESAWN konstant mit steigendem n bleibt, der Energieverbrauch von *authentisch Nicht-Aggregation* allerdings steigt, fällt damit der *relative* Energieverbrauch von ESAWN.

Die Steigung an der Sprungstellen müßte laut Theorie deutlich höher sein als in den Abbildungen: Bei einer Gesamtknotenzahl n' ist die Konfiguration mit einem k' ausreichend, wohingegen bei $n'+1$ dann $k'+1$ notwendig wird. An der Stelle $n', n'+1$ müßte dementsprechend eine steile Sprungstelle sein. Da in den Simulationen jedoch auf Grund hoher Simulationslaufzeiten die Stelle $n', n'+1$ nur approximativ mittels Algorithmus 17 bestimmt werden konnte, sind in den Abbildungen die Sprungstellen nicht ganz vertikal. Die approximative Annäherung an Sprungstellen ist auch der Grund für die unregelmäßige Wahl der simulierten Stützstellen.

Einen interessanten Kurvenverlauf beschreiben die Kurven $k = 4, p = 100\%$ in Abbildung 4.15(b) und $k = 5, p = 100\%$ in 4.16. Wie man erkennt, steigt der Energieverbrauch kontinuierlich mit zunehmender Knotenzahl n. Dies liegt darin begründet, daß ESAWN, insbesondere bei kleinem n, sehr häufig für Aggregationen nahe der Wurzel des Aggregationsbaumes nicht $k = 4$ oder $k = 5$ Zeugen findet, sondern nur $k' < k$. ESAWN wählt dann für diese Aggregationen auch nur k' Zeugen aus, womit die jeweilige Überprüfung deutlich günstiger wird. Mit zunehmendem n stehen auch im oberen Teil des Baumes relativ mehr Zeugen zur Verfügung, die ESAWN auswählt – ESAWN verteuert sich mehr und mehr, trotz gleichbleibendem k und p.

Wie man sieht, erreicht ESAWN bei angenommenen $\beta = 1\%$ korrumpierten Knoten und gewünschter 100% Sicherheit noch Energieeinsparungen von bis zu 50% gegenüber nicht aggregierendem Datentransport. Falls hingegen eine probabilistisch relaxierte WKA von 80% Prozent ausreicht, kann bei großen Sensornetzen von $n = 10000$ Knoten über 70% Energie eingespart werden. In kleineren Netzen von nur 1000 Knoten sind sogar Einsparungen von über 75% möglich. Bei stärkerem Angreifer, zum Beispiel angenommenen $\beta = 20\%$, erhöht sich der Energieaufwand für ESAWN deutlich. Doch auch hier kann, selbst bei $\mathrm{WKA} = 100\%$ und 10000 Knoten noch mehr als 30% Energie gegenüber nicht-aggregierendem, sicheren Datentransport eingespart werden.

Mit diesen Ergebnissen ist der Benutzer in der Lage, für eine gewünschte WKA sowie geschätzte n, δ und β den erwarteten Energieverbrauch zu bestimmen und letztlich zu entscheiden, ob ihm die Sicherheit entsprechend viel Energie wert ist. Umgekehrt kann der Benutzer wenn er nicht mehr als eine gewisse Obergrenze an Energie zur Verfügung hat, bestimmen, welche WKA sich dafür erzielen läßt.

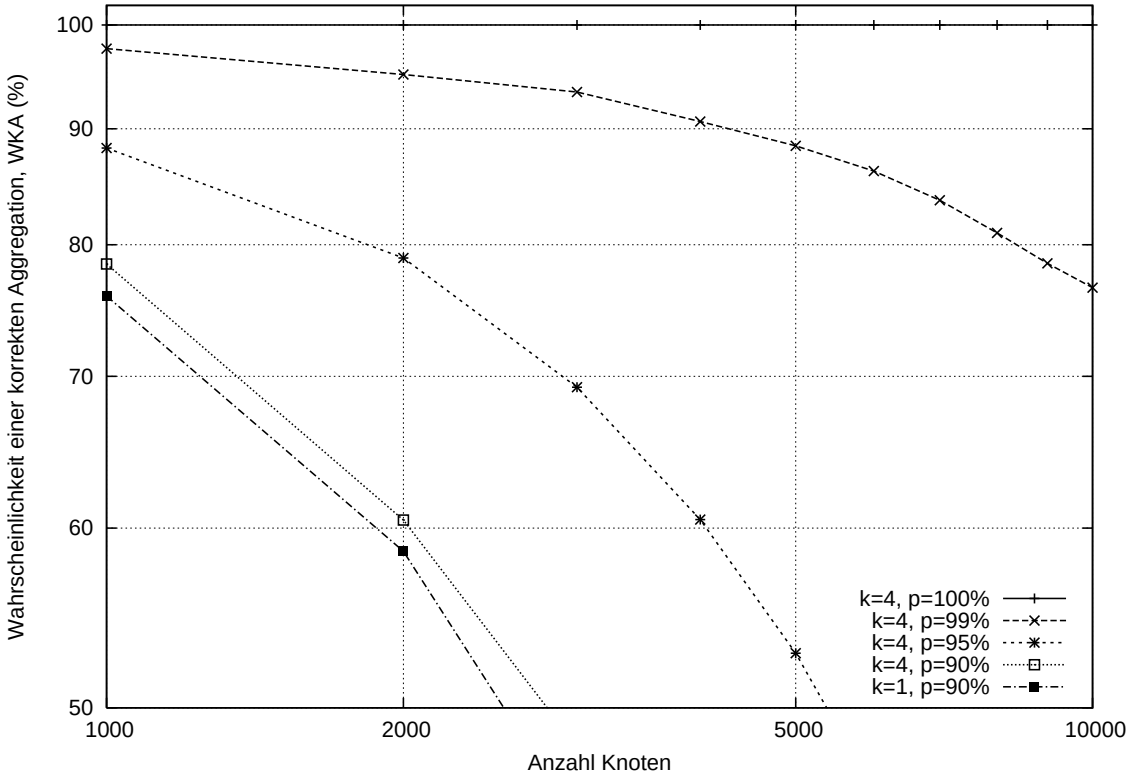


Abbildung 4.17 Verlauf der WKA in Abhängigkeit von p, $\beta = 1\%$

An dieser Stelle sei schließlich noch auf Anhang D, S. 193 verwiesen, der in 3 Tabellen die Simulationsergebnisse, deutlich umfangreicher als das in den Grafiken 4.15 und 4.16 der Fall ist, darstellt. Der Anhang zeigt nicht nur für Knoten $n = 100 \ldots 10000$ jeweils k und p, sondern auch für WKA's zwischen 50% und 100%, um die Ergebnisse weiter zu verdeutlichen.

4.4.3.3 Zum Verlauf der WKA-Kurven

Der Verlauf der Kurven aus den Abbildungen 4.15 und 4.16 mag auf den ersten Blick verwundern: Der Wert für p liegt hier immer bei 100%, selbst bei geforderten WKA's von nur 80%. Der Energieaufwand in den Abbildungen steigt ausschließlich in Sprüngen durch das Inkrementieren von k und nicht kontinuierlich durch Erhöhen von p.

Geringere Werte für p sind allerdings nicht möglich, damit erreicht ESAWN die gewünschte WKA ganz einfach nicht, wie Abbildung 4.17 zeigt. Die Abbildung skizziert die sich ergebende WKA bei verschiedenen p und k, allerdings festgehaltenem $b = 1\%$. Wie man sieht, fällt die WKA bei $p < 100\%$ mit zunehmender Anzahl Knoten relativ schnell ab. Selbst die $k = 4, p = 99\%$ Kurve erreicht eine WKA $\geq 90\%$ nur bei Knotenzahlen $n < 5000$. Eine höhere WKA, sogar bei $n \geq 5000$ Knoten, läßt sich, wie in Abbildung 4.15(a) zu sehen, allerdings schon durch die energetisch sehr viel günstigere Konfiguration $k = 2, p = 100\%$ erzielen.

Abbildung 4.17 zeigt damit: Die Wahl von p hat einen enormen Einfluß auf die WKA. Um *hohe* WKA's zu erzielen, *muß* p auf 100% gesetzt, somit jede Aggregation überprüft werden. Verglichen damit erscheint der Einfluß von k auf die WKA gering. Man betrachte den marginalen Unterschied der Sicherheit zwischen den Kurven $k = 1, p = 90\%$ und der

wesentlich teureren $k = 4, p = 90\%$. Dies veranschaulicht, warum in den Abbildungen 4.15 und 4.16 p immer den Wert 100% hat: p zu erhöhen, kostet weniger als k, bringt jedoch deutlich mehr Sicherheit.

Warum die WKA beim Erhöhen von p stärker steigt als beim Erhöhen von k erklärt sich wie folgt. Sobald ein gewisses k erreicht ist, mit dem in einem Netz mit n Knoten und β Prozent korrumpierten Knoten durchschnittlich auf jedem Aggregationspfad ausreichend Zeugen zur Verfügung stehen, bringt das weitere Erhöhen von k nur noch geringe Sicherheit. Dadurch werden lediglich die unwahrscheinlichen Fälle abgedeckt, in denen der zufällige Aggregationsbaum über eine höhere als die im Durchschnitt zu erwartende Baumhöhe h verfügt. Das Erhöhen von p bringt allerdings *immer* mehr Sicherheit, das heißt eine höhere WKA. Es lohnt sich demnach aus Energie- *und* aus Sicherheitssicht, p vor k zu erhöhen.

4.4.4 Wahl von k bei ESAWN und SKEY

Sowohl SKEY als auch ESAWN verwenden beide einen Parameter k, mit dem sie gegen eine Anzahl korrumpierter Knoten, die sich am Schlüsselaustausch oder der Aggregation beteiligen, justiert werden. Wie man sieht, benötigt dabei ESAWN verglichen mit SKEY bei gleicher Knotenzahl und gleicher Anzahl korrumpierter Knoten $\mathcal{B}$ wesentlich höhere k's. Das liegt daran, daß die Wahrscheinlichkeit, daß bei SKEY die initialen Zufallsknoten derart in ein und dem selben Teilbaum liegen und die gleichen $k' > k$ korrumpierten Vorgänger ihre Schlüsselteile weiterleiten, eher gering ist. Damit reicht für SKEY die Wahl $k \geq \beta \cdot (h-1)$ aus. Hingegen ist es bei ESAWN doch wahrscheinlich, gerade bei großem $\mathcal{B}$, daß auf mindestens einem Aggregationspfad irgendwo im Baum doch $k' > k$ korrumpierte Knoten hintereinanderliegen. Damit muß für ESAWN das größere $k \geq (h-1)$ gewählt werden.

4.4.5 Kombinierter Aufwand von SKEY und ESAWN

Die beiden Protokolle SKEY und ESAWN arbeiten unabhängig voneinander. ESAWN benötigt als Voraussetzung lediglich paarweise symmetrische Schlüssel zwischen Knoten, die auf dem selben Aggregationspfad liegen.

Der Aufwand dieser beiden Protokolle in Bezug auf Speicher- und Energieverbrauch ist dementsprechend auch unabhängig voneinander und kann einfach addiert werden. Der Speicheraufwand von SKEY und ESAWN zusammen skaliert mit $O(\log n)$ pro Knoten (SKEY $O(\log n)$, ESAWN $O(1)$), der Energieaufwand ebenso mit $O(\log n)$ pro Knoten (SKEY $O(\log n)$, ESAWN $O(\log n)$).

4.4.6 Ergebnisse im Überblick

ESAWN erfüllt die vorab geforderten Entwurfsziele:

- *Funktionalität*
  ESAWN sichert Authentizität zwischen allen Knoten im Netz trotz Aggregation zu.

  1. Jeder Empfänger x eines Aggregates kann sicher sein, daß dieses Aggregat korrekt aggregiert worden ist und die Knoten, welche Daten zur Bildung dieses Aggregats beigetragen haben, diejenigen sind, die dafür auch laut Aggregationsbaum zuständig sein müssen.
     Die Überprüfung der Authentizität ist x dabei möglich, bevor er aus den empfangenen Daten selbst ein Aggregat bilden und versenden muß.

2. ESAWN unterstützt dabei kaskadierende Aggregationen sowie beliebige, mathematisch berechenbare Aggregationsfunktionen.

- *Sicherheit*
  ESAWN ist dabei sicher gegen das in dieser Arbeit angenommenen Angreifermodell mit korrumpierten Knoten. Der Benutzer des Sensornetzwerks kann durch Schätzen der zu erwartenden Gesamtgröße des Sensornetzes n sowie den voraussichtlichen Fähigkeiten des Gegner ($\mathcal{B}$) ESAWN so parametrisieren, daß ESAWN gegen diesen Angreifer authentisch Daten transportiert.

  Falls sich n und die Gesamtanzahl korrumpierter Knoten $\mathcal{B}$ nicht abschätzen lassen, kann der Benutzer ESAWN gegen eine bestimmte Mindestanzahl korrumpierter Knoten $\mathcal{B}$ parametrisieren, gegen die ESAWN noch sicher funktioniert. Der Benutzer kann auf diese Weise einen Kompromiß zwischen Authentizität gegen möglicherweise viele korrumpierten Knoten und dem daraus resultierenden Energieverbrauch finden.

- *Dezentralität*
  ESAWN benötigt für seine Arbeit keinerlei feste Infrastrukturen, Server, einen Notar oder ähnliches. Auch der Benutzer muß, während das Sensornetz seinen normalen Dienst erbringt, keinerlei besondere Interaktionen mit einzelnen Knoten durchführen.

  Auch die Senke stellt bei ESAWN keine Infrastrukturkomponente dar. Sie überprüft – genau wie jeder andere Knoten im Netz – lediglich die Aggregationen ihrer Nachfolgerknoten auf den k Höhen unterhalb im Aggregationsbaum. Falls die Senke als Zeuge ausfallen würde oder nicht erreichbar wäre, hätte dies die selben Konsequenzen wie der Ausfall jedes beliebigen anderen Zeugen einer Aggregation.

- *Effizienz*
  Die Effizienz von ESAWN, sein Energieverbrauch, läßt sich vom Benutzer über den Sicherheit–Energie Tradeoff parametrisieren. Als Ergebnis der Simulationen kann festgehalten werden, daß selbst bei hohen Sicherheitsanforderungen von 80, 90 oder gar 100% bezüglich der Authentizität in Gegenwart von bis zu 20% korrumpierten Knoten im Netz immer noch deutliche Energieeinsparungen gegenüber authentischem Datentransport ohne Aggregation erreichbar sind.

  Grundsätzlich skaliert sowohl der Energieverbrauch von ESAWN als auch der Speicherverbrauch pro Knoten mit $O(1)$ bei steigender Gesamtknotenzahl n.

4.5 Zusammenfassung

Neben der Geheimhaltung von transportierten Daten gegenüber einem Angreifer stellt vor allem ihre Authentizität eine besondere Herausforderung im Sensornetz dar. Während Geheimhaltung und Authentizität zwischen zwei Knoten alleine durch den Besitz gemeinsamer Schlüssel zugesichert werden können, gilt die Zusicherung von Authentizität als die weitaus schwierigere Aufgabe.

Dieses Kapitel hat zunächst die besonderen Auswirkungen und Anforderungen von Sensornetzen in Bezug auf authentischen Datentransport untersucht und daraus eine Reihe von Entwurfszielen hergeleitet, die ein geeignetes, effizientes Protokoll zum authentischen Datentransport erfüllen muß. Dabei ist vor allem zu bedenken, daß die Authentizität von Daten im Widerspruch zu der in dieser Arbeit angenommen, sehr allgemeinen

Form kaskadierender Aggregation steht. Ein geeignetes Protokoll zum authentischen Datentransport muß dieses Dilemma beachten.

Eine Übersicht über den aktuellen Stand der Forschung hat gezeigt, daß die bisher veröffentlichten Lösungen den aufgestellten Entwurfszielen nicht genügen: Sie schränken häufig die möglichen Aggregationsfunktionen ein und sind unsicher gegenüber Angreifern, welche Sensorknoten im Netz korrumpieren können.

Im Rahmen dieser Arbeit ist deshalb ESAWN, ein Protokoll zum authentischen Datentransport in Sensornetzen, entworfen worden. Die Grundidee von ESAWN basiert auf einem *Sicherheit–Energie-Tradeoff*, um das Dilemma des Widerspruchs zwischen Authentizität und Aggregation zu lösen. ESAWN realisiert diesen Tradeoff durch *probabilistisches Relaxieren* der Authentizität: Empfangene Aggregate sind nur mit einer bestimmten Wahrscheinlichkeit authentisch. Der „Benutzer“ eines Sensornetzes kann *graduell* abwägen, wie sicher oder wie energieeffizient Daten transportiert werden. Je *sicherer* Daten gegen um so *mehr* korrumpierte Knoten transportiert werden sollen, desto *energieaufwendiger* wird der Transport. Je *günstiger* Daten transportiert werden sollen, desto *unsicherer* wird der Transport gegen *weniger* korrumpierte Knoten.

In Gegenwart von korrumpierten Knoten im Netz hebt ESAWN die Aggregation auf einigen Ebenen im Netz auf – dadurch läßt sich dann die Authentizität der Daten verifizieren. Da ESAWN die Aggregation nur teilweise aufhebt, können dennoch Daten aggregiert und energieeffizient durchs Netz transportiert werden. ESAWNs Energieverbrauch bleibt invariant von der Gesamtzahl der Knoten n im Netz, er skaliert mit $O(1)$ pro Knoten.

Zusammen mit der Schlüsselverteilung SKEY ermöglicht ESAWN damit einen sicheren aber dennoch effizienten, aggregierenden Datentransport in drahtlosen Sensornetzen.

5. Zusammenfassung und Ausblick

Durch die zunehmende Miniaturisierung von Computersystemen dringen diese immer mehr in alltägliche Lebensbereiche des Menschen ein. Sensornetze, drahtlose Netzwerke tausender Kleinstcomputer, unterstützen den Menschen in vielen Situationen, in denen Ereignisse oder Phänomene beobachtet, gemessen und verarbeitet werden sollen. Sensornetze charakterisieren sich dabei dadurch, daß sie vom Benutzer weitestgehend allein und selbstorganisierend arbeiten und auf zentrale Infrastrukturkomponenten wie Server verzichten müssen. Aufgrund ihrer extrem beschränkten Ressourcen, rechenschwacher CPUs, wenig Speicher und einer Batterie als einzige Energieversorgung sind sie auf besonders effiziente Protokolle angewiesen. Eine weitere Besonderheit betrifft die Form der drahtlosen Kommunikation in Sensornetzen: Aggregation. Die im Sensornetz gemessenen Daten werden auf ihrem Weg zu einer Datensenke bereits mehrfach vorverarbeitet.

Vor diesem Hintergrund erscheint der Wunsch nach Sicherheit für die im Sensornetz transportieren Daten als eine neue Herausforderung. Häufig mißt und transportiert das Sensornetz sehr sensible Daten, beispielsweise medizinische Daten eines Patienten. Diese müssen vor Abhören oder Modifikation durch einen Angreifer geschützt werden. Dieses Problem, dem sich die vorliegende Arbeit gewidmet hat, läßt sich in zwei Teile aufteilen. Erstens: Wie lassen sich in einem solchen Sensornetz kryptographische Schlüssel austauschen? Schlüssel sind zwingend notwendige Voraussetzung für jede weitere Kommunikation zwischen Sensorknoten. Zweitens: Wie läßt sich im Anschluß an den Schlüsselaustausch sicherstellen, daß die Daten im Netz authentisch transportiert werden? Authentizität bedeutet, daß der Empfänger eines Datums überprüfen kann, ob dieses Datum tatsächlich so von einem angenommenen Sender verschickt worden ist.

Dabei handelt es sich jedoch um keine einfachen Aufgaben. Die aufgezählten Eigenschaften von Sensornetzen erweisen sich gerade im Kontext von Sicherheit als äußerst ungünstig: Die beschränkten Ressourcen und die häufig endliche Energieversorgung der Sensoren erfordern einen besonders bewußten Umgang mit energieaufwendiger Funkkommunikation und kryptographischen Primitiven. Die Selbstorganisation des Netzes erschwert

den Schlüsselaustausch, die aggregierende Form des Datentransports erschwert die Authentizitätsüberprüfungen von Daten. Weiterhin kommt hinzu, daß der Angreifer im Gegensatz zu klassischen Netzen über die Möglichkeit verfügt, einen Teil der Sensorknoten zu korrumpieren – er bringt diese Knoten in seine Gewalt und steuert sie fern.

Bisherige Lösungansätze behandeln die Thematik nur unzureichend. Schlüsselaustauschprotokolle behandeln die aggregierende Form des Datentransports nicht und kennzeichnen sich besonders durch hohen Ressourcenverbrauch und ihre Anfälligkeit gegenüber korrumpierten Knoten. Die bisherigen Protokolle zum authentischen Transport der Daten schränken meist die Form der Aggregation im Netz stark ein, so daß sie sich nur für sehr spezielle Szenarien eignen.

5.1 Ergebnisse der Arbeit

Im Rahmen dieser Arbeit sind zwei Protokolle präsentiert worden: SKEY und ESAWN.

Betrachtet man aggregierenden Datentransport, so stellt man fest, daß hier Kommunikation nur zwischen Sensorknoten stattfindet, die zu einander in sogenannter Aggregationsbeziehung stehen. Demzufolge müssen Schlüssel auch nur zwischen Knoten ausgetauscht werden, die zueinander in Aggregationsbeziehung stehen. Um weiterhin Authentizität von erstellten Aggregaten überprüfbar zu machen, sind insgesamt Schlüssel notwendig zwischen allen Knoten und denjenigen auf ihrem sogenannten Aggregationspfad. Dies leistet das Protokoll SKEY. Der dafür benötigte Energiebedarf sowie der Speicherbedarf steigen pro Knoten nur logarithmisch mit der Gesamtanzahl aller Knoten im Netz. Dies ist insbesondere in Relation zu den häufig zitierten Schlüssellisten-basierten Protokollen bemerkenswert: Je nach Konfiguration des Netzes und erwartetem Anteil korrumpierter Knoten erreicht SKEY bis zu Faktor 600 weniger Speicherverbrauch und Faktor 4 weniger Energieverbrauch. SKEY läßt sich weiterhin auf den möglichen Anteil korrumpierter Knoten im Netz parametrisieren. Dies führt zu einem ersten *Sicherheit-Energie-Kompromiß*: Je größer die Anzahl korrumpierter Knoten, je höher also die Sicherheitsanforderungen, desto höher steigt auch der Energieverbrauch von SKEY.

Einen *Sicherheit-Energie-Kompromiß* präsentiert auch ESAWN. Zunächst ist festzuhalten, daß sich bei beliebiger Aggregation die Authentizität von entstehenden Aggregaten nicht überprüfen läßt. Um dennoch Authentizität überprüfbar zu machen, schlägt ESAWN vor, Aggregation teilweise aufzuheben. Damit lassen sich dann Aggregate in Gegenwart eines Anteils korrumpierter Knoten überprüfen. Zur Kompensierung des daraus entstehenden Mehraufwands an Energie erlaubt ESAWN probabilistische Überprüfungen von Authentizität. Dies mündet in einer *probabilistisch relaxierten* Authentizität in Gegenwart eines bestimmten Anteils korrumpierter Knoten auf der einen sowie dem daraus resultierenden Energieaufwand auf der anderen Seite. Je nach Konfiguration des Netzes, Anteil korrumpierter Knoten und probabilistischer Sicherheit erreicht ESAWN bis zu 80% weniger Energieverbrauch als authentischer Datentransport ohne Aggregation

Beide Protokolle sind dabei in Bezug auf ihre Leistungsfähigkeit nicht nur theoretisch untersucht worden, sondern auch im Rahmen einer Implementierung für eine Simulationsumgebung. Die Ergebnisse der Simulationen haben dabei die theoretisch getroffenen Aussagen gestützt.

5.2 Weiterführende Arbeiten

Die Idee, mit Hilfe eines Sicherheit-Energie-Kompromisses auf Kosten niedrigeren Energieverbrauchs Sicherheit graduell abzustufen, erscheint gerade für drahtlose Sensornetze mit ihren endlichen Batteriereserven geeignet. Solche Kompromisse ließen sich sicherlich auf vielen Ebenen dieser Systeme finden.

Kompromisse auf anderen Ebenen des Systems

So wurden beispielsweise kryptographische Verschlüsselungsmechanismen in dieser Arbeit als per se sichere Black-Box verwendet. Auch hier könnte man aber die Sicherheit der Chiffre gegen ihren Energieverbrauch abwägen und parametrisieren. Eine erste Idee dazu schlägt zum Beispiel *Battery Power-Aware Encryption* aus Chandramouli et al. [60] vor.

Es ist auch vorstellbar, daß Sensoren nicht einzeln allein ihre Daten verschlüsseln, Authentizität prüfen usw. Bei wenig vorhandener Energie könnte sich stattdessen eine Gruppe von Sensoren zusammenschließen und gemeinsam an der Durchführung einer Operation für Sicherheit arbeiten oder abwechselnd. Hier ließen sich sicherlich noch viele weitere Kompromisse finden.

Robustheit

Das deutsche Homonym Sicherheit steht für die englischen Begriffe *Security* und *Safety*. Diese Arbeit hat sich ausschließlich mit Security-Aspekten beschäftigt, also mit der wichtigen Frage nach dem Schutz der Daten im Netz vor Angreifern.

Der zweite Aspekt, Safety, im deutschen Sprachgebrauch häufig mit Robustheit oder Zuverlässigkeit übersetzt, spielt aber gerade in Sensornetzen eine ebenso wichtige Rolle. Die einfachen Funkschnittstellen von Sensoren verfügen nur über eine geringe Sendeleistung und sind besonders anfällig gegen elektromagnetische Störungen, Reflexionen usw. Neben einem Angreifer, der beispielsweise die Integrität von Daten bedroht, existieren demnach verwandte Bedrohungen durch normale Naturphänomene. Wie kann der Datentransport gegen Bitfehler während der Übertragung „abgesichert“ werden? Wie kann in Sensornetzen mit hohen Nachrichtenverlustraten umgegangen werden?

Auch hier eigenen sich klassische Lösungsansätze, beispielsweise sogenannte ARQ-Verfahren wie „Stop-and-Wait“ oder „Go-Back-N“ nur sehr eingeschränkt: Ihr Prinzip basiert auf fortlaufenden Wiederholungen von Funkübertragungen. Gerade Funkübertragungen sind aber die in Sensornetzen energetisch mit Abstand teuerste Operation. Robuster und zuverlässiger Datentransport stellt sich folglich als weitere Herausforderung für Sensornetze dar.

Möglicherweise läßt sich zu dieser Problematik wieder ein Kompromiß finden, zum Beispiel ein Robustheit-Energie-Kompromiß beziehungsweise auch ein Genauigkeit-Energie-Kompromiß, den Arbeiten wie Blaß et al. [30] vorschlagen. Die gesamte Thematik des zuverlässigen Datentransport ist Gegenstand aktueller Forschungsarbeiten im BW-FIT Projekt ZeuS [223].

Grundsätzlich sind sowohl Security als auch Safety aufeinander angewiesen: Es kann keinen sicheren Datentransport gegenüber einem Angreifer geben, wenn Nachrichten durch elektromagnetische Störungen falsch empfangen werden. Genauso hilft robuster Datentransport nicht, wenn ein Angreifer gezielt gefälschte Nachrichten erzeugt.

Andere Systemmodelle

In dieser Arbeit sind starke Annahmen zum Beispiel bezüglich des Angreifers getroffen worden. In der Realität mag es durchaus auch schwächere Angreifer geben, die nicht zwangsläufig über all die in dieser Arbeit angenommenen Eigenschaften verfügen. Hier stellt sich dann die Frage, inwieweit sich Protokolle in Abhängigkeit der Fähigkeiten des Angreifers kostengünstiger als bisher verhalten. Auch die Annahmen über das Sensornetz selbst sind variierbar. Welche Auswirkungen hat die Anordnung der Sensoren speziell in Gitterform, beispielsweise in einen Teppich eingewebte Sensoren? Oder in einem Kreis oder einer langen Kette? Der Energieverbrauch und die Sicherheit der Protokolle ließe sich in Abhängigkeit der zugrundeliegenden Systemmodelle anpassen.

A. Zum Energieverbrauch der MICA2-Knoten

In der Literatur werden zum Strom- beziehungsweise Energieverbrauch eines MICA2-Sensorknoten häufig unterschiedliche, widersprüchliche Angaben gemacht.

Der Energieverbrauch eines Knotens setzt sich im allgemeinen aus dem Verbrauch des Mikrocontrollers, der Funkschnittstelle sowie des sogenannten Boards, auf dem die eigentliche Sensorik integriert ist, zusammen.

Prozessor

Für den Mikrocontroller, den ATMega128, finden sich in der Literatur folgende unterschiedliche Angaben: Crossbow gibt *maximal* 8 mA [67] an, allerdings ohne die variabel einstellbare Taktfrequenz und ebenso variierbare angelegte Spannung zu benennen. Andere Publikationen sprechen von 5 mA Energieverbrauch ohne Angabe von Taktfrequenz und Spannung [151]. Piotrowski et al. [178] nennen 10 mA bei 3V. Atmel selbst gibt auf dem Datenblatt des Mikrocontrollers zwischen 5 mA und 6 mA Energieverbrauch bei 2,7 V Spannung und 8 MHz Taktfrequenz an. Eigene Messungen haben ebenso ≈ 5 mA Energieverbrauch bei 2,7 V Spannung und 8 MHz Taktfrequenz ergeben. Dabei entsprechen 8 MHz Prozessortakt einem angelegten externen Takt von $\approx 7,3$ MHz.

Funkschnittstelle

Hersteller Crossbow selbst gibt für die Funkschnittstelle unterschiedliche Energieverbräuche an, zum Beispiel 16 mA [69] oder 25 mA [67] zum Versenden. Die Funkschnittstelle sendet dabei mit einem Datendurchsatz von 38400 Bit/s auf 433 MHz.

Grundsätzliche Ungenauigkeiten

Zum Versenden einer TinyOS-Nachricht muß nicht nur die Funkschnittstelle, sondern für die Dauer der Übertragung der Nachricht auch der Mikrocontroller aktiv sein. Die Energieverbräuche von Funkschnittstelle und Mikrocontroller addieren sich hier also. Das

hieße beispielsweise bei 16 mA für den Funk und 5 mA für den Mikrocontroller insgesamt $16 + 5 = 21$ mA Energieverbrauch während der Übertragung einer Nachricht. Bei Annahme der oben genannten anderen Werte 8 mA und 25 mA ergeben sich insgesamt $8 + 25 = 33$ mA Energieverbrauch. Andere Arbeiten kommen auf völlig andere Energieverbräuche zum Versenden von Nachrichten (Funk+Mikrocontroller), beispielsweise von insgesamt 25 mA [236] oder auch nur 12 mA [98].

Dabei wird jeweils immer die Sensorik auf dem Board des Sensorknotens nicht mitbetrachtet, die auf jeden Fall auch Energie kosten muß. In Welsh et al. [240] mißt man angeblich einen *Gesamtenergieverbrauch* des kompletten MICA2-Sensorknotens von 15–20 mA.

Weiterhin vernachlässigen oder vereinfachen alle Arbeiten, daß Energieverbräuche über die Zeit hinweg nicht konstant bleiben: Beim Anschalten und kurzen Aufwärmen der Funkschnittstelle entstehen, wenn auch in einem geringen Zeitintervall, Peaks, die wesentlich über die oben angegebenen Verbräuche hinausgehen. Etwas ähnliches läßt sich auch beim Einschalten oder Aufwachen des Mikrocontrollers feststellen.

Fazit

Die doch stark unterschiedlichen Angaben zum Energieverbrauch der MICA2-Plattform in der Literatur lassen darauf schließen, daß exakte Messungen sich als eine große Herausforderung und alles andere als einfach darstellen. Zu vielfältig sind die Konfigurationsmöglichkeiten der Hardware in Form von (externen) Taktfrequenzen, angelegten Spannungen und zum Einsatz kommender Sensor-Boards. Zur Zeit existiert noch kein allgemeines und präzises Energiemodell für die MICA2-Plattform.

Für die Simulationen während der Evaluierung von SKEY und ESAWN wird in dieser Arbeit durchgängig *ein festes* Modell des Energieverbrauches (und aller anderer wichtiger Systemparameter) angenommen. Dies findet sich sowohl im Grundlagenkapitel auf Seite 13 sowie ausführlicher in der Einleitung zu den Simulationen von SKEY auf Seite 94. Es basiert auf der ersten obigen Annahme von Crossbow [11, 69], daß der Atmel Mikrocontroller 5 mA und das Versenden insgesamt 21 mA Energie verbrauchen. Das Verhältnis dieser beiden Zahlen zueinander ist dabei ≈ 4 – genau wie in der zweiten Annahme von Crossbow [67], daß der Mikrocontroller 8 mA und die Funkschnittstelle 33 mA Energie benötigen. Genauere Untersuchungen und Messungen zum Energieverbrauch der MICA2-Knoten sind nicht Teil dieser Arbeit.

B. Analyse zufallsverteilter Schlüssellisten

Im folgenden werden Verfahren analysiert, die auf der zufälligen Verteilung von Schlüssellisten auf Knoten basieren. Diese Arbeiten werden in der Literatur sehr häufig zitiert. Aufgrund ihrer Popularität sind sie in dieser Arbeit genauer untersucht worden. Es hat sich dabei gezeigt, daß sie im Gegensatz zur landläufigen Meinung bestimmte Sicherheits- und Effizienzeigenschaften nicht *erfüllen*.

B.1 Funktionsweise

Das Grundprinzip dieser sogenannten *Random Key Pre-Distribution* Protokolle, siehe unter anderem Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [88, 89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251], geht zurück auf die ursprüngliche Idee von Eschenauer und Gligor in [88]. Zunächst berechnet der Benutzer eine sehr große, durchnumerierte Liste symmetrischer Schlüssel, den Key-*Pool* P, vorab offline. Auf jedem Knoten a wird vor seinem Deployment eine zufällige Teilmenge, dieser Schlüssel zusammen mit den jeweiligen Nummern der Schlüssel im sogenannten Key-*Ring* R_a vorab gespeichert.

Nun ist die Wahrscheinlichkeit relativ groß, daß selbst bei einem sehr großen Key-Pool zwei Sensoren a und b jeweils mindestens einen Schlüssel auf ihren Key-Ringen R_a und R_b gemeinsam haben. Das bedeutet: $R_a \cap R_b \neq \varnothing$. Wird demnach ein neuer Knoten a dem Netz hinzugefügt, so schickt a per Broadcast zu all seinen Nachbarn in direkter Funkreichweite eine Challenge α im Klartext zusammen mit einer Chiffratliste $C_{K_i} = E_{K_i}(\alpha)$, wobei K_i für die einzelnen Schlüssel seines Key-Rings R_a steht, $K_i \in R_a, i = 1 \ldots \|R_a\|$. Empfängt ein Knoten b dieses α zusammen mit der Chiffratliste, berechnet er mit all seinen Schlüsseln K_j von seinem Ring R_b mit $K_j \in R_b, j = 1 \ldots \|R_b\|$, die Chiffrate $C'_{K_j} = E_{K_j}(\alpha)$. Stimmt eines dieser C'_{K_j} mit einem der empfangenen C_{K_i} überein, so haben a und b einen gemeinsamen geheimen Schlüssel: genau $K_i = K_j$.

Finden zwei Nachbarn a und b auf diese Weise dennoch keinen gemeinsamen Schlüssel, $R_a \cap R_b = \varnothing$, oder möchte ein Knoten a mit einem anderen Knoten b kommunizieren,

der sich jedoch nicht in seiner direkten Nachbarschaft befindet, so kommen Indirektionsstufen zum Einsatz. Die Sensoren a und b versuchen, einen sogenannten *Schlüsselpfad* zueinander zu finden, der nur aus Knoten besteht, die bereits einen gemeinsamen Schlüssel auf ihren Key-Ringen miteinander haben. Ist ein solcher gefunden, kann a wieder eine Challenge α mit Chiffratliste $C_{K_i} = E_{K_i}(\alpha)$ an b schicken.

Neben dem dynamischen Hinzufügen neuer Knoten zu beliebigen Zeitpunkten, wie gerade beschrieben, schlägt Eschenauer und Gligor [88] ebenso eine Technik vor, nach der Knoten wieder sicher aus dem Netz entfernt werden können. Dies ist beim Einsatz zufallsverteilter Schlüssellisten ein schwieriges Problem: Falls ein Angreifer Zugriff auf einen ehemals dem Netz zugehörenden Knoten a bekommt, so gelangt er damit automatisch in den Besitz des kompletten Key-Rings R_a von a. Da Teile von R_a sich auch mit Key-Ringen anderer Knoten überschneiden, besteht die Gefahr, daß der Angreifer somit auch an die Schlüssel gelangt ist, mit denen andere Knoten ihre Kommunikation schützen.

Die Autoren Eschenauer und Gligor [88] empfehlen daher, beim Ausfall beziehungsweise dem Entfernen eines Knotens a netzweit alle Schlüssel, die Teil von R_a gewesen sind, zu streichen. Wie dies sicher initiiert und koordiniert wird, lassen Eschenauer und Gligor [88] dabei offen.

Die Arbeit Eschenauer und Gligor [88] soll an dieser Stelle stellvertretend für die Klasse aller *Random Key Pre-Distribution*-Protokolle stehen. Es existiert eine ganze Reihe von Varianten des ursprünglichen Schemas, wie beispielsweise Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251], die den ein oder anderen Aspekt der Grundidee optimieren, für die jedoch die im folgenden getroffenen Sicherheitsaussagen ebenfalls gelten.

B.2 Analyse

Bei Annahme des in Abschnitt 2.4, S. 24 definierten Angreifermodells wird zunächst klar, daß die Broadcast-Phase, in der Challenge α und Chiffratlisten C_{K_i} von a an Nachbarn b versendet werden, anfällig gegen Man-in-the-Middle Angriffe [34] ist: Da die Chiffratlisten nicht authentisch übertragen werden, kann ein Angreifer x, der über einen eigenen Key-Ring R_x verfügt, versuchen, den Versand der Chiffratliste von a nach b zu blockieren und mit jeweils a und b eigene Schlüssel aushandeln. Ab dann kann der Angreifer alle Nachrichten von a nach b und zurück blockieren, entschlüsseln, den enthaltenen Klartext mitlesen und verändern, sowie schließlich neu verschlüsseln und weiterleiten. Möchte er nur eher passiv dem Datenverkehr mitlauschen, so kann er die anfangs von a versandte Chiffratlisten einmalig so manipulieren, daß a und b sich auf einen Schlüssel einigen, den auch der Angreifer kennt. Dies geschieht wie folgt: Angreifer x hofft dabei, daß $R_a \cap R_b \cap R_x \neq \varnothing$ gilt. Nun ersetzt x die Chiffratliste C_{K_i} von a durch eine modifizierte Version C_{K_j}, wobei $\forall K_j \in C_{K_j} : K_j \in R_a \cap R_x$ gilt, und schickt diese an b weiter. Knoten b wählt jetzt zwangläufig einen Schlüssel für die gemeinsame Kommunikation mit a aus, den auch Angreifer x kennt und damit abhören und manipulieren kann.

Eine weitere Man-in-the-Middle Attacke ist in der zweiten Phase des Protokolls möglich, in der Knoten a mit Knoten b einen Schlüssel indirekt über einen Knoten x austauschen muß. Dieser Fall tritt, wie oben beschrieben, immer dann ein, sobald sich a und b entweder nicht in physikalisch unmittelbarer Nachbarschaft liegen oder keinen gemeinsamen Schlüssel auf ihren Key-Ringen finden. Will nun Knoten a mit Knoten b dennoch ein

Schlüssel austauschen, hofft er, mit Hilfe derjenigen (Nachbar-)Knoten, mit denen a bereits einen Schlüssel ausgetauscht hat, einen Schlüssel mit b zu finden. Dazu fragt er sukzessive seine Nachbarn x, ob diese bereits einen Schlüssel mit b besitzen. Wenn ja, dann schickt a einen neuen Schlüssel $K_{a,b}$ an x mit der Bitte um Weiterleitung nach b. Die komplette Kommunikation ist hierbei vor einem externen Angreifer geschützt, da $K_{a,b}$ zunächst mit dem Schlüssel zwischen a und x sowie danach mit dem Schlüssel zwischen x und b chiffriert wird. Falls nun aber einer von a's Nachbarn x ein Angreiferknoten sein sollte, so wird dieser zunächst a's Anfrage immer positiv beantworten und vorgeben einen gemeinsamen Schlüssel mit b zu besitzen. Dadurch wird es x möglich, Schlüssel $K_{a,b}$ und damit sämtliche Kommunikation zwischen a und b abzuhören.

Weiterhin leiden alle Protokolle, die auf zufallsverteilten Schlüssellisten basieren, unter dem Problem, daß Angreifer durch die Übernahme von Knoten i auch in den Besitz von Key-Ring R_i gelangen – und somit mit einer bestimmten Wahrscheinlichkeit auch an Schlüssel, die zur Kommunikationsabsicherung zweier anderer Knoten x und y im Einsatz sind. Damit kann ein Angreifer die Kommunikation zwischen x und y abhören.

Im folgenden wird nun gezeigt, daß alle *Random Key Pre-Distribution*-Protokolle selbst bei einer sehr kleinen Anzahl angreifender Knoten, welche die oben beschriebenen Techniken einsetzen, sehr schnell komplett kompromittierbar und unsicher sind. Im Rahmen dieser Arbeit werden daher verschiedene Szenarien simuliert, die bei variabler Anzahl von Angreifern die tatsächlich *gebrochenen Kommunikationsassoziationen* berechnen. Eine *Assoziation* bezeichnet an dieser Stelle die Kommunikation zwischen zwei Knoten, die mit Hilfe eines Schlüssels gesichert wird. Eine solche Assoziation gilt dabei genau dann als *gebrochen*, wenn sie mit einem Schlüssel gesichert wird, der auf einem Key-Ring eines korrumpierten Knoten liegt bzw. während des zur Assoziation gehörenden Schlüsselaustausches durch eine Man-in-the-Middle-Attacke manipuliert worden ist. Kurz: Eine Assoziation ist genau dann gebrochen, wenn der Angreifer den dazugehörenden Schlüssel kennt. Die Unterscheidung zwischen Assoziation und Schlüssel macht Sinn, weil bei *Random Key Pre-Distribution*-Protokollen häufig mehrere Kommunikationsassoziationen mit ein und demselben Schlüssel gesichert sind. Kennt der Angreifer einen Schlüssel, kann er damit mehrere Assoziationen abhören oder manipulieren.

Bevor auf die Simulationen und ihre Ergebnisse im Detail eingegangen wird, muß aber zunächst die Wahl von P und R genauer erklärt werden. Diese beiden Parameter sollen – so zumindest die Theorie – die Sicherheit des Verfahrens unmittelbar beeinflussen. Allgemein kann der Schlüsselaustausch im Netz nur dann sicher sein, wenn es möglich ist, daß jeder Knoten mit jedem anderen Knoten, unter Umständen über mehrere Hops, einen Schlüssel austauschen kann. Gilt dies nicht, müßten Knoten neue Schlüssel im Klartext an ihre Partner verschicken. Der Angreifer könnte die Schlüssel einfach mitlesen. Nun betrachte man die n einzelnen Sensorknoten im Netz als die Knoten in einem Graphen G. Zwischen zwei Knoten existiere genau dann eine Kante, wenn beide einen gemeinsamen Schlüssel auf ihren Key-Ringen besitzen. Damit der Schlüsselaustausch überhaupt sicher sein kann, muß G *zusammenhängend* sein: Es muß zwischen zwei beliebigen Knoten in G ein Pfad existieren. In Eschenauer und Gligor [88] wird gezeigt, daß die Wahrscheinlichkeit, mit der ein Graph G bestehend aus n Knoten zusammenhängend ist, zum Beispiel $99,999\%$, nur vom durchschnittlichen Aggregationsgrad g abhängt. Je höher der durchschnittliche Aggregationsgrad, desto höher die Wahrscheinlichkeit für einen zusammenhängenden Graphen. Anders formuliert: Jeder Sensorknoten muß mit einer bestimmten Mindestwahrscheinlichkeit p mit jedem anderen Knoten mindestens einen gemeinsamen

Schlüssel besitzen. Es ist außerdem notwendig, daß Knoten gemeinsame Schlüssel mit den Knoten in ihrer direkten Funkreichweite, in ihrer Nachbarschaft, besitzen. Die Anzahl der Knoten in Funkreichweite wird mit der *Dichte* d bezeichnet. Für einen zusammenhängenden Graphen kann nun mit Hilfe der Parameter n, g und d die Wahrscheinlichkeit p und im Anschluß damit die Größe des Key-Pools P sowie die Größe der Key-Ringe R aller Knoten bestimmt werden. Hierfür gibt Eschenauer und Gligor [88] Empfehlungen an. Überlicherweise wird dabei R durch den extrem limitierten Hauptspeicher des Sensors beschränkt: Der Hauptspeicher ist auch durch die eigentliche Anwendung bzw. den Dienst, der auf einem Knoten erbracht wird, sowie notwendige „Middleware“, wie ein Netzwerk-Stack, belegt. Deshalb steht für kryptographische Schlüssel oftmals nur ein Bruchteil des Hauptspeichers zur Verfügung. Steht eine obere Grenze für die Anzahl der Schlüssel $|R|$ fest, kann so schließlich $|P|$ bestimmt werden.

Der anschließende Abschnitt zeigt jedoch, daß trotz verschiedenster Wahl der möglichen Parameter, *Random Key Pre-Distribution*-Protokolle nicht sicher und resistent gegen selbst kleine Angreiferzahlen sind und damit nicht das Entwurfsziel *Sicherheit* erfüllen.

B.2.1 Simulation von Angreifern

Im Rahmen dieser Arbeit sind einige Simulationen von Eschenauer und Gligor [88] durchgeführt worden, um die generelle Sicherheit von Schlüssellisten-basierten Verfahren genauer zu evaluieren. Details über die zum Einsatz gekommene Simulationsumgebung GloMoSim finden sich in Abschnitt 3.4.3.1, S. 93

Die Bestimmung der für die Simulationen notwendigen Parameter sei wie folgt:

1. Die Gesamtanzahl n der Knoten wird mit $n = 1000$, $n = 3000$ und $n = 5000$ variabel gehalten. Zwar sind sicherlich auch Sensornetze mit $n = 10000$ Knoten und mehr denkbar, jedoch soll mit den durchzuführenden Simulationen lediglich überprüft werden, inwieweit die Anzahl aller Knoten im Sensornetz Einfluß auf die Sicherheit nehmen kann. Aus den Werten für n läßt sich der minimale Grad g bestimmen, für den gilt, daß damit der Graph G zusammenhängt [213]. Für $n = 1000$ Knoten gilt $g = 18$, bei $n = 3000$ gilt $g = 19$ und bei $n = 5000$ ist $g = 20$. Damit ist G zu $99,999\%$ zusammenhängend.

2. Einen realistischen Wert für die Dichte d, die Anzahl der Knoten, die sich in direkter Funkreichweite befinden, zu bestimmen, ist äußerst schwierig. Er hängt von der tatsächlichen Funkreichweite, der Verteilung der Knoten auf dem zu simulierenden Gebiet, sowie der Größe des Gebietes ab. Eschenauer und Gligor [88] schlagen als Dichte $d = 40$ Knoten vor. Neben $d = 40$ sind an dieser Stelle auch Simulationen mit $d = 20$ durchgeführt worden, um die Auswirkungen von d auf die Sicherheit zu untersuchen.

3. Als Größe für R berechnen Eschenauer und Gligor [88] jeweils 50 Schlüssel pro Knoten. Bei einer Schlüsselgröße von 64 Bit entspricht dies schon einem Speicherverbrauch von insgesamt 400 Bytes – das sind etwa 10% des kompletten Speichers, dem zum Beispiel ein MICA2-Mote [66] insgesamt zur Verfügung stehen. Bei Einsatz von 128 Bit Schlüssel würden gar 20% des Gesamtspeichers mit Schlüsseln belegt. Inwiefern 10% Hauptspeicherverbrauch akzeptabel sind, hängt vom konkreten Szenario und der verwendeten Applikation ab. Zusätzlich zu 50 Schlüsseln

sind auch Konfigurationen mit 75 Schlüssel pro Ring R simuliert worden, um festzustellen, ob dies unter Umständen die Sicherheit beeinflußt. 75 Schlüsseln würden dabei allerdings schon 600 Byte Hauptspeicher ($\approx 15\%$) entsprechen.

4. Da g und d feststehen, läßt sich auch die Wahrscheinlichkeit p berechnen, mit der zwei benachbarte Knoten sich einen Schlüssel auf ihren Key-Ringen teilen müssen. Bei $d = 20$ und $d = 40$ muß $p = 0,9$ respektive $p = 0,5$ sein. (Siehe Eschenauer und Gligor [88] für Details der Herleitung.)

5. Schließlich läßt sich damit die Pool-Größe P bestimmen. Tabelle B.1 zeigt die sich dadurch insgesamt ergebenen vier Konfigurationen K1 bis K4, mit denen im folgenden simuliert wird. Details dazu siehe auch Huttel [111].

Konfiguration	d	p	$\|R\|$	$\|P\|$
K1	20	0,9	50	1000
K2	20	0,9	75	2000
K3	40	0,5	50	3700
K4	40	0,5	75	8300

Tabelle B.1 Parameterwahl für Simulationen

In jedem Szenario wird mit einem variablen Anteil β von korrumpierten Knoten, zwischen $0,1\%$ und 20% der Gesamtknotenzahl, simuliert. Der Simulator generiert dabei zufällige Sensornetze mit einem zufälligen Aggregationsbaum. Anhand dieses Aggregationsbaums steht fest, welche Knoten miteinander Schlüssel austauschen müssen: Mit den Knoten auf ihrem Aggregationspfad. Der durchschnittliche Aggregationsgrad eines Knotens, die Anzahl der unmittelbaren Nachfolger im Baum, ist dabei normalverteilt mit einem Erwartungswert $\mu = 3$ und einer Standardabweichung von $\sigma = 1$. Um die Auswirkungen korrumpierter Knoten auf den Schlüsselaustausch neuer Knoten zu untersuchen, fügt die Simulation einem Netz der Größe $\frac{n}{2}$ Knoten, in dem $b*n$ Knoten korrumpiert sind, weitere $\frac{n}{2}$ Knoten hinzu. Insgesamt existieren damit im Netz der Größe n Knoten $b\%$ korrumpierte Knoten. Die korrumpierten Knoten versuchen mit den zuvor beschriebenen Attacken den Schlüsselaustausch der anderen Knoten anzugreifen. Haben sich zwei Knoten auf einen gemeinsamen Schlüssel geeinigt, existiert eine *Assoziation* zwischen ihnen, über die sie Daten austauschen können. Solch eine Assoziation gilt dann als vom Angreifer *gebrochen*, sobald mindestens ein korrumpierten Knoten den diese Assoziation schützenden Schlüssel kennt. Die Unterscheidung zwischen *Schlüssel* und *Assoziation* ist an dieser Stelle notwendig, da bei *Random Key Pre-Distribution*-Protokollen ein Schlüssel *mehrere* Assoziationen schützt.

Die Ergebnisse der Simulationen, gemittelt über jeweils 10 Durchläufe, zeigen die Abbildungen B.1 bis B.3. Die jeweils mit *Baseline* bezeichnete Kurve gibt jeweils den Anteil der Assoziationen an, der alleine aufgrund der Korrumpierung von Knoten, ohne die oben beschriebenen Attacken, im Durchschnitt bei den abgebildeten Simulationen gebrochen worden ist. Das bedeutet: Sobald ein Angreifer einen Knoten korrumpiert, gelangt er automatisch in den Besitz der Schlüssel dieses Knotens und bricht damit auch automatisch die Assoziationen, die *dieser Knoten* mit seinen Schlüsseln schützt. Die Baseline ist demnach unabhängig vom Schlüsselaustauschprotokoll. Sie steigt dabei nicht linear mit dem Anteil korrumpierter Knoten, sondern schneller. Dies liegt daran, daß durch die Korrumpierung von Knoten im „oberen“ Teil des Aggregationsbaums überdurchschnittlich viele

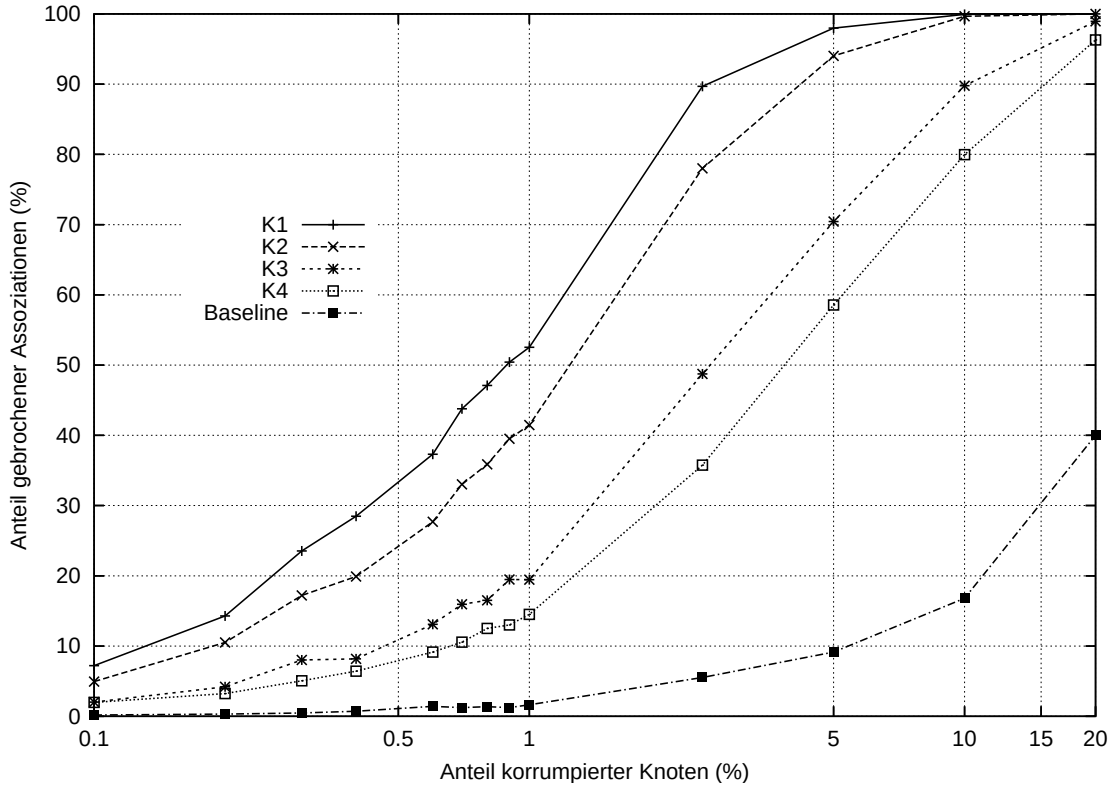


Abbildung B.1 Auswirkungen korrumpierter Knoten bei $n = 1000$

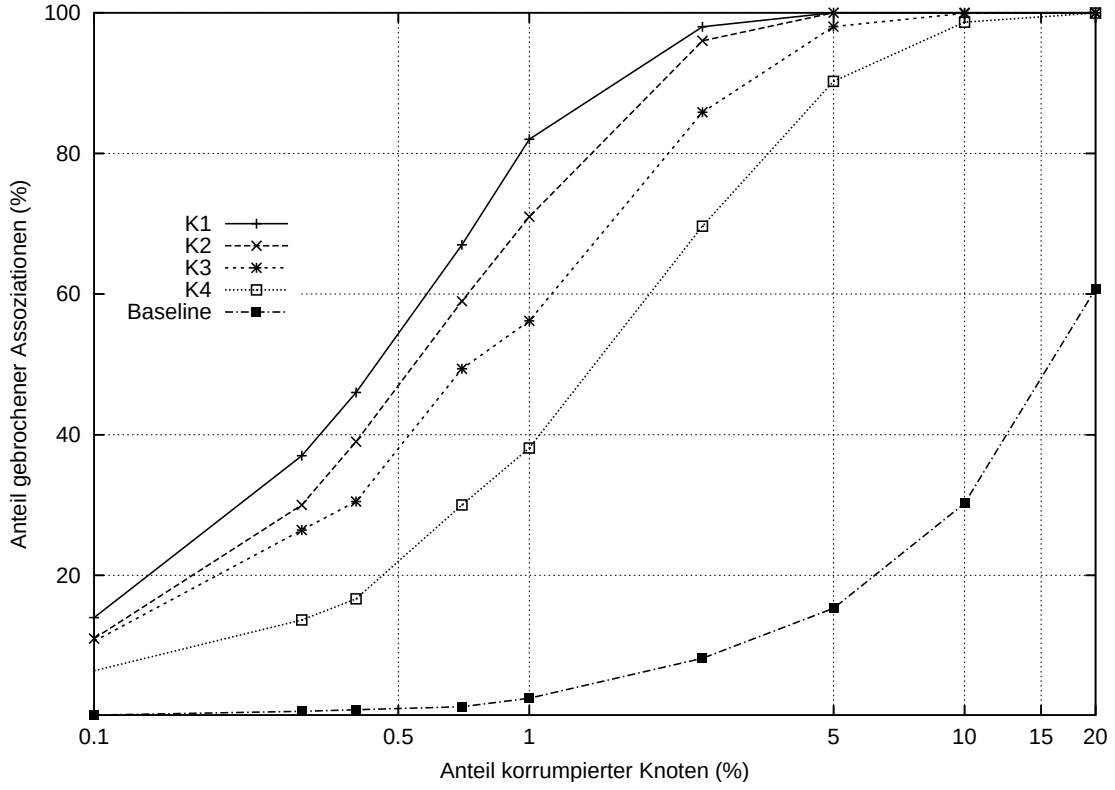


Abbildung B.2 Auswirkungen korrumpierter Knoten bei $n = 3000$

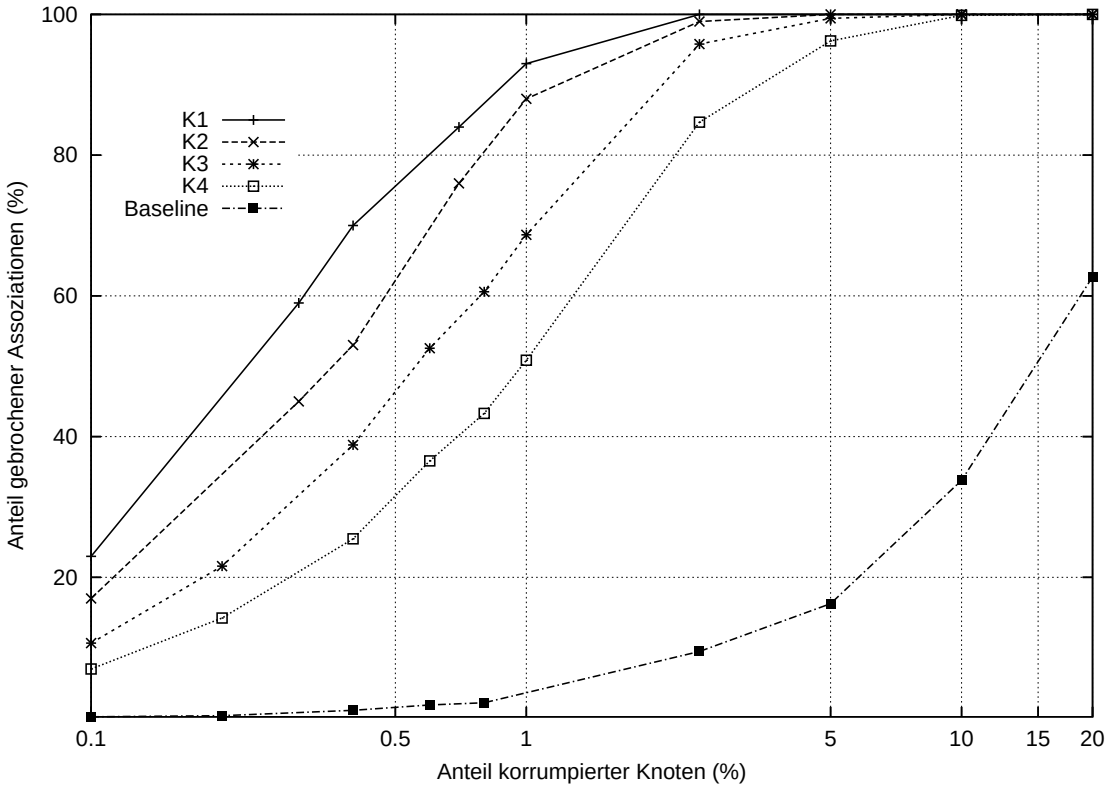


Abbildung B.3 Auswirkungen korrumpierter Knoten bei $n = 5000$

Assoziationen gebrochen werden. Ein Knoten, der Wurzel eines großen Teilbaums ist, hat Assoziationen mit allen Nachfolgern in seinem Teilbaum. Hingegen besitzt ein Knoten im „unteren“ Teil des Baumes weniger Assoziationen. Je höher der Anteil korrumpierter Knoten, desto höher die Wahrscheinlichkeit, daß mehr Knoten im oberen Teil des Baumes korrumpiert werden. Alle anderen Kurven geben den Anteil gebrochener Assoziationen wieder, welche die korrumpierten Knoten mit Hilfe der beschriebenen Angriffe erzielen konnten.

Wie eingangs beschrieben, steigt die Prozentzahl der gebrochenen Assoziationen überproportional mit der Anzahl der durch den Angreifer korrumpierten Knoten an, man beachte die logarithmische X-Achsenskalierung. Bei allen simulierten Konfigurationen und Knotenzahlen sind bereits bei etwa 4% Angreifern schon mehr als 50% aller Assoziationen gebrochen. Bei den in Eschenauer und Gligor [88] vorgeschlagenen Parametern sind sogar bei $< 0.5\%$ Angreifern etwa 30% aller Assoziationen unsicher. Zwar hilft die um immerhin die Hälfte erhöhte Pool-Größe etwas, rettet das Verfahren jedoch bei weiter ansteigenden Angreiferzahlen nicht.

An dieser Stelle soll allerdings der genaue Verlauf der Kurven nicht weiter diskutiert, sondern lediglich festgehalten werden, daß Protokolle basierend auf zufallsverteilten Schlüssellisten selbst durch geringen Anteil korrumpierter Knoten ein schlechtes Sicherheitverhalten zeigen. Dies liegt an der Funktionsweise der Protokolle: Durch den notwendigen Austausch von Schlüsseln über mehrere, unter Umständen korrumpierte Knoten, gelangt der Angreifer Man-in-the-Middle Attacken in den Besitz vieler Schlüssel. Außerdem sind vielfache mehrere Assoziationen mit ein und dem selben Schlüssel geschützt. Möglicherweise mag durch eine *enorme* Vergrößerung von P und R der Anteil gebrochener Asso-

ziationen reduziert werden, jedoch stehen die dafür notwendigen Speicherressourcen in Sensornetzen nicht zur Verfügung.

Die Arbeit Chan et al. [58] erkennt das grundsätzliche Problem des Schlüsselaustausches über Listen in der Theorie, betrachtet allerdings keine Man-in-the-Middle Angriffe. Als Verbesserung wird vorgeschlagen, daß zwei Knoten nicht nur genau einen gemeinsamen Schlüssel auf ihren Key-Ringen besitzen müssen, sondern $q > 1$. Aus diesen q Schlüsseln wird dann der Assoziationsschlüssel gebildet. Dies mag in einer etwas verbesserten Widerstandsfähigkeit gegenüber übernommenen Knoten resultieren, die dafür zwingend notwendige Mehraufwand von Speicher in Höhe von mehreren KByte ist aber für die meisten Sensoren unbezahlbar. In Moore [161] wird außerdem ein Angriff auf diese Verbesserung vorgeschlagen, der ebenso bei nur 5% Angreifern bereits 50% aller Assoziationen im Netz bricht. Die Veröffentlichung Huang et al. [109] untersucht die Auswirkungen zufälliger und gezielter Knotenübernahme auf die Sicherheit der bestehenden Kommunikationsassoziationen. Diese Auswirkungen, der daraus resultierende Anteil der gebrochenen Assoziationen, entspricht dem Verlauf der *Baseline*-Kurven in dieser Arbeit.

Faßt man die sämtlichen, oben aufgeführten Analysen zusammen, ist festzuhalten: Auf Schlüssellisten basierende Protokolle sind für den Einsatz in Sensornetzen nicht geeignet da *unsicher*.

B.2.2 Simulation von Dynamik

Der nun folgende Abschnitt untersucht das Verhalten der *Random Key Pre-Distribution*-Protokolle im Hinblick auf dynamisches Netzverhalten.

Sobald ein Knoten i das Netz verläßt, egal ob freiwillig oder gezwungenermaßen, müssen alle verbleibenden Knoten j sämtliche Schlüssel von i's Key-Ring R_i von ihren eigenen Key-Ringen R_j löschen. Fallen im Laufe eines Netzwerklebens mehrere oder gar viele Sensorknoten aus, so verkleinern sich die verschiedenen R_j allmählich. Als Konsequenz bedeutet das nicht nur, daß pro Knoten keine freien Schlüssel mehr zur Verfügung stehen, um neue Assoziationen zu anderen Knoten sicher eingehen zu können, sondern alte, bereits bestehende Assoziationen müssen als unsicher betrachtet werden. Denn gilt für einen Schlüssel K, daß er Teil des Key-Rings R_i eines ausgefallenen Knotens i gewesen ist, $K \in R_i$, aber ebenso auch $K \in R_j, K \in R_l$ und hat früher eine Assoziation zwischen Sensorknoten j und l bestanden, die mit K gesichert gewesen ist, so müssen j und l nun diese Assoziation als unsicher betrachten und einen neuen gemeinsamen Schlüssel für ihre Kommunikation austauschen. Hierbei kann jedoch ein Problem enstehen: Werden die Key-Ringe mit der Zeit immer kleiner, so stehen keine neuen Schlüssel mehr zur Verfügung – Knoten können dann nicht nur mit neuen Partnern nicht mehr (sicher) kommunizieren, sondern auch mit ihren bisherigen. Der Einfluß dynamischen Netzverhaltens auf die verfügbare Menge *gültiger* Schlüssel pro Knoten wird daher an dieser Stelle mit einigen Simulationen untersucht.

Mit den Parametern beziehungsweise Konfigurationen aus Tabelle B.1 werden Netze mit jeweils $n = 1000$, $n = 3000$ und $n = 5000$ Knoten simuliert. Die Abbildungen B.4 bis B.6 zeigen die Ergebnisse der Simulation, gemittelt über jeweils 10 Durchläufe. Auf der logarithmisch skalierten X-Achse ist der prozentuale Anteil an Ausfällen von Knoten aufgezeichnet. Es ist deutlich zu erkennen, daß selbst bei einem geringen Anteil Knotenausfall von nur 10% die Anzahl der Schlüssel auf unter 40% sinkt. Bei einem Knotenausfall von 20% sinkt die Anzahl der noch gültigen Schlüssel in allen Konfigurationen

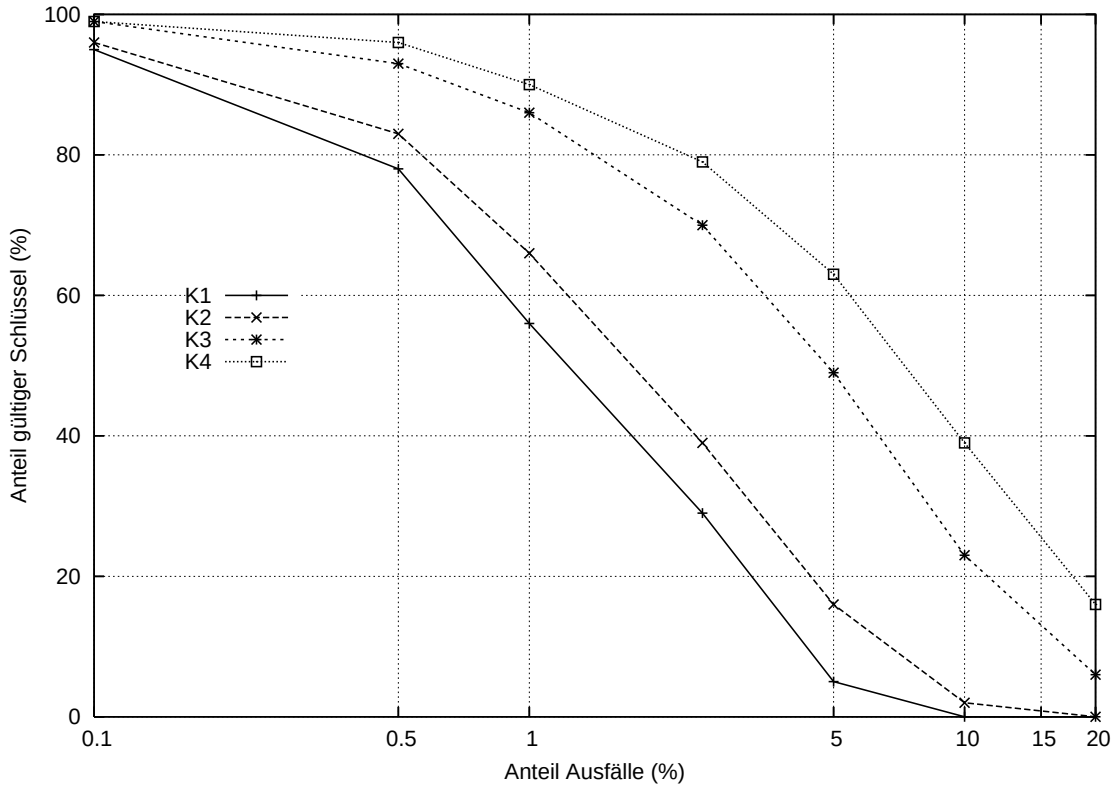


Abbildung B.4 Auswirkungen von Knotenausfall auf gültige Schlüssel, $n = 1000$

auf deutlich unter 20%, bei einigen Konfigurationen existieren überhaupt keinen gültigen Schlüssel mehr. Um diese Tatsache zu verdeutlichen: Fallen im Laufe des gesamten Sensornetzlebens 20% der Knoten aus oder werden zum Beispiel vom Wartungspersonal durch neue ersetzt, stehen den Knoten keine Schlüssel mehr zum Absichern ihrer Kommunikation zur Verfügung. Sichere Kommunikation wird damit unmöglich. Dies liegt am Prinzip der *Random Key Pre-Distribution*-Protokolle: Jeder Ausfall eines Knotens reduziert die Key-Ringe aller verbleibender Knoten im Netz. Bei Key-Ringen von $|R| = 50$ oder 75 steht somit bald kein gültiger Schlüssel mehr zur Verfügung. Auch hier würde eine Vergrößerung von R helfen, allerdings ist diese durch den Hauptspeicher nach oben beschränkt.

Abhilfe mag auch hier die deutliche Vergrößerung von P und R sein, aufgrund des beschränkten Speichers der Sensorknoten jedoch eher eine ungeeignete Maßnahme. Auf Schlüssellisten basierende Protokolle scheitern an dynamisches Szenarien – und erfüllen damit auch nicht das Entwurfsziel *Dynamik*.

B.2.3 Maximaler Speicherverbrauch

Zum Schluß dieses Abschnittes soll noch der maximale Speicherbedarf von *Random Key Pre-Distribution*-Protokollen untersucht werden. Auf weitere Details dazu gehen allerdings die Simulationen von Abschnitt 3.4.3, S. 91 ein.

Für verschiedene Knotenzahlen n sind wiederum Sensornetze simuliert worden, in denen Knoten anhand des Aggregationsbaums Schlüssel austauschen sollten. Den dabei gemessenen maximalen Speicherverbrauch zeigt Abbildung B.7. Je nach Protokollkonfiguration

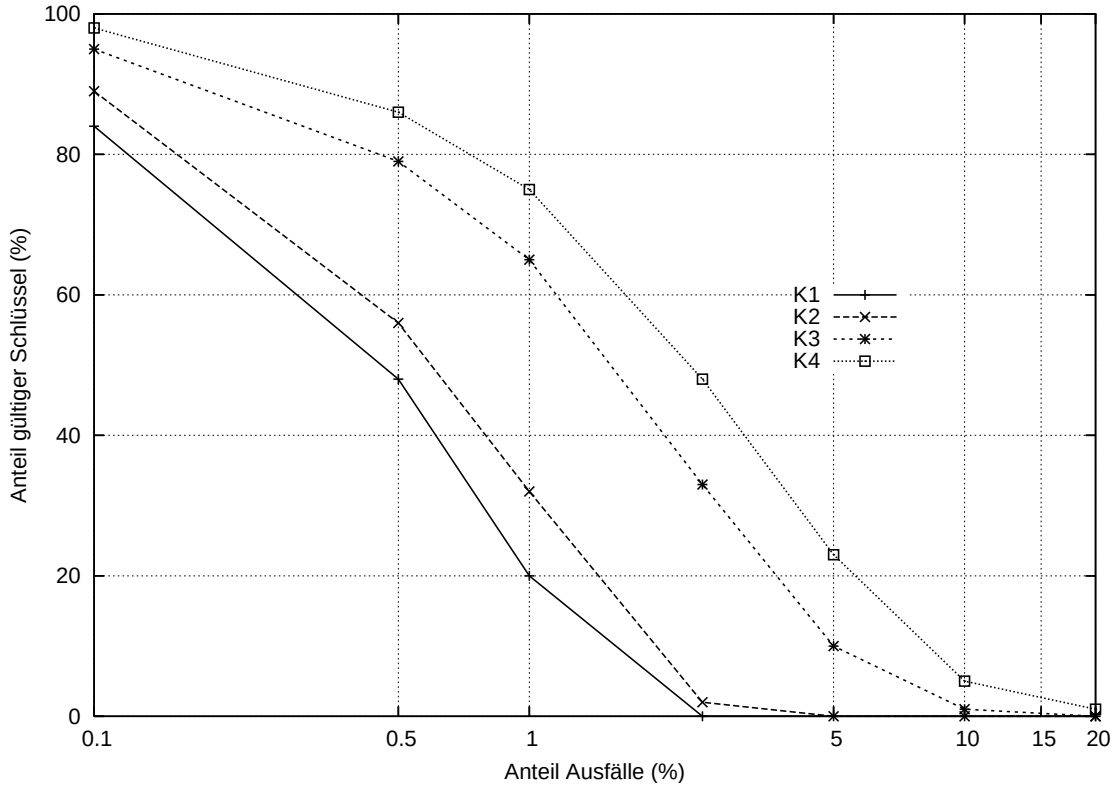


Abbildung B.5 Auswirkungen von Knotenausfall auf gültige Schlüssel, $n = 3000$

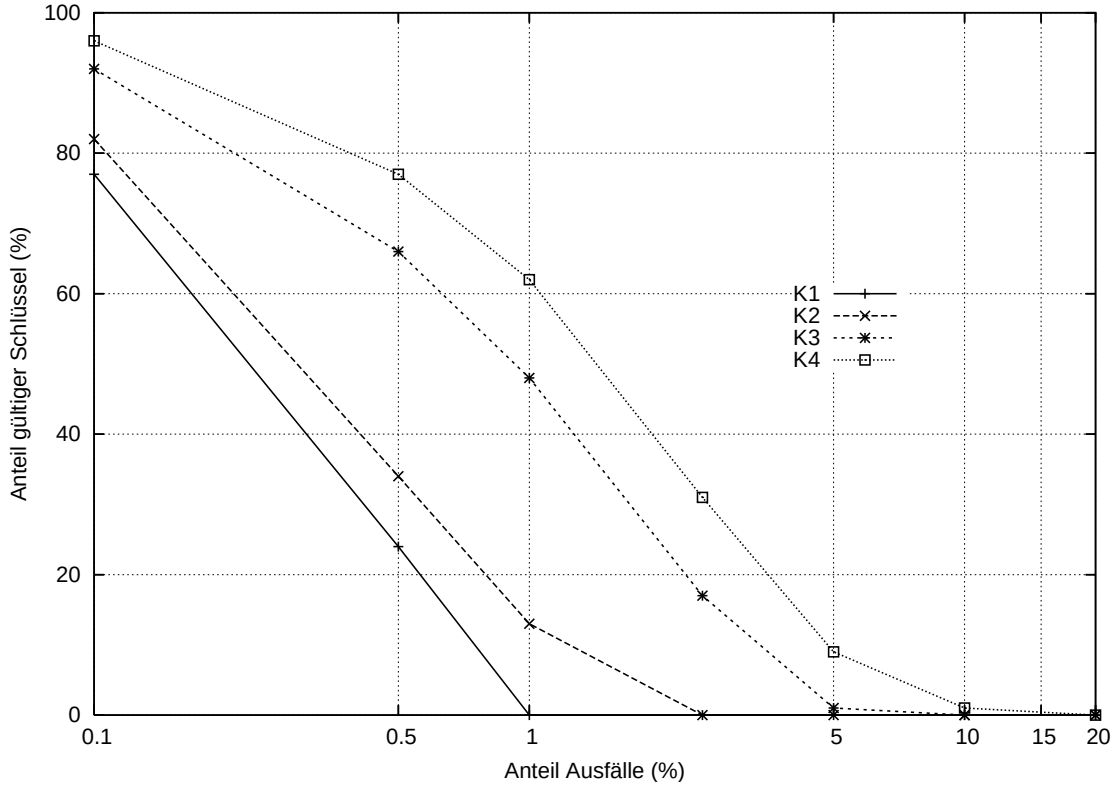


Abbildung B.6 Auswirkungen von Knotenausfall auf gültige Schlüssel, $n = 5000$

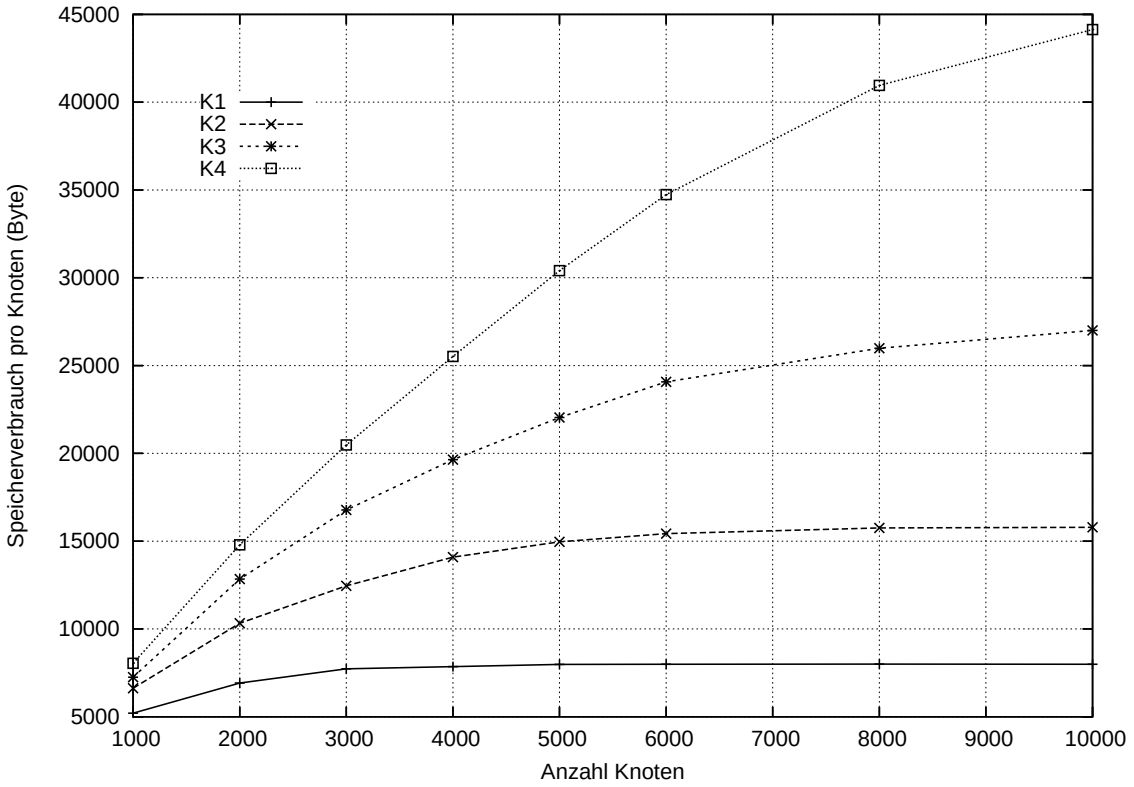


Abbildung B.7 Maximaler Speicherverbrauch pro Knoten

sind demnach bei einigen Knoten im Netz knapp 45 KByte an Hauptspeicher notwendig – unmöglich zum Beispiel bei MICA2 [66] Sensorknoten, die nur über insgesamt 4 KByte Hauptspeicher verfügen.

Der hohe Speicherbedarf läßt sich wie folgt erklären: Da gerade in großen Netzen einige Knoten, die miteinander kommunizieren möchten, keinen Schlüssel auf ihren Key-Ringen gemein haben, wird ihr Key-Ring jeweils um einen neuen Schlüssel erweitert. Das führt zu dem beobachteten Verhalten, daß einige Knoten extrem viele Schlüssel zusätzlich speichern und Key-Ringe sehr viel Hauptspeicher verbrauchen. Letztlich Weise führt dies dazu, daß in bestimmten Situationen auf Grund beschränkten Hauptspeichers keine neuen Schlüssel mehr ausgetauscht werden und sichere Kommunikation unmöglich ist.

Dieser Speicherbedarf widerspricht deutlich dem im Entwurfsziel *Effizienz* geforderten Verhalten.

B.2.4 Rücknahme von Schlüsseln

In den Situationen, in denen ein Angreifer einzelne Schlüssel oder sogar ganze Knoten in seinen Besitz bringen kann, stellt bei *Random Key Pre-Distribution*-Protokollen die Rücknahme von Schlüsseln ein großes Problem dar. Dadurch, daß die Schlüssel des kompromittierten Key-Rings mehrfach von anderen Knoten verwendet werden können, muß sichergestellt sein, daß nach der Kompromittierung kein legitimer Knoten im Netz mehr einen dieser Schlüssel benutzt. Dieser Vorgang nennt sich *Rücknahme von Schlüsseln* (engl. *Key Revocation*) [34].

Das Problem gliedert sich in zwei Teile:

1. Es muß auf sichere Art und Weise erkannt werden, daß ein oder mehrere, sich im Einsatz befindliche Schlüssel oder Knoten vom Angreifer korrumpiert worden sind.

2. Ist schließlich festgestellt, daß bestimmte Schlüssel korrumpiert wurden, so müssen alle Knoten, welche diese Schlüssel verwenden, *sicher* über diese Tatsache informiert werden. Außerdem sind Gegenmaßnahmen wie ein erneuter Schlüsselaustausch notwendig.

Es sei an dieser Stelle festgehalten, daß es sich hierbei nicht um ein allgemeines Problem von Protokollen zum Schlüsselaustausch handelt, sondern um ein grundsätzliches Sicherheitsproblem, dem *Erkennen* korrumpierter Knoten oder Schlüssel sowie danach dem sicheren Informieren des Netzes. Die Lösung dieses Sicherheitsproblems ist daher eigentlich keine Aufgabe eines Schlüsselaustauschprotokolls. Das Problem tritt allerdings auch nur dann in Erscheinung, wenn Schlüssel unter Umständen mehrfach zum Einsatz kommen – wie bei auf Schlüssellisten basierenden Protokollen. Bei dem in dieser Arbeit neu entworfenen Protokoll SKEY aus Abschnitt 3.3, S. 63 existieren zum Beispiel nur paarweise verschiedene Schlüssel auf jedem Knoten, so daß niemals Notwendig zur Rücknahme von Schlüsseln besteht. Arbeiten wie Chan et al. [56, 58], Di Pietro et al. [73, 74], Du et al. [81], Eschenauer und Gligor [88, 89], Huang et al. [109], Ito et al. [117], Lai et al. [135], Traynor et al. [220], Yu und Guan [251] müssen dementsprechend prinzipbedingt Key Revocation im Sensornetzen mitbetrachten. Für andere Arbeiten wie beispielsweise das in Abschnitt 3.3 entworfene SKEY gilt diese Problematik nicht.

Die Arbeiten schlagen sehr oberflächlich vor, über ein verteiltes Wahlverfahren abzustimmen, ob ein Knoten in der Zwischenzeit korrumpiert wurde. Das gesamte Wahlverfahren und damit auch ein möglicher Konsens werden ins Netz geflutet. Alternativ dazu kann die Datensenke oder Basisstation von sich aus entscheiden, ob Knoten korrumpiert werden oder nicht – auch diese Information wird dann ins Netz geflutet. In *jedem Fall* impliziert die Notwendigkeit zur Rücknahme von Schlüsseln bei *Random Key Pre-Distribution*-Protokollen einiges an Mehraufwand durch zusätzlich zu versendende Nachrichten, ein Nachteil bzgl. des Entwurfsziels *Effizienz*. Daher sind die Wahlverfahren zur Rücknahme von Schlüsseln in dieser Arbeit nicht weiter untersucht worden.

C. Simulationen zum 802.11 MAC-Verhalten

In den Kapiteln 3 und 4 sind Simulationen mit großen Knotenzahlen von bis zu $n = 10000$ Knoten durchgeführt worden. Bei der Simulation solch großer Sensornetze steigt die Simulationszeit pro Simulation unter Mitbeachtung von zufälligem elektromagnetischem Rauschen, zufälligen Bitfehlern, IEEE 802.11 CSMA/CA Zugriffen, Paketverlusten, Kollisionen und daraus resultierenden Übertragungswiederholungen auf mehrere Tage an.

Aus diesem Grund sind in den Simulationen die Routinen zum Messen solcher MAC-Aspekte aus GloMoSim auskommentiert worden. Gemessen worden ist – neben Speicherverbrauch sowie Energieverbrauch für kryptographische Operationen – nur der Energieverbrauch, der durch das Übertragen von TinyOS-Nachrichten mit jeweils 56 Byte Länge entsteht. Vereinfacht ist angenommen worden, daß jeweils eine komplette 56 Byte TinyOS-Nachricht im Simulator einer 802.11 MAC-Dateneinheit entspricht.

Anmerkung

Die 56 Byte Nachricht mit 29 Byte Nutzdaten, die TinyOS spezifiziert, entspricht in TinyOS einer „MAC-Dateneinheit“. Genau diese 56 Byte werden durch die Funkschnittstelle des MICA2-Knotens übertragen. Der Begriff Dateneinheit wird an dieser Stelle als Synonym für die im MAC-Kontext häufig geläufigen Ausdrücke „Rahmen“ oder „Frame“ verwendet.

C.1 Simulationen mit 802.11 MAC

Um zu untersuchen, inwieweit mögliche Kollisionen und Paketverlust den Energieverbrauch der präsentierten Protokolle beeinflussen beziehungsweise *erhöhen*, sind sowohl von SKEY als auch von ESAWN weitere Simulationen mit $n = 100$ bis $n = 1000$ Knoten durchgeführt worden. In diesen Simulationen sind jeweils alle versendeten und damit Energiekosten erzeugenden CSMA/CA MAC-Dateneinheiten mitsimuliert worden: Das sind sowohl RTS-, CTS- als auch die eigentlichen Nutzdaten- und ACK-Dateneinheiten.

Vereinfacht ist für jede Übertragung dieser MAC-Dateneinheiten jeweils ein Energieverbrauch von $245\ \mu$As berechnet worden, siehe Abschnitt 3.4.3.1.

Um möglichst eine *obere* Schranke für den Energiemehraufwand durch Mitsimulieren der MAC-Dateneinheiten zu bestimmen, haben sich sämtliche Sensorknoten in direkter Funkreichweite zueinander befunden. Auf diese Weise ist die Gefahr von Kollisionen bei gleichzeitigen Funkübertragungen am größten, genau wie die Zahl der Wiederholungen und damit der Energieverbrauch.

Da sich der Energieverbrauch nicht nur durch Kollisionen, sondern auch durch fehlerhaft übertragene Dateneinheiten erhöht, ist zunächst mit einem bestimmten elektromagnetischem Rauschen und daraus resultierendem Paketverlust simuliert worden. Als Wert kam die Voreinstellung von GloMoSim für 802.11, die *Noise Figure* von $3,999 \cdot 10^{-17}$ mW, zum Einsatz. Dies entspricht $-163,98$ dBm. GloMoSim bietet als Alternative zu einem Wert für elektromagnetisches Rauschen das Spezifizieren einer sogenannten *Bit Error Rate* (BER, durchschnittliche Bit-Fehlerrate). Da eine Bit Error Rate einen wesentlich anschaulicheren Wert als eine Noise Figure darstellt, sind im Anschluß noch Simulationen mit BER-Werten zwischen 10^{-3} und 10^{-6} gefolgt.

C.2 Simulationsergebnisse

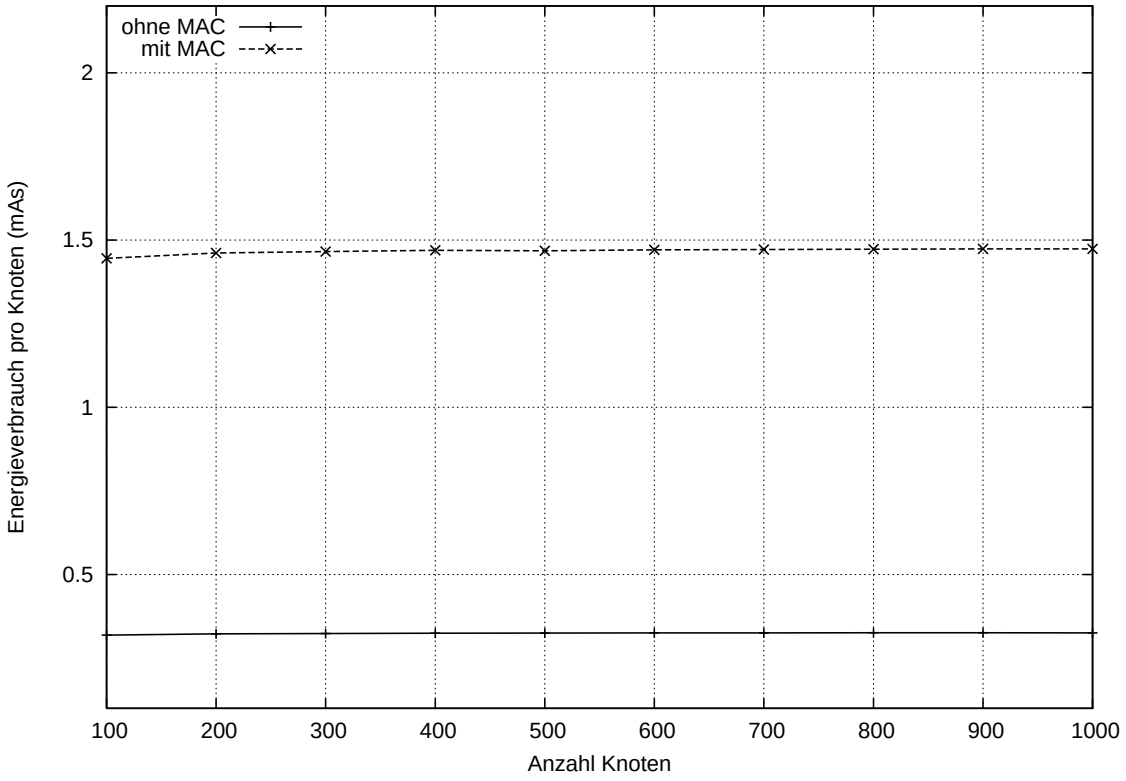
Zunächst zu den Simulationen bei einer Noise Figure von $3,999 \cdot 10^{-17}$ mW: Die Abbildungen C.1 und C.2 zeigen den Energieverbrauch von ESAWN jeweils pro Knoten. Abbildung C.3 zeigt anschließend den Energieverbrauch von SKEY. Die unteren Kurven geben dabei in allen Abbildungen den Energieverbrauch ohne Betrachtung der MAC-Dateneinheiten an, die oberen Kurven jeweils den mit allen gemessenen MAC-Dateneinheiten. Die relative Standardabweichung aller durchgeführten Simulationen hat zwischen $0,4\%$ bei $n = 1000$ und 4% bei $n = 100$ gelegen.

Abbildung C.1 zeigt den Energieaufwand bei ESAWN mit $k = 1,2$ und $p = 100\%$ zwischen $n = 100$ und $n = 1000$ Sensorknoten. Abbildung C.2 zeigt zum Vergleich $k = 1,2$ bei $p = 50\%$. Da die Energieverbräuche unterschiedlicher δ untereinander parallel verlaufen, siehe zum Beispiel Abbildung 4.14, S. 158, zeigen die Abbildungen jeweils nur $\delta = 3$. Abbildung C.3(a) stellt den Energieverbrauch von SKEY bei $k = 1$, Abbildung C.3(b) bei $k = 2$ dar.

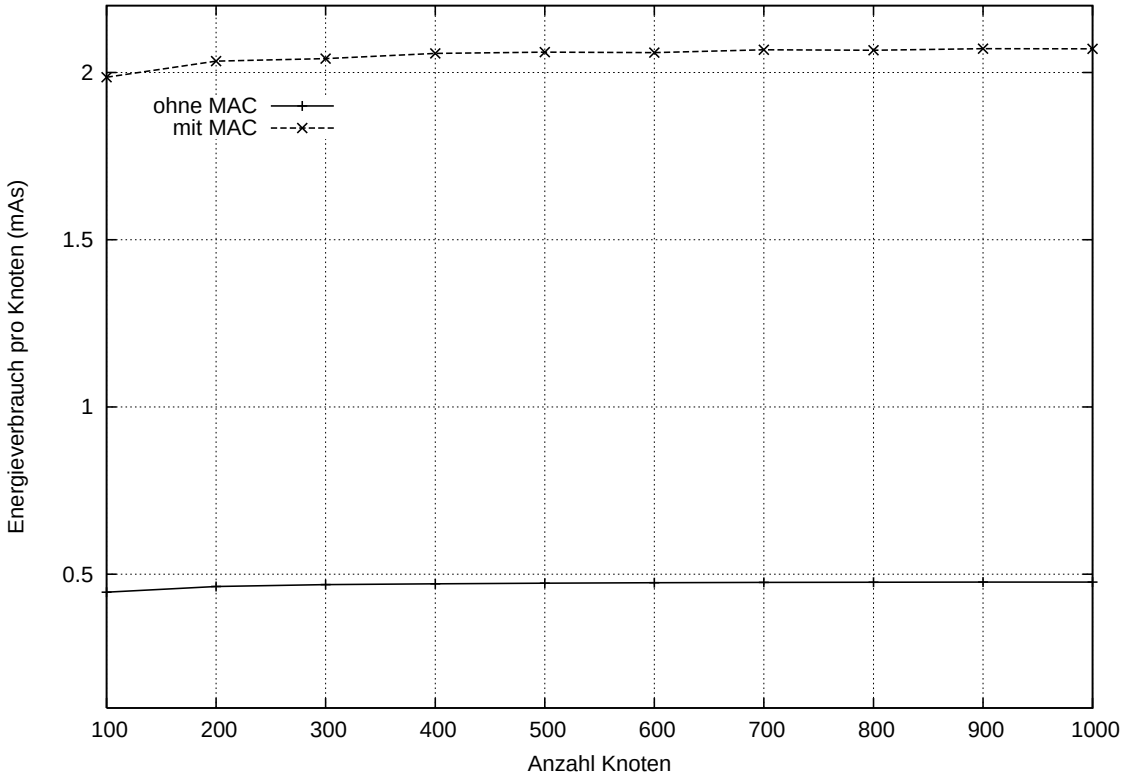
Betrachtet man sämtliche Abbildungen, so läßt sich zusammenfassen, daß sowohl bei ESAWN als auch bei SKEY das Verhältnis zwischen Energieverbrauch mit MAC zu dem ohne MAC bei knapp unter $4,5 : 1$ liegt. Für jede Nutzdateneinheit müssen zusätzlich mindestens eine RTS-Dateneinheit, eine CTS-Dateneinheit sowie eine ACK-Dateneinheit übertragen werden. Pro Nutzdateneinheit werden selbst bei der ungünstigen Knotenkonfiguration in gegenseitiger Funkreichweite demnach im Durchschnitt nur $0,5$ zusätzliche Dateneinheiten durch mögliche Kollisionen oder Bitfehler übertragen – ein geringer Wert. Dies zeigt, daß beim Protokollablauf von SKEY und ESAWN *wenig* Kollisionen entstehen und die Anzahl der Kollisionen unabhängig von der Knotenzahl ist. Beide Protokolle versenden Nachrichten demnach sehr sequentiell.

Variierende BER

Im Anschluß sind von ESAWN und SKEY noch Simulationen mit unterschiedlichen Bit Error Rates durchgeführt worden. Sowohl ESAWN als auch SKEY sind dabei wie oben parametrisiert worden.

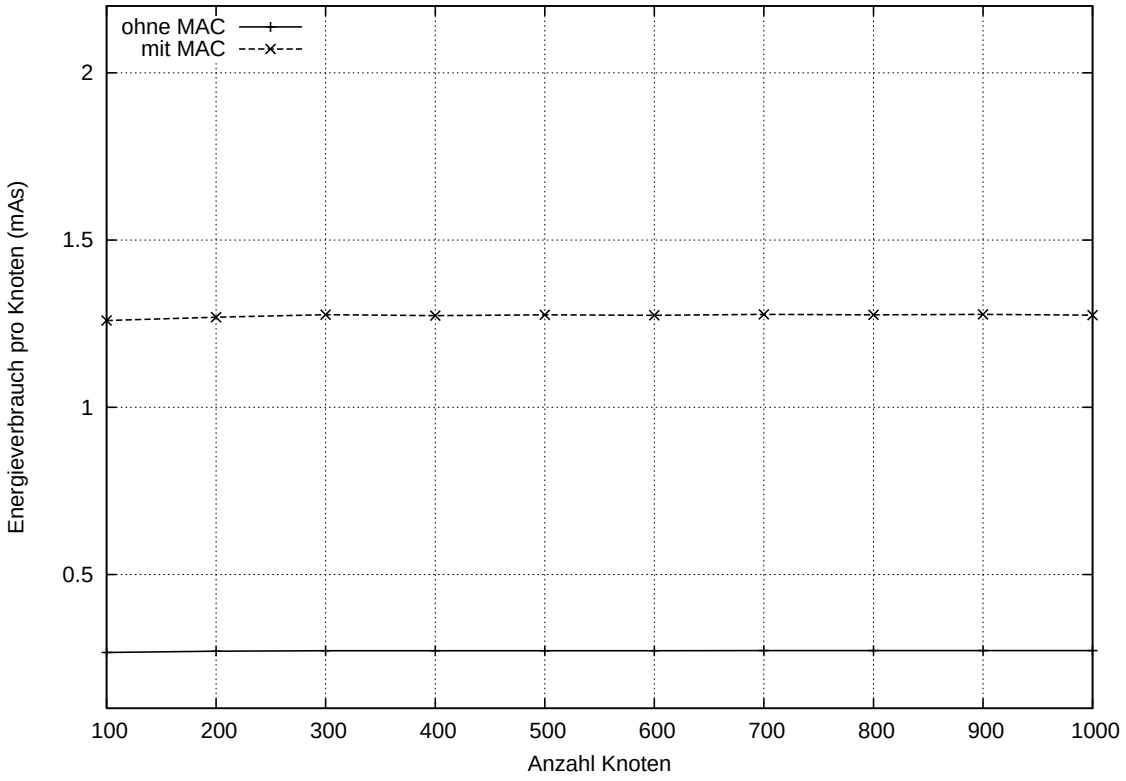


(a) $k = 1$

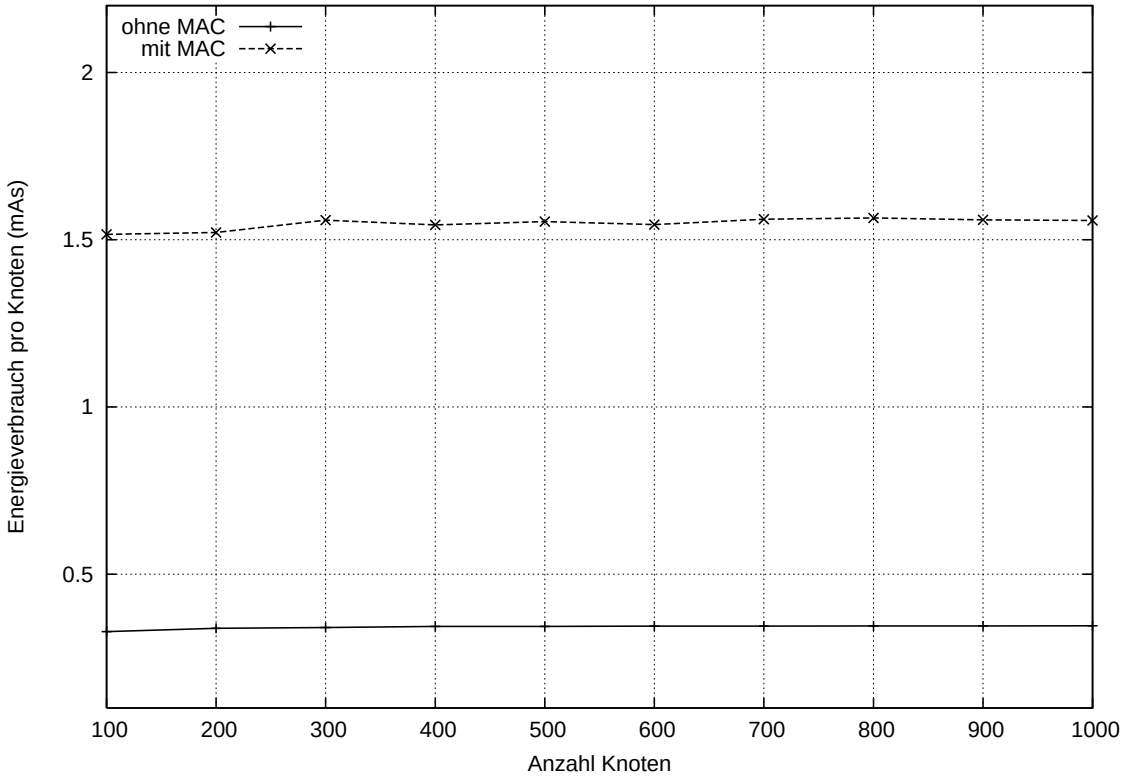


(b) $k = 2$

Abbildung C.1 ESAWN-Energieaufwand ohne und mit 802.11 MAC, jeweils $p = 100\%$

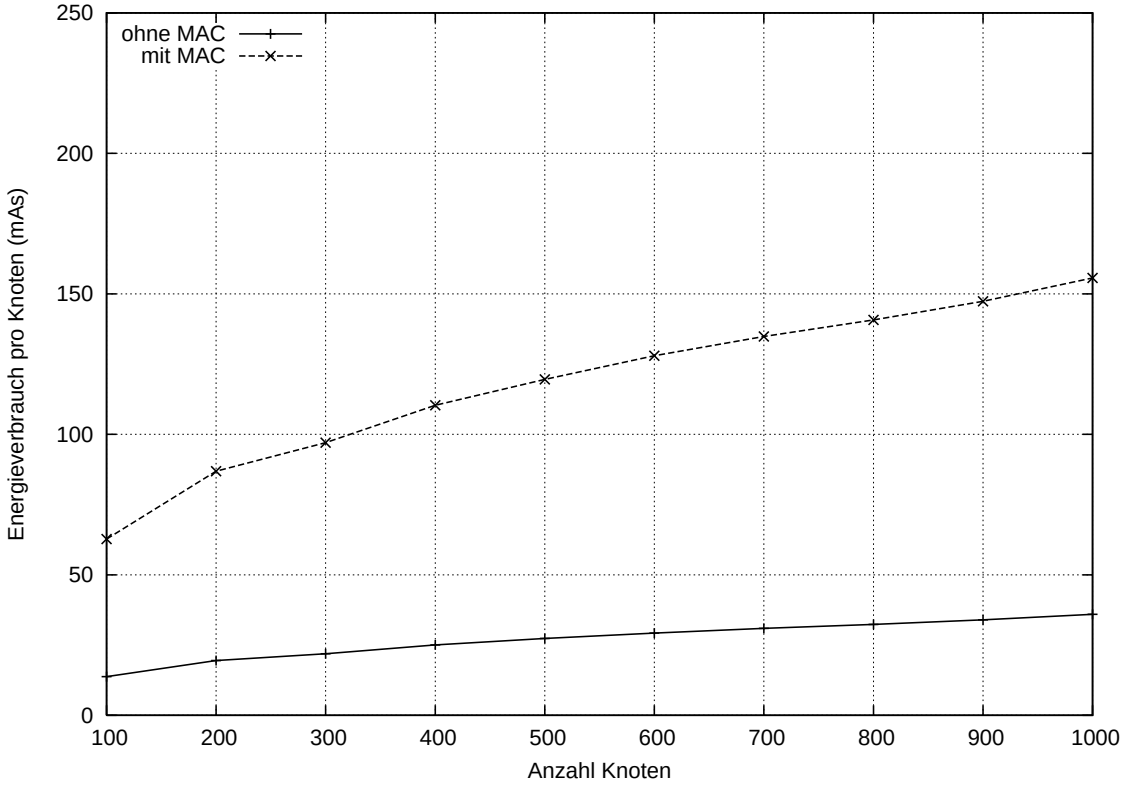


(a) $k = 1$

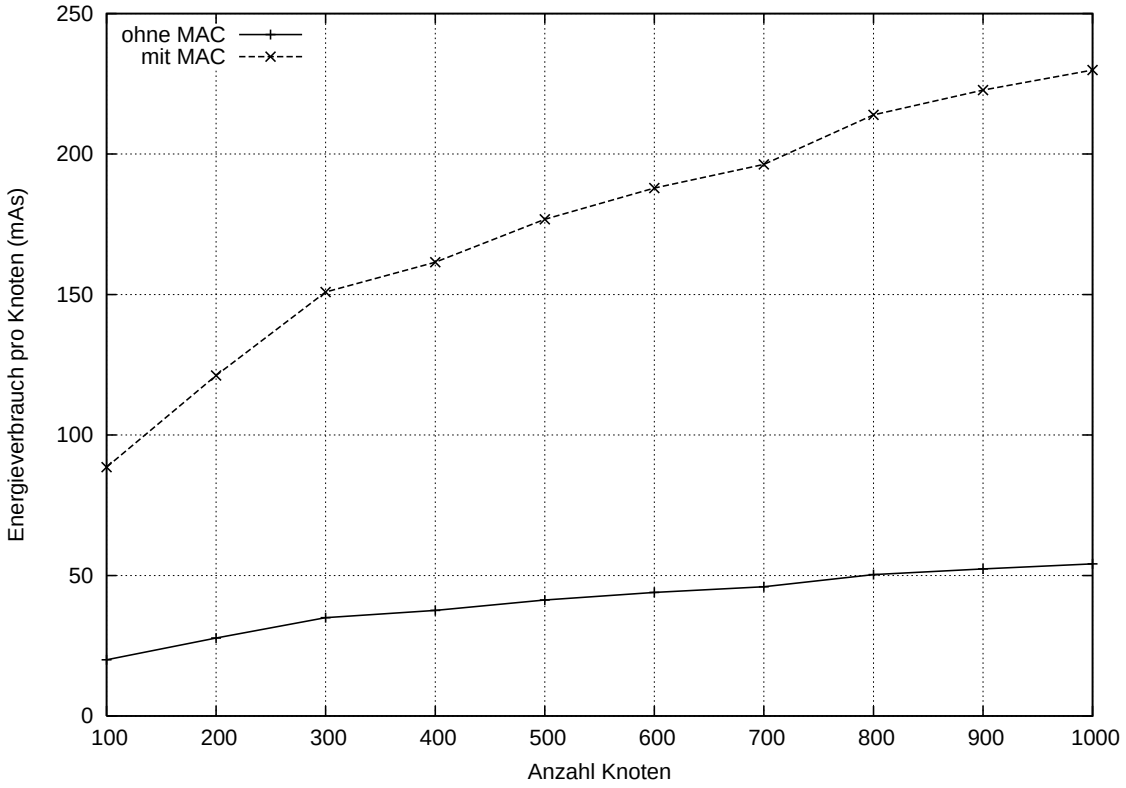


(b) $k = 2$

Abbildung C.2 ESAWN-Energieaufwand ohne und mit 802.11 MAC, jeweils $p = 50\%$



(a) $k = 1$



(b) $k = 2$

Abbildung C.3 SKEY-Energieaufwand ohne und mit 802.11 MAC

BER	10^{-6}	10^{-5}	10^{-4}	10^{-3}
Verhältnis (MAC vs. ohne MAC)	$4,5$	$4,57$	$5,26$	$28,19$

Tabelle C.1 Auswirkungen von Bit Error Rates auf das Energieverhältnis

Auch mit diesen Simulationen mit unterschiedlichen Bit Error Rates ist das Verhältnis der Energieverbräuche zueinander *unabhängig* von der Knotenzahl geblieben. Anstatt weiterer Graphen für unterschiedliche Bit Error Rates faßt Tabelle C.1 die durchschnittlichen Verhältnisse bei variierenden Werten für die BER zusammen. Dabei ist zu erkennen, daß der Standardwert für elektromagnetisches Rauschen bei GloMoSim von $3,999 \cdot 10^{-17}$ mW im Durchschnitt einer BER von $\approx 10^{-6}$ entspricht. Grundsätzlich läßt sich festhalten, daß eine höhere Wahrscheinlichkeit für Bit-Fehler auch in einem größeren Verhältnis zwischen den Energieverbräuchen führt. Dies ist relativ offensichtlich, da durch höheren Fehlerwahrscheinlichkeiten um so häufiger Dateneinheiten wiederholt übertragen werden müssen.

C.3 Zusammenfassung

Durch den Versand von RTS-, CTS- und ACK-Dateneinheiten auf MAC-Ebene sowie durch Kollisionen und Rauschen steigt der Energieverbrauch zum erfolgreichen Versenden einer TinyOS-Dateneinheit an.

Das Verhältnis zwischen dem Energieverbrauch mit und ohne Betrachtung von MAC ist dabei *unabhängig* von der Knotenzahl konstant und nur von der Bit-Fehlerrate beziehungsweise dem elektromagnetischen Rauschen abhängig. Die Anzahl der Kollisionen nimmt nicht überproportional mit steigender Knotenzahl zu.

Die von GloMoSim standardmäßig für 802.11 CSMA/CA vorgegebene Noise Figure entspricht in ihrem Gesamtenergieverbrauch einer BER von 10^{-6} und einem Energiemehraufwand von Faktor $4,5$. Zu beachten ist dabei allerdings, daß sich die durchschnittliche Bit-Fehlerrate oder Noise Figure für die Funkschnittstelle und das einfache sogenannte *pre-B-MAC*-Protokoll [226] ohne RTS, CTS und ACK der MICA2-Knoten unter Umständen ganz anders verhält als das in GloMoSim implementierte IEEE 802.11 CSMA/CA. Der in den Simulationen gemessene Faktor von $4,5$ hat damit nur eine eher begrenzte Aussage. Der Energieverbrauch der MICA2-Knoten wäre wahrscheinlich deutlich geringer – als weitere ist aber spekulativ und soll in dieser Arbeit nicht beachtet werden.

Von Bedeutung sei an dieser Stelle lediglich festgehalten, daß das gemessene Verhältnis konstant in Abhängigkeit von der Knotenzahl bleibt, gering ist und mit steigender Knotenzahl nicht die Anzahl der Kollisionen überproportional zunehmen. Die Energieaussagen und Energieverläufe in den Kapiteln 3 und 4 sind damit plausibel.

D. Simulationen zu k und p

Im folgenden zeigen die Tabellen D.1 bis D.3 die Simulationsergebnisse der Untersuchungen über den Zusammenhang zwischen der *Wahrscheinlichkeit für eine korrekte Aggregation* (WKA) sowie der ESAWN-Protokollparameter p und k bei verschiedener Gesamtknotenanzahl n und $\delta = 4$. Siehe hierzu auch Abschnitt 4.4.3.2, S. 158.

Die Tabellen ermöglichen dem Benutzer die aus Energiesicht sparsamste Wahl von k und p zu finden, die im Durchschnitt in einem Netz mit n Knoten und $\delta = 4$ bei $\beta = 1\%$, 10%, 20% eine gewünschte Mindest-WKA erreicht.

WKA	$\geq 50\%$		$\geq 60\%$		$\geq 70\%$		$\geq 80\%$		$\geq 90\%$		100%	
n	k	p	k	p	k	p	k	p	k	p	k	p
100	1	21	1	25	1	27	1	30	1	59	2	100
200	1	24	1	30	1	30	1	60	1	80	2	100
300	1	27	1	31	1	52	1	73	1	85	2	100
400	1	30	1	50	1	64	1	78	1	91	2	100
500	1	44	1	58	1	71	1	83	1	92	2	100
600	1	56	1	66	1	77	1	86	1	95	2	100
700	1	61	1	70	1	82	1	88	1	95	2	100
800	1	67	1	75	1	83	1	90	1	96	2	100
900	1	70	1	78	1	84	1	90	1	97	2	100
1000	1	74	1	78	1	87	1	93	1	97	2	100
2000	1	88	1	91	1	94	1	97	1	99	2	100
3000	1	92	1	95	1	97	1	98	1	100	3	100
4000	1	94	1	96	1	98	1	100	1	100	3	100
5000	1	96	1	97	1	98	1	100	2	100	3	100
6000	1	97	1	98	1	99	1	100	2	100	3	100
7000	1	97	1	98	1	100	1	100	2	100	3	100
8000	1	98	1	99	1	100	1	100	2	100	3	100
9000	1	99	1	99	1	100	2	100	2	100	3	100
10000	1	99	1	100	1	100	2	100	2	100	3	100

Tabelle D.1 Energetisch günstigste k und p bei verschiedenen n und $\beta = 1\%$

WKA	$\geq 50\%$		$\geq 60\%$		$\geq 70\%$		$\geq 80\%$		$\geq 90\%$		100%	
n	k	p	k	p	k	p	k	p	k	p	k	p
100	1	78	1	86	1	89	1	97	2	97	3	100
200	1	93	1	96	2	94	2	97	2	99	3	100
300	1	99	2	95	2	96	2	98	2	100	3	100
400	2	94	2	96	2	97	2	99	2	100	4	100
500	2	96	2	98	2	98	2	99	2	100	4	100
600	2	97	2	98	2	99	2	100	2	100	4	100
700	2	97	2	98	2	99	2	100	3	100	4	100
800	2	97	2	99	2	99	2	100	3	100	4	100
900	2	98	2	99	2	100	2	100	3	100	4	100
1000	2	98	2	99	2	100	2	100	3	100	4	100
2000	2	100	2	100	2	100	3	100	3	100	5	100
3000	2	100	3	100	3	100	3	100	3	100	5	100
4000	2	100	3	100	3	100	3	100	3	100	5	100
5000	3	100	3	100	3	100	3	100	3	100	5	100
6000	3	100	3	100	3	100	3	100	3	100	5	100
7000	3	100	3	100	3	100	3	100	4	100	5	100
8000	3	100	3	100	3	100	3	100	4	100	6	100
9000	3	100	3	100	3	100	3	100	4	100	6	100
10000	3	100	3	100	3	100	3	100	4	100	6	100

Tabelle D.2 $\beta = 10\%$

WKA	$\geq 50\%$		$\geq 60\%$		$\geq 70\%$		$\geq 80\%$		$\geq 90\%$		100%	
n	k	p	k	p	k	p	k	p	k	p	k	p
100	1	95	2	90	2	94	2	96	2	99	3	100
200	2	95	2	97	2	98	2	100	3	99	3	100
300	2	98	2	99	2	100	3	99	3	100	4	100
400	2	99	2	100	3	99	3	99	3	100	4	100
500	2	99	3	99	3	99	3	100	3	100	4	100
600	2	100	3	99	3	100	3	100	3	100	4	100
700	3	99	3	99	3	100	3	100	3	100	4	100
800	3	99	3	100	3	100	3	100	3	100	4	100
900	3	99	3	100	3	100	3	100	4	100	5	100
1000	3	99	3	100	3	100	3	100	4	100	5	100
2000	3	100	3	100	3	100	4	100	4	100	5	100
3000	3	100	4	100	4	100	4	100	4	100	5	100
4000	4	100	4	100	4	100	4	100	4	100	6	100
5000	4	100	4	100	4	100	4	100	5	100	6	100
6000	4	100	4	100	4	100	4	100	5	100	6	100
7000	4	100	4	100	4	100	4	100	5	100	6	100
8000	4	100	4	100	4	100	5	100	5	100	6	100
9000	4	100	4	100	4	100	5	100	5	100	6	100
10000	4	100	4	100	4	100	5	100	5	100	6	100

Tabelle D.3 $\beta = 20\%$

E. Symbole

Tabelle E.1 listet alle mathematischen Symbole, die in dieser Arbeit kapitelübergreifend verwendet werden und von besonderer Bedeutung sind, auf. In den einzelnen Kapiteln werden die dort jeweils zusätzlich zum Einsatz kommenden Symbole erklärt.

Symbol	Bedeutung
δ	der durchschnittliche oder erwartete *Aggregationsgrad* eines Aggregationsbaums, $\delta > 1$ in dieser Arbeit
n	die Gesamtanzahl aller Knoten im Netz, $n \approx \frac{\delta^{h+1}-1}{\delta-1}$ in dieser Arbeit
Λ	Die Menge aller Aggregationsknoten im Netz, $\Lambda = \{x_i \| x_i \in V, \delta_{x_i} > 0\}$
h	die durchschnittliche oder erwartete *Höhe* eines Aggregationsbaums, $h \approx \log_\delta(1 + (\delta - 1)n) - 1$ in dieser Arbeit
$\mathcal{B}$	die *Gesamtanzahl* aller Knoten, die der Angreifer korrumpiert hat, $\mathcal{B} < n$ in dieser Arbeit
β	der Anteil oder *Prozentsatz* korrumpierter Knoten, $\beta = \frac{\mathcal{B}}{n} < 1$ in dieser Arbeit
$\mathbb{P}$	der *Aggregationspfad* eines Knotens: die Menge aller Knoten, die Vorgänger oder Nachfolger von ihm im Aggregationsbaum sind
k	die bei SKEY und ESAWN maximale Anzahl hintereinander auf einem Pfad $\mathcal{P}$ liegenden, korrumpierten Knoten, gegen die SKEY und ESAWN noch sicher funktionieren, $k \leq h$

Tabelle E.1 Symbole und ihre Bedeutung

Literaturverzeichnis

[1] ABADI, M. und FEIGENBAUM, J.: *Secure circuit evaluation*. In: *Journal of Cryptology*, Vol. 2, Nr. 1, Seiten 1–12, 1990. ISSN 0933-2790

[2] ACHARYA, M., GIRAO, J. und WESTHOFF, D.: *Secure Comparison of Encrypted Data in Wireless Sensor Networks*. In: *Proceedings of IEEE Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. Seiten 47–53. Riva del Garda, Italien, 2005, IEEE Press. ISBN 0-7695-2267-X

[3] ACHTZEHN, A., BENENSON, Z. und ROHNER, C.: *Implementing Agreement Protocols in Sensor Networks*. In: *Proceedings of International Conference on Mobile Adhoc and Sensor Systems*. Seiten 858–863, IEEE Press, 2006. ISBN 1-4244-0507-6

[4] ÁCS, G., BUTTYÁN, L. und VAJDA, I.: *Modelling adversaries and security objectives for routing protocols in wireless sensor networks*. In: *Proceedings of ACM Workshop on Security of Ad hoc and Sensor Networks*. Seiten 49–58. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-554-1

[5] „ALEPH ONE“: *Smashing The Stack For Fun And Profit*. Phrack.org, Volume Seven, Issue Forty-Nine, File 14 of 16, 2001.
http://www.phrack.org/archives/49/P49-14

[6] ALKASSAR, A., STÜBLE, C. und SADEGHI, A.R.: *Secure object identification: or: solving the Chess Grandmaster Problem*. In: *Proceedings of the New Security Paradigms Workshop*. Seiten 77–85. Ascona, Schweiz, 2003, ACM Press. ISBN 1-58113-880

[7] ALPERN, B. und SCHNEIDER, F.B.: *Recognizing Safety and Liveness*. In: *Distributed Computing*, Vol. 2, Nr. 3, Seiten 117–126, 1987. ISSN 0178-2770

[8] ANDERSON, R.: *Security Engineering – A Guide to Building Dependable Distributed Systems*. John Wiley and Sons, 2001. ISBN 047138922

[9] ANDERSON, R., BERGADANO, F., CRISPO, B., LEE, J.-H., MANIFAVAS, C. und NEEDHAM, R.: *A new family of authentication protocols*. In: *ACM SIGOPS Operating Systems Review*, Vol. 32, Nr. 4, Seiten 9–20, 1998. ISSN 0163-5980

[10] ANDERSON, R., CHAN, H. und PERRIG, A.: *Key Infection: Smart Trust for Smart Dust*. In: *Proceedings of IEEE International Conference on Network Protocols*. Seiten 206–215. Berlin, 2004, IEEE Press. ISBN 0-7695-2161-4

[11] ATMEL CORPORATION: *ATMEL ATmega128 datasheet*. 2002.
http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[12] BAKER, F. und SAVOLA, P.: *Ingress Filtering for Multihomed Networks*. RFC 3704 (Best Current Practice), IETF, 2004.
http://www.ietf.org/rfc/rfc3704.txt

[13] BALFANZ, D., SMETTERS, D.K., STEWART, P. und WONG, H.: *Talking to Strangers: Authentication in Ad-Hoc Wireless Networks*. In: *Proceedings of Symposium on Network and Distributed Systems Security*. San Diego, USA, 2002, ISOC

[14] BARAK, B., CANETTI, R., LINDELL, Y., PASS, R. und RABIN, T.: *Secure Computation Without Authentication*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 361–377. Santa Barbara, USA, 2005, Springer Verlag. ISBN 3-540-28114-2

[15] BAUMGART, Ingmar: *Analyse, Optimierung und Evaluierung eines sicheren verteilten Dienstverzeichnisses für drahtlose Sensornetze*. Diplomarbeit, Institut für Telematik, Universität Karlsruhe (TH), 2005.
http://www.tm.uka.de

[16] BECHER, A., BENENSON, Z. und DORNSEIF, M.: *Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks*. In: *Proceedings of International Conference on Security in Pervasive Computing*. Seiten 104–118. York, UK, 2006, Springer Verlag. ISBN 3540333762

[17] BELLARE, M., CANETTI, R. und KRAWCZYK, H.: *Keying Hash Functions for Message Authentication*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 1–15. Santa Barbara, USA, 1996, Springer Verlag. ISBN 3-540-61512-1

[18] BELLARE, M. und NAMPREMPRE, C.: *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm*. In: *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*. Seiten 531–545. Kyoto, Japan, 2000, Springer Verlag. ISBN 3-540-41404-5

[19] BELLARE, M. und ROGAWAY, P.: *Random Oracles are practical: a Paradigm for designing Efficient Protocols*. In: *Proceedings of ACM Conference on Computer and Communications Security*. Seiten 62–73. Fairfax, USA, 1993, ACM Press. ISBN 0-89791-629-8

[20] BELLOVIN, S.M.: *Problem Areas for the IP Security Protocols*. In: *Proceedings of the Usenix UNIX Security Symposium*. Seiten 1–16. San Diego, USA, 1996, ISOC. Online verfügbar unter:
http://www.cs.columbia.edu/~smb/papers/badesp.pdf

[21] BENALOH, J.C. und MARE, M. de: *One-Way Accumulators: A Decentralized Alternative to Digital Signatures*. In: *Proceedings of IACR Advances in Cryptology, EUROCRYPT*. Seiten 274–285. Lofthus, Norwegen, 1993, Springer Verlag

[22] BENENSON, Z., CHOLEWINSKI, P.M. und FREILING, F.C.: *Simple Evasive Data Storage in Sensor Networks*. In: *Proceedings of International Workshop on Distributed Algorithms and Applications for Wireless and Mobile Systems*. Seiten 779–784. Phoenix, USA, 2005. ISBN 0-88986-525-6

[23] BETH, T., BORCHERDING, M. und KLEIN, B.: *Valuation of Trust in Open Networks*. In: *Proceedings of European Symposium on Research in Computer Security ESORICS*. Seiten 3–18. Brighton, UK, 1994, Springer Verlag. ISBN 3-540-58618-0

[24] BETH, T. und DESMEDT, Y.: *Identification Tokens – or: Solving the Chess Grandmaster Problem*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 169–177. Santa Barbara, USA, 1990, Springer Verlag. ISBN 3-540-54508-5

[25] BLASS, E.-O., CONRAD, M. und ZITTERBART, M.: *A Tree-Based Approach for Secure Key Distribution in Wireless Sensor Networks*. In: *Proceedings of REALWSN: Real World Wireless Sensor Networks*. Stockholm, Schweden, 2005

[26] BLASS, E.-O. und ZITTERBART, M.: *An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks*. In: *Proceedings of ACM Symposium on Information, Computer, and Communications Security*. Seiten 303–310. Taipeh, Taiwan, 2005, ACM Press. ISBN 1-59593-272-0

[27] BLASS, E.-O., HOF, H.-J., HURLER, B. und ZITTERBART, M.: *Erste Erfahrungen mit der Karlsruher Sensornetz-Plattform*. In: *Proceedings of GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*. Berlin, 2003

[28] BLASS, E.-O., HOF, H.-J. und ZITTERBART, M.: *S-CAN: Sicheres Overlay für Sensornetze*. In: *Proceedings of GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*. Karlsruhe, 2004

[29] BLASS, E.-O., JUNKER, H. und ZITTERBART, M.: *Effiziente Implementierung von Public-Key Algorithmen für Sensornetze*. In: *LNCS Lecture Notes in Informatics: Informatik 2005, Vol. 2*. Seiten 140–144, Springer Verlag, 2005. ISBN 3-88579-397-0

[30] BLASS, E.-O., TIEDE, L. und ZITTERBART, M.: *An Energy-Efficient and Reliable Mechanism for Data Transport in Wireless Sensor Networks*. In: *Proceedings of International Conference on Networked Sensing Systems*. Seiten 211–216. Chicago, USA, 2006, TRF. ISBN 0-9743611-3-5

[31] BLASS, E.-O., WILKE, J. und ZITTERBART, M.: *A Security–Energy Trade-Off for Authentic Aggregation in Sensor Networks*. In: *Proceedings of IEEE Communications Society Conference on Sensor and AdHoc Communications and Networks*. Seiten 135–137. Washington D.C., USA, 2006, IEEE Press. ISBN 1-4244-0732-X

[32] BLASS, E.-O. und ZITTERBART, M.: *Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks*. 2005. Technischer Bericht, Institut für Telematik, Universität Karlsruhe
http://doc.tm.uka.de/2005/tm-2005-1.pdf

[33] BLASS, E.-O. und ZITTERBART, M.: *Towards Acceptable Public-Key Encryption in Sensor Networks*. In: *Proceedings of ACM 2nd International Workshop on Ubiquitous Computing*. Seiten 88–93. Miami, USA, 2005, INSTICC Press. ISBN 972-8865-24-4

[34] BLESS, R., MINK, S., BLASS, E.-O., CONRAD, M., HOF, H.-J., KUTZNER, K. und SCHÖLLER, M.: *Sichere Netzwerkkommunikation*. Springer Verlag, 2005. ISBN 3-540-21845-9

[35] BLOM, R.: *An optimal class of symmetric key generation systems*. In: *Proceedings of IACR Advances in Cryptology, EUROCRYPT*. Seiten 335–338. Paris, Frankreich, 1985, Springer Verlag. ISBN 0-387-16076-0

[36] BLUNDO, C., SANTIS, A. D., HERZBERG, A., KUTTEN, S., VACCARO, U. und YUNG, M.: *Perfectly-Secure Key Distribution for Dynamic Conferences*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 471–486. Santa Barbara, USA, 1992, Springer Verlag. ISBN 3-540-57340-2

[37] BOEING, N.: *Big Doctor is watching you*. Die Zeit, Februar 2006. Ausgabe 6, http://www.zeit.de/

[38] BOHGE, M. und TRAPPE, W.: *An authentication framework for hierarchical ad hoc sensor networks*. In: *Proceedings of Workshop on Wireless Security*. Seiten 79–87. San Diego, USA, 2003, ACM Press. ISBN 1-58113-769-9

[39] BONEH, D. und FRANKLIN, M.: *Identity-Based Encryption from the Weil Pairing*. In: *Proceedings of International Cryptology Conference*. Seiten 219–229. Santa Barbara, 2001, Springer Verlag. ISBN 3-540-42456-3

[40] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Verteilung der Computer-Viren aus gemeldetem Aufkommen*. 2003. http://www.bsi.bund.de/av/virenstatistik/vaus.htm

[41] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK (BSI): *IT-Grundschutz-Kataloge, Gefährdungskatalog Vorsätzliche Handlungen*. 2005. http://www.bsi.de/gshb/deutsch/g/g05.htm

[42] BUNDESANSTALT FÜR GEOWISSENSCHAFTEN UND ROHSTOFFE (BGR): *Radionuklides Stationsnetz des IMS*. 2006. http://www.seismologie.bgr.de

[43] BURROWS, M., ABADI, M. und NEEDHAM, R.: *A Logic of Authentication*. In: *ACM Transactions on Computer Systems*, Vol. 8, Nr. 1, Seiten 18–36, 1990. ISSN 0734-2071

[44] BUTTYÁN, L., SCHAFFER, P. und VAJDA, I.: *RANBAR: RANSAC-Based Resilient Aggregation in Sensor Networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 83–90. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-554-1

[45] CA CERT: *Denial-of-Service Attack via ping*. Carnegie Mellon University, CERT Advisory CA-1996-26, 1996. http://www.cert.org/advisories/CA-1996-26.html

[46] CA CERT: *TCP SYN Flooding and IP Spoofing Attacks*. Carnegie Mellon University, CERT Advisory CA-1996-21, 2000.
http://www.cert.org/advisories/CA-1996-21.html

[47] CARMAN, D., KRUUS, P. und MATT, B.: *Constraints and Approaches for Distributed Sensor Network Security* / NAI Labs Technical Report. 2000. Forschungsbericht.
http://download.nai.com/products/media/nai/zip/nailabs-report-00-010-final.zip

[48] CARMAN, D.W., MATT, B.J. und CIRINCIONE, G.H.: *Energy-efficient and low-latency key management for sensor networks*. In: *Proceedings of 23rd Army Science Conference*. Orlando, USA, 2002, Abrufbar unter: Networks Associates Laboratories, Advanced Security Research Journal, Seiten 31–40.
http://www.isso.sparta.com/research/documents/asrjv5.pdf

[49] CASTELLUCCIA, C., MYKLETUN, E. und TSUDIK, G.: *Efficient Aggregation of Encrypted Data in Wireless Sensor Networks*. In: *Proceedings of ACM/IEEE Mobiquitous*. Seiten 109–117. San Diego, USA, 2005, IEEE Press. ISBN 0-7695-2375-7

[50] ÇAM, H., OZDEMIR, S., NAIR, P. und MUTHUAVINASHIAPPAN, D.: *ESPDA: energy-efficient and secure pattern-based data aggregation for wireless sensor networks*. In: *Proceedings of 2nd IEEE Sensors*. Seiten 732–736. Toronto, Kanada, 2003, IEEE Press. ISBN 0-7803-8133-5

[51] ÇAM, H., ÖZDEMIR, S., MUTHUAVINASHIAPPAN, D. und NAIR, P.: *Energy-Efficient security protocol for Wireless Sensor Networks*. In: *Proceedings of IEEE Vehicular Technology Conference*. Seiten 2981–2984. Orlando, USA, 2003, IEEE Press. ISBN 0-7803-7954-3

[52] ÇAM, H., ÖZDEMIR, S., SANLI, H.O. und NAIR, P.: *Secure Differential Data Aggregation for Wireless Sensor Networks*. In: *Sensor Network Operations*. IEEE Press, 2006. ISBN 0-471-71976-5

[53] ÇAMTEPE, S.A. und YENER, B.: *Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks*. In: *Proceedings of European Symposium on Research in Computer Security ESORICS*. Seiten 293–308. Sophia Antipolis, Frankreich, 2004, Springer Verlag

[54] ÇAMTEPE, S.A. und YENER, B.: *Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks*. In: *ACM Transactions on Networking*, Vol. 15, Nr. 2, Seiten 346–358, 2007. ISSN 1063-6692

[55] CERTICOM: *Certicom Announces Elliptic Curve Cryptosystem (ECC) Challenge Winner*. 2002.
http://www.certicom.com/index.php?action=company,press_archive&view=121

[56] CHAN, H., GLIGOR, V.D., PERRIG, A. und MURALIDHARAN, G.: *On the Distribution and Revocation of Cryptographic Keys in Sensor Networks*. In: *IEEE Transactions on Dependable and Secure Computing*, Vol. 2, Nr. 3, Seiten 233–247, 2005. ISSN 1545-5971

[57] CHAN, H. und PERRIG, A.: *PIKE: Peer Intermediaries for Key Establishment in Sensor Networks*. In: *Proceedings of IEEE INFOCOM*. Seiten 524–535. Miami, USA, 2005, IEEE Press. ISBN 0-7803-8968-9

[58] CHAN, H., PERRIG, A. und SONG, D.: *Random Key Predistribution Schemes for Sensor Networks*. In: *Proceedings of IEEE Symposium on Security and Privacy*. Seiten 197–215. Berkeley, USA, 2003, IEEE Press. ISBN 0-7695-1940-7

[59] CHAN, H., PERRIG, A. und SONG, D.: *Secure Hierarchical In-Network Aggregation in Sensor Networks*. In: *Proceedings of Conference on Computer and Communications Security*. Seiten 278–287. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-518

[60] CHANDRAMOULI, R., BAPATLA, S., SUBBALAKSHMI, K.P. und UMA, R.N.: *Battery power-aware encryption*. In: *ACM Transactions on Information and System Security*, Vol. 9, Nr. 2, Seiten 162–180, 2006. ISSN 1094-9224

[61] CHORZEMPA, M., PARK, J. und ELTOWEISSY, M.: *SECK: Survivable and efficient clustered keying for wireless sensor networks*. In: *Proceedings of IEEE Workshop on Information Assurance in Wireless Sensor Networks*. Seiten 453–458. Phoenix, USA, 2005, IEEE Press. ISBN 0-7803-8991-3

[62] CHORZEMPA, M., PARK, J.-M., ELTOWEISSY, M. und HOU, T.: *Key Management for Wireless Sensor Networks in Hostile Environments*. In: *Security in Sensor Networks*. Auerbach, 2006. ISBN 0849370582

[63] CLAUSEN, T., JACQUET, P., LAOUITI, A., MUHLETHALER, P., QAYYUM, A. und VIENNOT, L.: *Optimized Link State Routing Protocol*. In: *Proceedings of IEEE INMIC*. Seiten 62–68. Lahore, Pakistan, 2001, IEEE Press. ISBN 0-7803-7406-1

[64] COVINGTON, M.J., AHMAD, M., ESSA, I. und VENKATESWARAN, H.: *Parameterized Authentication*. In: *Proceedings of European Symposium on Research in Computer Security ESORICS*. Seiten 276–292. Valbonne, Frankreich, 2004, Springer Verlag. ISBN 3-540-22987-6

[65] CRAMER, R. und DAMGÅRD, I.: *Introduction to Secure Multi-Party Computations*. In: *Contemporary Cryptology: Advanced Courses in Mathematics*. Birkhauser, 2005, Seiten 41–87. ISBN 3-7643-7294-X

[66] CROSSBOW INC.: *Motes, Smart Dust Sensors, Wireless Sensor Networks*. 2005. http://www.xbow.com/Products/productsdetails.aspx?sid=3

[67] CROSSBOW INC.: *MICA2 – Wireless Measurement System*. 2006. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-04_A_MICA2.pdf

[68] CROSSBOW INC.: *MPR410CB MICA2, 900/433MHz, Price*. 2006. http://www.xbow.com/Products/productdetails.aspx?sid=174

[69] CROSSBOW INC.: *RADIO, RF Concepts, and TOS Radio Stack*. 2006. http://xbow.com/Support/Support_pdf_files/Motetraining/Wireless.pdf und http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/PresentationOverheads/Day1_Sect06_RFConcepts.pdf

[70] DAVIS, D. und SWICK, R.: *Network Security via Private-Key Certificates*. In: *ACM Operating System Review*, Vol. 24, Nr. 4, Seiten 64–67, 1990. ISSN 0163-5980

[71] DENG, J., HAN, R. und MISHRA, S.: *Security support for in-network processing in Wireless Sensor Networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 83–93. Fairfax, USA, 2003, ACM Press. ISBN 1-58113-783-4

[72] DENG, J., HAND, R. und MISHRA, S.: *Defending against path-based DoS attacks in wireless sensor networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 89–96. Alexandria, USA, 2005, ACM Press. ISBN

[73] DI PIETRO, R., MANCINI, L.V., MEI, A. und PANCONESI, A.: *How to Design Connected Sensor Networks that Are Provably Secure*. In: *Proceedings of International Conference on Security and Privacy in Communication Networks*. Baltimore, USA, 2006.
http://www.securecomm.org/2006/

[74] DI PIETRO, R., MANCINI, L.V., MEI, A., PANCONESI, A. und RADHAKRISHNAN, J.: *Connectivity properties of secure wireless sensor networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 53–58. Washington D.C., USA, 2004, ACM Press. ISBN 1-58113-972-1

[75] DIERKS, T. und RESCORLA, E.: *The Transport Layer Security (TLS) Protocol Version 1.1*. RFC 4346 (Proposed Standard), IETF, 2006.
http://www.ietf.org/rfc/rfc4346.txt

[76] DIFFIE, W. und HELLMAN, M.: *New Directions in Cryptography*. In: *IEEE Transactions on Information Theory*, Vol. 22, Nr. 6, Seiten 644–654, 1976. ISSN 0018-9448

[77] DIMITRIOU, T. und FOTEINAKIS, D.: *Secure and Efficient In-Network Processing for Sensor Networks*. In: *Proceedings of Workshop on Broadband Advanced Sensor Networks*. San Jose, USA, 2004

[78] DISTRIBUTED.NET: *Projekt RC5*. 2006.
http://distributed.net/rc5/

[79] DOLEV, D. und YAO, A.: *On the security of public key protocols*. In: *IEEE Transactions on Information Theory*, Vol. 29, Nr. 2, Seiten 198–208, 1983. ISSN: 0018-9448

[80] DOMINGO-FERRER, J.: *A provably secure additive and multiplicative privacy homomorphism*. In: *Proceedings of Information Security Conference ISC*. Seiten 471–483. Sao Paulo, Brasilien, 2002, Springer Verlag. ISBN 3540442707

[81] DU, W., DENG, J., HAN, Y. S., CHEN, S. und VARSHNEY, P. K.: *A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge*. In: *Proceedings of IEEE INFOCOM*. Seiten 586–597. Hong Kong, China, 2004, IEEE Press. ISBN 0-7803-8355-9

[82] DU, W., DENG, J., HAN, Y.S. und VARSHNEY, P.: *A Witness-Based Approach For Data Fusion Assurance In Wireless Sensor Networks*. In: *Proceedings of ACM Global Communications Conference*. Seiten 1435–1439. San Francisco, USA, 2003, ACM Press. ISBN 0-7803-7974-8

[83] DU, W., DENG, J., HAN, Y.S. und VARSHNEY, P.K.: *A Pairwise Key Predistribution Scheme for Wireless Sensor Networks*. In: *Proceedings of ACM Conference on Computer and Communications security*. Seiten 42–51. Washington D.C., USA, 2003, ACM Press. ISBN 1-58113-738-9

[84] DU, W., WANG, R. und NING, P.: *An efficient scheme for authenticating public keys in sensor networks*. In: *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing*. Seiten 58–67. Urbana-Champaign, USA, 2005, ACM Press. ISBN 1-59593-004-3

[85] DUTERTRE, B., CHEUNG, S. und LEVY, J.: *Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust* / Computer Science Laboratory SRI International. 2004. Forschungsbericht.
http://www.csl.sri.com/users/cheung/

[86] ELTOWEISSY, M., HEYDARI, H., MORALES, L. und SUDBOROUGH, H.: *Combinatorial optimizations of group key management*. In: *Journal of Networks and Systems Management*, Vol. 12, Nr. 1, Seiten 33–50, 2004. ISSN 1064-7570

[87] ENGELBRECHT, N. und PENZHORN, W.T.: *Secure Authentication Protocols Used for Low Power Wireless Sensor Networks*. In: *Proceedings of the International Symposium on Industrial Electronics*. Seiten 1777–1782. Dubrovnik, Kroatien, 2005, IEEE Press. ISBN 0-7803-8738-4

[88] ESCHENAUER, L. und GLIGOR, V.: *A Key Management Scheme for Distributed Sensor Networks*. In: *Proceedings of ACM Computer and Communications Security*. Seiten 41–47. Washington D.C. USA, 2002, ACM Press. ISBN 1-58113-612-9

[89] ESCHENAUER, L. und GLIGOR, V. D.: *Method and apparatus for key management in distributed sensor networks*. 2006.
http://www.freshpatents.com/, Patent No. 20050140964

[90] FERGUSON, P. und SENIE, D.: *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2267 (Informational), IETF, 1998.
http://www.ietf.org/rfc/rfc2267.txt

[91] FERGUSON, P. und SENIE, D.: *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2827 (Best Current Practice), IETF, 2000.
http://www.ietf.org/rfc/rfc2827.txt

[92] FIAT, A. und SHAMIR, A.: *How to prove yourself: practical solutions to identification and signature problems*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 186–194. Santa Barbara, USA, 1987, Springer Verlag. ISBN 0-387-18047-8

[93] FISCHER, M.A. und BOLLES, R.C.: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. In: *Communications of the ACM*, Vol. 24, Nr. 6, Seiten 381–395, 1981. ISSN 0001-0782

[94] GHOSH, S.K.: *On Optimality of Key Pre-distribution Schemes for Distributed Sensor Networks*. In: *Proceedings of Workshop on Security and Privacy in Ad hoc and Sensor Networks*. Hamburg, 2006

[95] GIRAO, J., WESTHOFF, D. und SCHNEIDER, M.: *CDA: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks*. In: *Proceedings of IEEE International Conference on Communications*. Seiten 3044–3049. Seoul, Korea, 2005, IEEE Press. ISBN 0-7803-8938-7

[96] GLIGOR, V.D. und DONESCU, P.: *Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes*. In: *Proceedings of International Workshop on Fast Software Encryption*. Seiten 92–108. Yokohama, Japan, 2001, Springer Verlag. ISBN 3-540-43869-6

[97] GÜNTHER, C.G.: *An Identity-Based Key-Exchange Protocol*. In: *Proceedings of Workshop on the Theory and Application of of Cryptographic Techniques*. Seiten 29–37. Houthalen, Belgium, 1989, Springer Verlag. ISBN 3-540-53433-4

[98] GUPTA, R. und DAS, S. R.: *Tracking Moving Targets in a Smart Sensor Network*. In: *Proceedings of Vehicular Technology Conference*. Seiten 3035–3039. Orlando, USA, 2003, IEEE Press. ISBN 0-7803-7954-3

[99] GURA, N., PATEL, A., WANDER, A., EBERLE, H. und SHANTZ, S.C.: *Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs*. In: *Proceedings of Cryptographic Hardware and Embedded Systems*. Seiten 119–132. Cambridge, USA, 2004, Springer Verlag. ISBN 3-540-22666-4

[100] HADIM, S. und MOHAMED, N.: *Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks*. In: *IEEE Distributed Systems Online*, Vol. 7, Nr. 3. ISSN 1541-4922

[101] HEINZELMAN, W.R., CHANDRAKASAN, A. und BALAKRISHNAN, H.: *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*. In: *Proceedings of Hawaii International Conference on System Sciences*. Seiten 3005–3014. Hawaii, USA, 2000, IEEE Computer Society. ISBN 0-7695-0493-0

[102] HOF, H. und ZITTERBART, M.: *S-CAN: A secure service directory for service-centric wireless sensor networks*. In: *Computer Communications*, Vol. 28, Nr. 13, Seiten 1517–1522, 2005. ISSN 0140-3664

[103] HOF, H.-J., BLASS, E.-O., FUHRMANN, T. und ZITTERBART, M.: *Design of a Secure Distributed Service Directory for Wireless Sensornetworks*. In: *Proceedings of 1st European Workshop on Wireless Sensor Networks*. Seiten 276–290. Berlin, 2004, Springer Verlag. ISBN 3-540-20825-9

[104] HOF, H.-J., BLASS, E.-O. und ZITTERBART, M.: *Secure Overlay for Service Centric Wireless Sensor Networks*. In: *Proceedings of 1st European Workshop on Security in Ad-hoc and Sensor Networks*. Seiten 125–138. Heidelberg, 2004, Springer Verlag. ISBN 3-540-24396-8

[105] HOFFSTEIN, J., PIPHER, J. und SILVERMAN, J.H.: *NTRU: A Ring-Based Public Key Cryptosystem*. In: *Proceedings of the Third International Symposium on Algorithmic Number Theory*. Seiten 267–288. Portland, USA, 1998, Springer Verlag. ISBN 3-540-64657-4

[106] HOWARD, J. und LONGSTAFF, T.: *A Common Language for Computer Security Incidents* / Carnegie Mellon University. 1998. Forschungsbericht.
http://www.cert.org/research/

[107] HU, L. und EVANS, D.: *Secure Aggregation for Wireless Networks*. In: *Proceedings of IEEE Symposium on Applications and the Internet Workshops (SAINT)*. Seiten 384–391. Orlando, USA, 2003, IEEE Press. ISBN 0-7695-1873-7

[108] HUAND, Q., CUKIER, J.I., KOBAYASHI, H., LIU, B. und ZHANG, J.: *Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks*. In: *Proceedings of ACM Wireless Sensor Networks and Applications*. Seiten 141–150. San Diego, USA, 2003, ACM Press. ISBN 1-58113-764-8

[109] HUANG, D., MEHTA, M., MEDHI, D. und HARN, L.: *Location-aware key management scheme for wireless sensor networks*. In: *Proceedings of ACM Workshop on Security of ad hoc and Sensor Networks*. Seiten 29–42. Washington D.C., USA, 2004, ACM Press. ISBN 1-58113-972-1

[110] HUANG, Q., KOBAYASHI, H:, LIU, B., GU, D. und ZHANG, J.: *Energy/Security Scalable Mobile Cryptosystem*. In: *Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. Seiten 2755–2759. Beijing, China, 2003, IEEE Press. ISBN 0-7803-7822-9

[111] HUTTEL, A.: *Vergleich und Bewertung von Verfahren zum Schlüsselaustausch in Sensornetzen*. Studienarbeit, Institut für Telematik, Universität Karlsruhe (TH), 2006.
http://www.tm.uka.de

[112] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: *IEEE 802.11i, 2nd Edition, Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. 2005. ISO/IEC 8802-11
urlhttp://standards.ieee.org/reading/ieee/std/lanman/restricted/8802-11-2005.pdf

[113] INTANAGONWIWAT, C., GOVINDAN, R. und ESTRIN, D.: *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*. In: *Proceedings of IEEE/ACM MobiCom*. Seiten 56–67. Boston, USA, 2000, IEEE Press. ISBN 1-58113-197-6

[114] INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D., HEIDEMANN, J. und SILVA, F.: *Directed diffusion for Wireless Sensor Networking*. In: *ACM Transactions on Networking*, Vol. 11, Nr. 1, Seiten 2–16, 2003. ISSN 1063-669

[115] INTERNATIONAL TELECOMMUNICATION UNION – TELECOMMUNICATION STANDARDIZATION SECTOR (ITU-T) / INTERNATIONAL ORGANISATION FOR STANDARDIZATION (Hrsg.): *X.509: Information technology - Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*. ITU: International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) / International Organisation for Standardization, März 2003. http://www.itu.int/ITU-T/e-business/mou/MoUMG-standards.html

[116] ISO/IEC 13888-2:1998(E): *Information technology – Security techniques – Non-repudiation – Part 2: Mechanisms using symmetric techniques*. 1998. International Organization for Standardization

[117] ITO, T., OHTA, H., MATSUDA, N. und YONEDA, T.: *A key pre-distribution scheme for secure sensor networks using probability density function of node deployment*. In: *Proceedings of ACM Workshop on Security of ad hoc and Sensor Networks*. Seiten 69–75. Alexandria, USA, 2005, ACM Press. ISBN 1-59593-227-5

[118] JADIA, P. und MATHURIA, A.: *Efficient Secure Aggregation in Sensor Networks*. In: *Proceedings of High Performance Computing HiPC*. Seiten 40–49. Bangalore, Indien, 2004, Springer Verlag. ISBN 3-540-24129-9

[119] JUNKER, H.: *Analyse und Realisierung effizienter Sicherheitsmechanismen für Sensornetze*. 2005. Diplomarbeit, Institut für Telematik, Universität Karlsruhe (TH),
http://www.tm.uka.de

[120] JUTLA, C.S.: *Encryption modes with almost free message integrity*. In: *Proceedings of IACR Advances in Cryptology, EUROCRYPT*. Seiten 529–544. Innsbruck, Österreich, 2001, Springer Verlag. ISBN 3-540-42070-3

[121] KAMVAR, S. D., SCHLOSSER, M.T. und GARCIA-MOLINA, H.: *The EigenTrust Algorithm for Reputation Management in P2P Networks*. In: *Proceedings of ACM International World Wide Web Conference*. Seiten 640–651. Budapest, Ungarn, 2003, ACM Press. ISBN 1-58113-680-3

[122] KARLOF, C. und WAGNER, D.: *Secure routing in wireless sensor networks: attacks and countermeasures*. In: *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications*. Seiten 113–127. Anchorage, Alaska, 2003, IEEE Press. ISBN 0-7803-7879-2

[123] KARLOFF, C., SAASTRY, N. und WAGNER, D.: *Tinysec: A link layer security architecture for wireless sensor networks*. In: *Proceedings of ACM Conference on Embedded Networked Sensor Systems*. Seiten 162–175. Baltimore, USA, 2004, ACM Press. ISBN 1-58113-879-2
http://www.cs.berkeley.edu/~nks/tinysec

[124] KELSEY, J., SCHNEIER, B. und FERGUSON, N.: *Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator*. In: *Proceedings of Selected Areas in Cryptography*. Seiten 13–33. Kingston, Kanada, 1999, Springer Verlag. ISBN 3-540-67185-4

[125] KELSEY, J., SCHNEIER, B., WAGNER, D. und HALL, C.: *Cryptanalytic Attacks on Pseudorandom Number Generators*. In: *Proceedings of International Workshop on Fast Software Encryption*. Seiten 168–188. Paris, Frankreich, 1998, Springer Verlag. ISBN 3-540-64265-X

[126] KENT, S.: *IP Encapsulating Security Payload (ESP)*. RFC 4303 (Standards Track), IETF, 2005.
http://www.ietf.org/rfc/rfc4303.txt

[127] KENT, S. und SEO, K.: *Security Architecture for the Internet Protocol*. RFC 4301 (Proposed Standard), IETF, 2005.
http://www.ietf.org/rfc/rfc4301.txt

[128] KERCKHOFFS, A.: *La cryptographie militaire II*. In: *Journal des sciences militaires*, Vol. II, Seiten 161–191, Feb. 1883

[129] KERCKHOFFS, A.: *La cryptographie militaire I*. In: *Journal des sciences militaires*, Vol. I, Seiten 5–38, Jan. 1883. Online verfügbar unter:
http://www.petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf

[130] KHALIL, I., BAGCHI, S. und SHROFF, N.B.: *Analysis and Evaluation of SECOS, a Protocol for Energy Efficient and Secure Communication in Sensor Networks*. In: *Ad-Hoc Networks Journal*, Vol. .
http://cobweb.ecn.purdue.edu/~dcsl/Publications/

[131] KOBLITZ, N.: *A Course in Number Theory and Cryptography*. Springer Verlag, 1994. ISBN 0387942939

[132] KOTZANIKOLAOU, P., MAGKOS, E., DOULIGERIS, C. und CHRISSIKOPOULOS, V.: *Hybrid Key Establishment for Multiphase Self-Organized Sensor Networks*. In: *Proceedings of IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing*. Seiten 581–587. Taormina, Italien, 2005, IEEE Press. ISBN 0-7695-2342-0-03

[133] KRISHANAMACHARI, B., ESTRIN, D. und WICKER, S.: *The impact of data aggregation in wireless sensor networks*. In: *Proceedings of IEEE Distributed Event-Based Systems*. Seiten 575–578. Wien, Österreich, 2002, IEEE Press. ISBN 0-7695-1588-6

[134] KUMAR, S., GIRIMONDO, M., WEIMERSKIRCH, A., PAAR, C., PATEL, A. und WANDER, A.S.: *Embedded end-to-end wireless security with ECDH key exchange*. In: *Proceedings of IEEE Midwest Symposium on Circuits and Systems*. Seiten 786–789. Kairo, Ägypten, 2003, IEEE Press. ISBN 0-7803-8294-3

[135] LAI, B.-C.C., HWANG, D.D., KIM, S.P. und VERBAUWHEDE, I.: *Reducing radio energy consumption of key management protocols for wireless sensor networks*. In: *Proceedings of IEEE International Symposium on Low Power Electronics and Design*. Seiten 351–356. Newport Beach, USA, 2004, IEEE Press. ISBN 1-58113-929-2

[136] LAMPORT, L., SHOSTAK, R. und PEASE, M.: *The Byzantine Generals Problem*. In: *ACM Transactions on Programming Languages and Systems*, Vol. 4, Nr. 3, Seiten 382–401, 1982. ISSN 0164-0925

[137] LAUTER, K.: *The Advantages of Elliptic Curve Cryptography for Wireless Security*. In: *IEEE Wireless Communications*, Vol. 11, Nr. 1, Seiten 62–67, 2004. ISSN 1536-1284

[138] LAW, Y., CORIN, R., ETALLE, S. und HARTEL, P.: *A Formally Verified Decentralized Key Management Architecture for Wireless Sensor Networks*. In: *Proceedings of Personal Wireless Communication*. Seiten 27–39. Venedig, Italien, 2003

[139] LAW, Y., ETALLE, S. und HARTEL, P.: *Key Management with Group-Wise Pre-Deployed Keying and Secret Sharing Pre-Deployed Keying* / University of Twente NL. 2002. Forschungsbericht.
http://doc.utwente.nl/38387

[140] LEE, J. und STINSON, D.R.: *Deterministic Key Predistribution Schemes for Distributed Sensor Networks*. In: *Proceedings of Workshop on Selected Areas in Cryptography*. Seiten 294–307. Waterloo, Kanada, 2004, Springer Verlag. ISBN 3-540-24327-5

[141] LEE, J. und STINSON, D.R.: *A Combinatorial Approach to Key Predistribution for Distributed Sensor Networks*. In: *Proceedings of IEEE Wireless Communications and Networking Conference*. Seiten 1200–1205. New Orleans, USA, 2005, IEEE Press. ISBN 0-7803-8966-2

[142] LENSTRA, A.K.: *Unbelievable Security. Matching AES Security Using Public Key Systems*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 67–86. Santa Barbara, USA, 2001, Springer Verlag. ISBN 3-540-42987-5

[143] LENSTRA, A.K. und VERHEUL, E.R.: *Selecting Cryptographic Key Sizes*. In: *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography*. Seiten 446–465. Melbourne, Australien, 2000, Springer Verlag. ISBN 3-540-66967-1

[144] LENSTRA, A.K. und VERHEUL, E.R.: *The XTR public key system*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 1–19. Santa Barbara, USA, 2000. ISBN 3-540-67907-3

[145] LIU, D. und NING, P.: *Establishing Pairwise Keys in Distributed Sensor Networks*. In: *Proceedings of ACM Computer and Communications Security*. Seiten 52–61. Washington D.C., USA, 2003, ACM Press. ISBN 1-58113-738-9

[146] LIU, D. und NING, P.: *Location-Based Pairwise Key Establishment for Static Sensor Networks*. In: *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 72–82. Fairfax, USA, 2003, ACM Press. ISBN 1-58113-783-4

[147] LIU, F. und CHENG, X.: *A Self-Configured Key Establishment Scheme for Large-Scale Sensor Networks*. In: *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems*. Seiten 447–456. Vancouver, Kanada, 2006, IEEE Press.
http://home.gwu.edu/~fliu/papers/SBK_MASS.pdf

[148] MADDEN, S., FRANKLIN, M.J., HELLERSTEIN, J.M. und HONG, W.: *TAG: a Tiny AGgregation service for ad-hoc sensor networks*. In: *ACM SIGOPS Operating Systems Review*, Vol. 36, Nr. SI, Seiten 131–146, 2002. ISSN 0163-5980

[149] MADDEN, S.R., FRANKLIN, M.J. und HELLERSTEIN, J.M.: *TinyDB: an acquisitional query processing system for sensor networks*. In: *ACM Transactions on Database Systems*, Vol. 30, Nr. 1, Seiten 122–173, 2005. ISSN 0362-5915

[150] MAGKOS, E., STEFANIDAKIS, M., KOTZANIKOLAOU, P. und VERGADOS, D.D.: *An Asymmetric Key Establishment Protocol for Multiphase Self-Organized Sensor Networks*. In: *Proceedings of European Wireless Conference EW2006*. Athen, Griechenland, 2006

[151] MAINWARING, A., CULLER, D., POLASTRE, J., SZEWCZYK, R. und ANDERSON, J.: *Wireless sensor networks for habitat monitoring*. In: *Proceedings of International Workshop on Wireless Sensor Networks and Applications*. Seiten 88–97. Atlanta, USA, 2002, ACM Press. ISBN 1-58113-589-0

[152] MALAN, D.J., WELSH, M. und SMITH, M.D.: *A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography*. In: *Proceedings of IEEE International Conference on Sensor and Ad Hoc Communications and Networks*. Seiten 71–80. Santa Clara, USA, 2004, IEEE Press. ISBN 0-7803-8796-1

[153] Kapitel: *Die technische Basis für das Internet der Dinge*. In: MATTERN, F.: *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer Verlag, 2005, Seiten 39–66. ISBN 3-540-24003-9

[154] MCCULLAGH, A. und CAELLI, W.: *Non-Repudiation in the Digital Environment*. In: *First Monday – Peer reviewed Journal on the Internet*, Vol. 5, Nr. 8. http://www.firstmonday.org/issues/issue5_8/mccullagh/index.html

[155] MENEZES, A.J.: *Elliptic Curve Public Key Cryptosystems*. Springer Verlag, 1993. ISBN 0792393686

[156] MENEZES, A.J., OORSCHOT, P.C. van und VANSTONE, S.A.: *Handbook of Applied Cryptography*. CRC Press, 2001. ISBN 0-8493-8523-7

[157] MERKLE, R.: *Secrecy, authentication, and public key systems*. 1979. Dissertation, Universität Stanford, USA http://www.merkle.com/papers/Thesis1979.pdf

[158] MILLER, S.P., NEUMAN, B. C., SCHILLER, J. I. und SALTZER, J.H.: *Kerberos Authentication and Authorization System* / Project Athena Technical Plan, MIT Project Athena. 1988. Forschungsbericht. http://web.mit.edu/Kerberos/papers.html

[159] MOHARRUM, M.A. und ELTOWEISSY, M.: *A Study of Static versus Dynamic Keying Schemes in Sensor Networks*. In: *Proceedings of Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*. Seiten 122–129. Montreal, Kanada, 2005, ACM Press. ISBN 1-59593-182-1

[160] MOORE, G.E.: *Cramming more components onto integrated circuits*. In: *Electronics*, Vol. 38, Nr. 9. Online verfügbar unter: http://download.intel.com/research/silicon/moorespaper.pdf

[161] MOORE, T.: *A Collusion Attack on Pairwise Key Predistribution Schemes for Distributed Sensor Networks*. In: *Proceedings of IEEE International Workshop on Pervasive Computing and Communication Security*. Seiten 251–255. Pisa, Italien, 2006, IEEE Press. ISBN 0-7695-2520-2

[162] MYKLETUN, E., GIRAO, J., GIRAO, J. und WESTHOFF, D.: *Public Key based Homomorphic Cryptoschemes for Data Concealment in Wireless Sensor Network*. In: *Proceedings of IEEE International Conference on Communications ICC*. Istanbul, Türkei, 2006, IEEE Press. ISBN 1424403545

[163] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *FIPS-197 – Advanced Encryption Standard*. 2001.
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[164] NETWORK ASSOCIATES, INC: *How PGP works*. 1999.
http://www.pgpi.org/doc/pgpintro/

[165] NICHOLSON, A.J., CORNER, M.D. und NOBLE, B.D.: *Mobile Device Security Using Transient Authentication*. In: *IEEE Transactions on Mobile Computing*, Vol. 5, Nr. 11, Seiten 1489–1502, 2006. ISSN 1536-1233

[166] NIST: *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised), NIST Special Publication 800-90*. 2007.
http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March 2007.pdf

[167] ÖNEN, M. und MOLVA, R.: *Secure Data Aggregation With Multiple Encryption*. In: *Proceedings of European Conference on Wireless Sensor Networks*. Seiten 117–132. Delft, Niederlande, 2007, Springer Verlag. ISBN 3540698299

[168] ORMAN, H. und HOFFMAN, P.: *Determing strengths for public keys used for exchanging symmetric keys*. RFC 3766 (Best Current Practice), IETF, 2004.
http://www.ietf.org/rfc/rfc3766.txt

[169] PATERSON, K.G. und LAU, A.K.L.: *Cryptography in Theory and Practice: The Case of Encryption in IPsec*. In: *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*. Seiten 12–29. St. Petersburg, Russland, 2006, Springer Verlag. ISBN 3-540-34546-9

[170] PATERSON, K.G. und YAU, A.K.L: *Lost in Translation: Theory and Practice in Cryptography*. In: *IEEE Security and Privacy*, Vol. 4, Nr. 3, Seiten 69–72, 2006. ISSN 1540-7993

[171] PERKINS, C.E. und ROYER, E.M.: *Ad hoc On-Demand Distance Vector Routing*. In: *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*. Seiten 90–100. New Orleans, USA, 1999, IEEE Press. ISBN 0-7695-0025-0

[172] PERRIG, A., CANETTI, R., SONG, D. und TYGAR, J.: *Efficient and Secure Source Authentication for Multicast*. In: *Proceedings of IEEE Network and Distributed System Security Symposium*. Seiten 35–46. San Diego, USA, 2001, Internet Society. ISBN 1-891562-10-X

[173] PERRIG, A., SZEWCZYK, R., TYGAR, J.D., WEN, V. und CULLER, D.E.: *SPINS: Security Protocols for Sensor Networks*. In: *Kluwer Wireless Networks*, Vol. 8, Nr. 5, Seiten 521–534, 2002. ISSN 1022-0038

[174] PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D. und TYGAR, J.: *SPINS: Security Protocols for Sensor Networks*. In: *Proceedings of ACM Mobile Computing and Networking*. Seiten 189–199. Rom, Italien, 2001, ACM Press. ISBN 1-58113-422-3

[175] PERRING, A., STANKOVIC, J. und WAGNER, D.: *Security in Wireless Sensor Networks*. In: *Communications of the ACM*, Vol. 47, Nr. 6, Seiten 53–57, 2004. ISSN 0001-0782

[176] PETER, S., PIOTROWSKI, K. und LANGENDOERFER, P.: *On Concealed Data Aggregation for Wireless Sensor Networks, Consumer Communications and Networking Conference*. In: *Proceedings of IEEE Consumer, Communications, and Networking Conference*. Las Vegas, USA, 2007, IEEE Press

[177] PFITZMANN, B.: *Digital Signature Schemes: General Framework and Fail-Stop Signatures*. Springer Verlag, 1996. ISBN 3540615172

[178] PIOTROWSKI, K., LANGENDOERFER, P. und PETER, S.: *How public key cryptography influences wireless sensor node lifetime*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 169–176. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-554-1

[179] POTTIE, G.J. und KAISER, W.J.: *Wireless Integrated Network Sensors*. In: *Communications of the ACM*, Vol. 43, Nr. 5, Seiten 51–58, 2000. ISSN 0001-0782

[180] PRZYDATEK, B., SONG, D. und PERRIG, A.: *SIA: secure information aggregation in sensor networks*. In: *Proceedings of ACM Conference on Embedded networked sensor systems SenSys*. Seiten 255–265. Los Angeles, USA, 2003, ACM Press. ISBN 1-58113-707-9

[181] RABIN, M.O.: *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. 1979. Technischer Bericht, TR-212, Massachusetts Institute of Technology, Cambridge, USA

[182] RAINA, M., GHOSH, S., PATRO, R. K., GANAPATHY, V. und THEJASWI, C.: *Secure Data Aggregation using commitment schemes and quasi commutative functions*. In: *Proceedings of IEEE International Symposium on Wireless Pervasive Computing*. Phuket, Thailand, 2006, IEEE Press. ISBN 0-7803-9410-0

[183] RAMACHANDRAN, U., KUMAR, R., WOLENETZ, M., COOPER, B., AGARWALLA, B., SHIN, J., HUTTO, P. und PAUL, A.: *Dynamic Data Fusion for Future Sensor Networks*. In: *ACM Transactions on Sensor Networks*, Vol. 2, Nr. 3, Seiten 404–443, 2006. ISSN 1550-4859

[184] REITER, M.K. und STUBBLEBINE, S.G.: *Authentication metric analysis and design*. In: *ACM Transactions on Information and System Security*, Vol. 2, Nr. 2, Seiten 138–158, 1999. ISSN 1094-9224

[185] RESEARCH GROUP FOR DISTRIBUTED SYSTEMS, ETH ZÜRICH: *BTnodes – A Distributed Environment for Prototyping Ad Hoc Networks*. 2005. http://www.btnode.ethz.ch/

[186] REYNOLDS, J.K.: *Helminthiasis of the Internet*. RFC 1135 (Informational), IETF, 1989. http://www.ietf.org/rfc/rfc1135.txt

[187] RIVEST, R.: *The RC5 Encryption Algorithm*. In: *Proceedings of Workshop on Fast Software Encryption*. Seiten 86–96. Leuven, Belgien, 1995, Springer Verlag

[188] RIVEST, R., SHAMIR, A. und ADELMAN, L.M.: *A method for obtaining digital signatures and pulic-key cryptosystems*. In: *Communications of the ACM*, Vol. 21, Nr. 2, Seiten 120–126, 1978

[189] RIVEST, R.L., ADLEMAN, L. und DERTOUZOS, M.L.: *On data banks and privacy homomorphisms*. In: *Foundations of Secure Computation*. Academic Press, 1978, Seiten 169–179. ISBN 0122103505

[190] ROGAWAY, P., BELLARE, M. und BLACK, J.: *OCB: A block-cipher mode of operation for efficient authenticated encryption*. In: *ACM Transactions on Information and System Security*, Vol. 6, Nr. 3, Seiten 365–403, 2003. ISSN 1094-9224

[191] RÖMER, K. und MATTERN, F.: *The Design Space of Wireless Sensor Networks*. In: *IEEE Wireless Communications*, Vol. 11, Nr. 6, Seiten 54–61, 2004. ISSN 1536-1284

[192] ROTHE, J.: *Some facets of complexity theory and cryptography: A five-lecture tutorial*. In: *ACM Computing Surveys*, Vol. 34, Nr. 4, Seiten 504–549, 2002. ISSN 0360-0300

[193] ROUNDY, S., STEINGART, D., FRECHETTE, L., WRIGHT, P.K. und RABAEY, J.M.: *Power Sources for Wireless Sensor Networks*. In: *Proceedings of European Workshop on Wireless Sensor Networks*. Seiten 1–17. Berlin, 2004, Springer Verlag. ISBN 3-540-20825-9

[194] ROY, S., SETIA, S. und JAJODIA, S.: *Attack-Resilient Hierarchical Data Aggregation in Sensor Networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 71–72. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-554-1

[195] RSA LABORATORIES: *The RSA Laboratories Secret-Key Challenge*. 2004. http://www.rsasecurity.com/rsalabs/node.asp?id=2100

[196] RSA LABORATORIES: *RSA-640 is factored!* 2005. http://www.rsasecurity.com/rsalabs/node.asp?id=2964

[197] RSA LABORATORIES: *The RSA Factoring Challenge*. 2005. http://www.rsasecurity.com/rsalabs/node.asp?id=2092

[198] SANCHEZ, D.S. und BALDUS, H.: *A Deterministic Pairwise Key Pre-distribution Scheme for Mobile Sensor Networks*. In: *Proceedings of IEEE Conference on Security and Privacy for Emerging Areas in Communications Networks*. Seiten 277–288. Athen, Griechenland, 2005, IEEE Press. ISBN 0-7695-2369-2

[199] SANDER, T. und TSCHUDIN, C.F.: *Protecting Mobile Agents Against Malicious Hosts*. In: *Lecture Notes In Computer Science; Mobile Agents and Security* Bd. 1419. Seiten 44–60, Springer Verlag, 1998. ISBN 3-540-64792-9

[200] SANLI, H.O., OZDEMIR, S. und ÇAM, H.: *SRDA: Secure Reference-Based Data Aggregation Protocol for Wireless Sensor Networks*. In: *Proceedings of IEEE Vehicular Technology Conference*. Seiten 4650–4654. Los Angeles, USA, 2004, IEEE Press. ISBN 0-7803-8521-7

[201] SCATTERWEB: *The self-configuring wireless communication platform*. 2006. http://www.scatterweb.com/

[202] SCHNEIER, B.: *Angewandte Kryptographie – Der Klassiker. Protokolle, Algorithmen und Sourcecode in C*. Pearson Studium, 2005. ISBN 3827372283

[203] SCHÖNING, U.: *Theoretische Informatik – kurzgefaßt*. Spektrum Akademischer Verlag, 2001. ISBN 3827410991

[204] SHAMIR, A.: *Identity-Based Cryptosystems and Signature Schemes*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 47–53. Santa Barbara, USA, 1984, Springer Verlag. ISBN 0-387-15658-5

[205] SHAMIR, A. und TROMER, E.: *On the cost of factoring RSA-1024*. In: *RSA CryptoBytes*, Vol. 6, Nr. 2, Seiten 10–19, 2003. Online verfügbar unter http://www.wisdom.weizmann.ac.il/~tromer/papers/cbtwirl.pdf

[206] SHANNON, C.: *Communication Theory of Secrecy Systems*. In: *Bell System Technical Journal*, Vol. 28, Nr. 4, Seiten 656–715, 1949. Online verfügbar unter: http://www.eecs.umich.edu/~zmao/eecs589/papers/shannon1949.pdf

[207] SHEHAB, M., BERTINO, E. und GHAFOOR, A.: *Efficient Hierarchical Key Generation and Key Diffusion for Sensor Networks*. In: *Proceedings of IEEE Communications Society Conference on Sensor and AdHoc Communications and Networks*. Seiten 76–84. Santa Clara, USA, 2005, IEEE Press. ISBN 0-7803-9011-3

[208] SIMMONS, G. J.: *How to Insure that Data Acquired to Verify Treaty Compliance Are Trustworthy*. In: *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992, Seiten 617–630. ISBN 0879422777

[209] SIMMONS, G. J.: *An introduction to shared secret and/or shared control schemes and their application*. In: *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992, Seiten 441–497. ISBN 0879422777

[210] SIMONOVA, K., LING, A.C.H. und WANG, X.S.: *Location-aware Key Predistribution Scheme for Wide Area Wireless Sensor Networks*. In: *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*. Seiten 157–168. Alexandria, USA, 2006, ACM Press. ISBN 1-59593-554-1

[211] SOG-IS: *Information Technology Security Evaluation Criteria – ITSEC, Ver. 1.2*. 1991. http://www.bsi.bund.de/zertifiz/itkrit/itsec-dt.pdf

[212] SOHRABI, K., GAO, J., AILAWADHI, V. und POTTIE, G.J.: *Protocols for Self-Organization of a Wireless Sensor Network*. In: *IEEE Personal Communications*, Vol. 7, Nr. 5, Seiten 16–27, 2000. ISSN 1070-9916

[213] SPENCER, J.: *The Strange Logic of Random Graphs, Algorithms and Combinatorics 22*. Springer Verlag, 2000. ISBN 3-540-41654-4

[214] SRIKANTH, T.K. und TOUEG, S.: *Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms*. In: *Distributed Computing*, Vol. 2, Nr. 2, Seiten 80–94, 1987. ISSN 0178-2770

[215] STAJANO, F.: *Security for Ubiquitous Computing*. John Wiley and Sons, 2002. ISBN 0-470-84493-0

[216] STAJANO, F. und ANDERSON, R.: *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*. In: *Proceedings of International Workshop on Security Protocols*. Seiten 172–194. Cambridge, UK, 1999, Springer Verlag. ISBN 3-540-67381-4

[217] STRAFGESETZBUCH: *Verletzung von Privatgeheimnissen*. Besonderer Teil, 15. Abschnitt, § 203,
http://dejure.org/

[218] THE PREPARATORY COMMISSION FOR THE COMPREHENSIVE NUCLEAR-TEST-BAN TREATY ORGANIZATION: *Comprehensive Nuclear-Test-Ban Treaty*. 2006.
http://www.ctbto.org/

[219] TILBORG, Henk C. (Hrsg.): *Encyclopedia of Cryptography and Security*. Springer Verlag, 2005. ISBN 038723473X

[220] TRAYNOR, P., CHOI, H., CAO, G., ZHU, S. und PORTA, T. L.: *Establishing Pair-Wise Keys in Heterogeneous Sensor Networks*. In: *Proceedings of IEEE INFOCOM*. Barcelona, Spanien, 2006, IEEE Press. ISBN 1424402212

[221] UCLA PARALLEL COMPUTING LABORATORY: *About GloMoSim*. 2006.
http://pcl.cs.ucla.edu/projects/glomosim/

[222] UCLA PARALLEL COMPUTING LABORATORY: *Parallel Simulation Environment for Complex systems*. 2006.
http://pcl.cs.ucla.edu/projects/parsec/

[223] UNIVERSITÄT KARLSRUHE, UNIVERSITÄT MANNHEIM: *Zuverlässige Informationsbereitstellung in energiebewussten ubiquitären Systemen*. 2007.
http://zeus-bw-fit.de/

[224] UNIVERSITY OF CALIFORNIA BERKELEY: *Tiny OS Hardware Designs*. 2004.
http://www.tinyos.net/scoop/special/hardware

[225] UNIVERSITY OF CALIFORNIA BERKELEY: *TinyOS*. 2005.
http://www.tinyos.net/

[226] UNIVERSITY OF CALIFORNIA BERKELEY: *TinyOS FAQ: What MAC algorithm is used on the your-favorite-mote Mote?* 2007.
http://www.tinyos.net/faq.html#SEC-76

[227] VOGT, H.: *Exploring Message Authentication in Sensor Networks*. In: *Proceedings of European Workshop on Security in Ad Hoc and Sensor Networks*. Seiten 19–30. Heidelberg, 2004, Springer Verlag. ISBN 3540243968

[228] VOYDOCK, V.L. und KENT, S.T.: *Security Mechanisms in High-Level Network Protocols*. In: *ACM Computing Surveys*, Vol. 15, Nr. 2, Seiten 135–171, 1983. ISSN 0360-0300

[229] WACKER, A., HEIBER, T., , CERMANN, H. und MARRON, P.: *A fault-tolerant Key-Distribution Scheme for Securing Wireless Ad-Hoc Networks*. In: *Proceedings of International Conference on Pervasive Computing*. Seiten 194–212. Linz, Österreich, 2004, Springer Verlag. ISBN 3-540-21835-1

[230] WACKER, A., HEIBER, T. und CERMANN, H.: *A Key-Distribution Scheme for Wireless Home Automation Networks*. In: *Proceedings of IEEE Conference on Consumer Communications and Networking*. Seiten 47–52. Las Vegas, USA, 2004, IEEE Press. ISBN 0-7803-8145-9

[231] WACKER, A., KNOLL, M., HEIBER, T. und ROTHERMEL, K.: *A New Approach for Establishing Pairwise Keys for Securing Wireless Sensor Networks*. In: *Proceedings of ACM Conference on Embedded Networked Sensor Systems*. Seiten 27–38. San Diego, USA, 2005, ACM Press. ISBN 1-59593-054-X

[232] WADAA, A., OLARIU, S., WILSON, L. und ELTOWEISSY, M.: *Scalable Cryptographic Key Management in Wireless Sensor Networks*. In: *Proceedings of IEEE Distributed Computing Systems Workshops*. Seiten 797–802. Tokio, Japan, 2004, IEEE Press. ISBN 0-7695-2087-1

[233] WAGNER, D.: *Cryptanalysis of an Algebraic Privacy Homomorphism*. In: *Proceedings of Information Security Conference ISC*. Seiten 234–239. Bristol, UK, 2003, Springer Verlag. ISBN 3-540-20176-9

[234] WAGNER, D.: *Resilient Data Aggregation*. In: *Proceedings of ACM Workshop on Security of Ad hoc and Sensor Networks*. Seiten 78–87. Washington D.C., USA, 2004, ACM Press. ISBN 1-58113-972-1

[235] WEBSTER, A.F. und TAVARES, S.E.: *On the design of S-Boxes*. In: *Proceedings of IACR Advances in Cryptology, CRYPTO*. Seiten 523–534. Santa Barbara, USA, 1986, Springer Verlag. ISBN 0-387-16463-4

[236] WEIMERSKIRCH, A.: *Authentication in Ad-hoc and Sensor Networks*. 2004. Dissertation, Universität Bochum
http://deposit.ddb.de/cgi-bin/dokserv?idn=975679244

[237] WEIMERSKIRCH, A. und WESTHOFF, D.: *Zero-Common Knowledge Authentication for Pervasive Networks*. In: *Proceedings of Workshop on Selected Areas in Cryptography*. Seiten 73–87. Ottawa, Kanada, 2003, Springer Verlag. ISBN 3-540-21370-8

[238] WEISER, M.: *The Computer for the Twenty-First Century*. In: *Scientific American*, Vol. , September, Seiten 94–104, 1991

[239] WEISER, Mark: *Ubiquitous Computing*. 1996. http://sandbox.xerox.com/ubicomp/

[240] WELSH, M., MALAN, D., DUNCAN, B., FULFORD-JONES, T. und MOULTON, S.: *Wireless Sensor Networks for Emergency Medical Care*. 2004. Harvard University http://www.eecs.harvard.edu/~mdw/talks/ge-codeblue.pdf

[241] WESTHOFF, D., GIRAO, J. und ACHARYA, M.: *Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation*. In: *IEEE Transactions on Mobile Computing*, Vol. 5, Nr. 10, Seiten 1417–1431, 2006. ISSN: 1536-1233

[242] WESTHOFF, D., GIRAO, J. und MYKLETUN, E.: *TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks*. In: *To appear in Elsevier Journal on Ad Hoc Networks*, Vol.

[243] WHITING, D., HOUSLEY, R. und FERGUSON, N.: *Counter with CBC-MAC (CCM)*. RFC 3610 (Informational), IETF, 2003. http://www.ietf.org/rfc/rfc3610.txt

[244] WILKE, J.: *Authentischer und effizienter Datentransport in drahtlosen Sensornetzen*. 2006. Studienarbeit, Institut für Telematik, Universität Karlsruhe (TH), http://www.tm.uka.de

[245] WOOD, A.D. und STANKOVIC, J.A.: *Denial of Service in Sensor Networks*. In: *IEEE Computer*, Vol. 35, Nr. 10, Seiten 54–62, 2002. ISSN 0018-9162

[246] WU, Y., MA, D., LI, T. und DENG, R.H.: *Classify Encrypted Data in Wireless Sensor Networks*. In: *Proceedings of IEEE Vehicular Technology Conference*. Seiten 3236–3239. Los Angeles, USA, 2004, IEEE Press. ISBN 0-7803-8521-7

[247] XU, W., MA, K., TRAPPE, W. und ZHANG, Y.: *Jamming sensor networks: attack and defense strategies*. In: *IEEE Network*, Vol. 20, Nr. 3, Seiten 41–47, 2006. ISSN 0890-8044

[248] XU, W., TRAPPE, W., ZHANG, Y. und WOOD, T.: *The feasibility of launching and detecting jamming attacks in wireless networks*. In: *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*. Seiten 46–57. Urbana-Champaing, USA, 2005, ACM Press. ISBN 1-59593-004-3

[249] YANG, C., ZHOU, J., ZHANG, W. und WONG, J.: *Pairwise key establishment for large-scale sensor networks: from identifier-based to location-based*. In: *Proceedings of ACM International Conference on Scalable Information System*. Hong Kong, China, 2006, ACM Press. Artikel Nr. 27, ISBN 1-59593-428-6

[250] YANG, Y., WANG, X., ZHU, S. und CAO, G.: *SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks*. In: *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*. Seiten 356–367. Florenz, Italien, 2006, ACM Press. ISBN 1-59593-368-9

[251] YU, Z. und GUAN, Y.: *A Key Pre-Distribution Scheme Using Deployment Knowledge for Wireless Sensor Networks*. In: *Proceedings of Information Processing in Sensor Networks ISPN*. Seiten 261–268. Los Angeles, USA, 2005, IEEE Press. ISBN 0-7803-9202-7

[252] ZHANG, W., DAS, S.K. und LIU, Y.: *A Trust Based Framework for Secure Data Aggregation in Wireless Sensor Networks*. In: *Proceedings of IEEE Conference on Sensor and Adhoc Communications and Networks*. Washington, USA, 2006, IEEE Press. ISBN 1-4244-0626-9

[253] ZHANG, Y., LIU, W., FANG, Y. und WU, D.: *Secure Localization and Authentication in Ultra-Wideband Sensor Networks*. In: *IEEE Journal on Selected Areas in Communications*, Vol. 24, Nr. 4, Seiten 829–835, 2006. ISSN 0733-8716

[254] ZHANG, Y., LIU, W., LOU, W. und FANG, Y.: *Securing Sensor Networks with Location-Based Keys*. In: *Proceedings of IEEE Wireless Communications and Networking Conference*. Seiten 1909–1914. New Orleans, USA, 2005, IEEE Press. ISBN 0-7803-8966-2

[255] ZHANG, Y., LIU, W., LOU, W. und FANG, Y.: *Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks*. In: *IEEE Journal on Selected Areas in Communications*, Vol. 24, Nr. 2, Seiten 247–260, 2006. ISSN 0733-8716

[256] ZHOU, Y., ZHANG, Y. und FANG, Y.: *LLK: a link-layer key establishment scheme for wireless sensor networks*. In: *Proceedings of IEEE Wireless Communications and Networking Conference*. Seiten 1921–1926. New Orleans, USA, 2005, IEEE Press. ISBN 0-7803-8966-2

[257] ZHU, S., SETIA, S. und JAJODIA, S.: *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks*. In: *Proceedings of ACM Conference on Computer and Communications Security*. Seiten 62–72. Washington D.C., USA, 2003, ACM Press. ISBN 1-58113-738-9

[258] ZHU, S., SETIA, S. und JAJODIA, S.: *LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Networks*. In: *ACM Transactions on Sensor Networks*, Vol. 2, Nr. 4, Seiten 500–528, 2006. ISSN 1550-4859

[259] ZHU, S., SETIA, S., JAJODIA, S. und NING, P.: *An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks*. In: *Proceedings of IEEE Symposium on Security and Privacy*. Oakland, USA, 2004, IEEE Press. ISBN 0-7695-2136-3

[260] ZIMMERMANN, H.: *OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection*. In: *IEEE Transactions on Communications*, Vol. 28, Nr. 4, Seiten 425–432, 1980. ISSN: 0096-2244

ISBN: 978-3-86644-142-2

www.uvka.de