Eine Machbarkeitsstudie zum Einsatz eines zentralen Authentisierungssystems in Wireless Mesh Networks

Studienarbeit

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

eingereicht von:	Christian Ricardo Kühne Gómez
geboren am:	02.06.1986
in:	Berlin

Gutachter(innen): Prof. Dr. Jens-Peter Redlich

eingereicht am: ..... verteidigt am: .....

Inhaltsverzeichnis

1 Einleitung

Die Forschung zur drahtlose Kommunikation hat in den letzten beiden Jahrzehnten im Bereich der zivilen Nutzung einen weiteren Entwicklungsschritt hin zu einem neuen Netzwerktyp vollzogen: dem drahtlosen Maschennetzwerk (engl. Wireless Mash Network, kurz: WMN). Die Sicherheit und die darin einbegriffene Zuverlässigkeit eines solchen Netzwerktyps sind aktuelle Themen in diesem Forschungsgebiet und begründen auch das Thema dieser Arbeit.

Arbeitsdefinition

Drahtlose Maschennetzwerke sind multi-hop Netzwerke, deren Knoten aus Maschen-Routern und Maschen-Klienten bestehen. Die Maschen-Router bilden dabei unter sich ein sogenanntes Backbone-Netz, das den Maschen-Klienten bestimmte Netzwerkdienste wie zum Beispiel Bridging, Routing oder Gateways anbietet. Damit ein Maschen-Klient diese Dienste nutzen kann, bieten alle Maschen-Router in ihrer doppelten Funktion als Access Point ein sogenanntes Zugangsnetz an. Die Besonderheit von WMNs ist ihr flexibler Aufbau und ihre Fähigkeit, dynamisch Selbstorganisation zu betreiben, die unter anderem auch Mechanismen der Selbstkonfiguration und Selbstheilung einschließen kann. Der Zweck dieser Selbstorganisation ist die (Wieder-)Herstellung und Aufrechterhaltung der Netzwerkkonnektivität zwischen den einzelnen Knoten und des Weiteren das Angebot an Netzwerkdiensten. Diese Netzwerkdienste beinhalten zum Beispiel die Vernetzung unterschiedlicher Kommunikanten, die geographisch auseinander liegen, oder eben der Zugang zu anderen Netzen wie das Internet, Wi-Fi, zelluläre oder Sensoren-Netzwerken mittels Gateways.

An Hand der Literatur lassen sich verschiedene theoretische Vorteile festhalten:[1] (a) WMNs sind in größeren geographischen Gebieten mit geringen oder keinen Informations- und Kommunikationsinfrastrukturen einsetzbar; (b) Die Installations-, Wartungs- und Betriebskosten sind gegenüber drahtgebundenen Netzwerktypen geringer; (c) Die flexibel erweiterbare Netzwerkarchitektur und die Eigenschaft der Selbstorganisation tragen zur einem gewissen Maß an Fehlertoleranz bei. Man

[1] Vergleiche zum Beispiel Akyildiz et al. (2005, S. 5).

kann der Gesamtheit dieser Gesichtspunkte jedoch skeptisch gegenübertreten und sich fragen, in wie fern es sich um überzogene Versprechungen einer neu aufkommenden Technologie oder um realistische technische Vorhaben handelt.

Gesellschaftlicher Nutzen

Der gesellschaftliche Nutzen dieser Netzwerktechnologie ist vielfältig und lässt gegenüber den klassischen drahtgebundenen Netzen viele neue unterschiedliche Anwendungsszenarien zu. Ein erstes Szenario wäre die Vernetzung von Gemeinschaften oder Nachbarschaften mit geographischer Nähe, um eine robuste und von den Telekommunikationsanbietern unabhängige Netzwerkinfrastruktur aufzubauen. In einem zweiten Szenario, dem Stadtszenario, sind Metropolitan Area Networks (MAN) von Bedeutung. Auch hier wäre das Argument, dass WMNs eine kostengünstige und flexible Alternative für weiträumige geographische Gebiete darstellt, um Peer-to-Peer-Kommunikation (P2P) und einen Internet-Zugang zu ermöglichen. Die Freifunk-Initiative sieht in dieser Technologie sogar eine Möglichkeit, durch freie Netzwerke die Kommunikationsmedien zu demokratisieren.[2] Ein drittes Szenario beträfen spontane bzw. Notfalleinsätze in Katastrophen- oder Krisengebieten, um eine P2P-Kommunikation aufzubauen, die bspw. als Grundlage für die Koordination und Organisation von Rettungsteams oder Hilfsgütern dient. Doch welche Bedeutung hat die Sicherheit in WMNs?

Sicherheitsbedürfnisse

Die Sicherheit in WMNs spielt hinsichtlich der gesellschaftlichen Akzeptanz der Technik für Informations- und Kommunikationszwecke eine wichtige Rolle und orientiert sich vornehmlich an den Sicherheitsbedürfnissen der Menschen. Zu diesen Bedürfnissen gehören z. B. der Schutz des privaten Lebensbereichs gegenüber Personen, Organisationen und Institutionen oder die Datensicherheit in Organisationen und Unternehmen. Diese allgemeinen Sicherheitsbedürfnisse (oder Schutzziele) lassen sich weiter verfeinern in Ziele wie Authentizität, Vertraulichkeit, Integrität und Verfügbarkeit.

Sicherheitsbedürfnisse sind eine Reaktion auf Fehler, Unfälle und Angriffe auf das technische bzw. organisatorische System.[3] Diese Phänomene treten auf, weil Systeme konzeptionelle oder konkrete Schwachstellen besitzen. Eine Schwachstelle ist eine Eigenschaft des Systems oder seiner Umgebung, die im Zusammen-

[2] Siehe Freifunk (2013).
[3] Vergleiche Anderson (2001, S. 11).

hang mit einer äußeren oder inneren Bedrohung zu einem unsicheren Zustand (gemäß einer formulierten Sicherheitsrichtlinie) führen kann. Wird eine Schwachstelle ausgenutzt, wird die Bedrohung zu einer Gefahr und schließlich zu einem Angriff und Schadensfall.

Problemstellung

Die vorliegende Arbeit nimmt ihren Stand in der Problemstellung, welche Konzepte im Bezug auf die Sicherheitsarchitektur und -Protokolle von WMNs geeignet sind, um als notwendige Grundlage für weitere Netzwerkdienste zu dienen. Die drei Wissenschaftler Martignon, Paris und Capone (2009) veröffentlichten dazu im Jahre 2009 eine Arbeit, in der sie das Konzept von MobiSEC vorstellten. Es handelt sich dabei nach ihren Angaben um ein vollständiges Sicherheitsprotokoll,[4] das sowohl die Backbone-Sicherheit als auch die Zugangssicherheit behandelt und dabei auf bestehende Standards der drahtlosen Sicherheitstechnik zurückgreift. Eine der grundsätzlichen Entwurfsentscheidungen ist der zentralisierte Authentisierungsmechanismus samt Schlüsselverteilung. Das Bemerkenswerte dieser Arbeit liegt vor allem darin, dass sie den wissenschaftlichen Streit über die Frage, ob zentralisierte oder dezentralisierte Ansätze zu verfolgen sind, wieder aufnimmt. Typischerweise werden in klassischen drahtgebundenen LANs zentralisierte Ansätze erfolgreich angewendet. Spätestens seit Ende der 1990er Jahren herrscht jedoch die wissenschaftliche Meinung, dass WMNs insbesondere im Bezug auf die Anforderungen der Selbstorganisation (u. a. zur Vermeidung eines „Single-Point-of-Failure“) und Skalierbarkeit ein dezentrales Sicherheitsprotokoll benötigen.[5]

Dies muss jedoch nicht so sein. Wie schon eingangs angedeutet lassen WMNs eine Vielfalt von Einsatzmöglichkeiten zu, von denen einige auch weniger restriktive Anforderungen bzgl. der Sicherheit der drahtlosen Kommunikation stellen. Denkbar wären Heim- und Unternehmensnetzwerke, bei denen drahtgebundener Netzwerkinfrastrukturen teuer oder physikalisch nicht umsetzbar sind. Hier kann man von einer geringen Mobilität, keinen Energiebeschränkungen, einem langlebigen Netzwerk, zuverlässige Knoten und ausreichender Rechenleistung ausgehen.

Ein weiteres Argument besteht darin, dass die in MobiSEC zum Einsatz kom-

[4] Der originale Ausdruck ist „complete security architecture“. Es lässt sich streiten darüber, in wie fern das MobiSEC-Konzept eine Architektur oder eher ein Protokoll darstellt. Ich werde aus Gründen der Konsistenz und Verständlichkeit mit dem Ausdruck „Protokoll“ weiterarbeiten.

[5] Vergleiche Akyildiz et al. (2005, S. 473) und Zhou und Haas (1999, S. 3).

menden Techniken schon im einzelnen gut untersucht und erprobt worden sind, während die dezentralen Ansätze zum gegenwärtigen Zeitpunkt nach wie vor Gegenstand aktueller Forschung sind.[6] (Man kann daher berechtigterweise die Frage stellen, ob die theoretische Diskussion nicht hemmend auf die Untersuchung zentralisierter Ansätze gewirkt hat.[7])

Fragestellung

Im Rahmen des oben genannten Problems liegt dieser Arbeit die Fragestellung zugrunde, ob klassische Ansätze zentralisierter Sicherheitsarchitekturen und -Protokolle für WMNs zu Gunsten von dezentralisierten aufgegeben werden müssen. Zur Beantwortung dieser Frage kommt es vor allem darauf an zu prüfen, ob die untersuchte Sicherheitstechnik ihr Versprechen der Sicherheit einlöst, ohne die Leistungsfähigkeit von WMNs signifikant zu beeinträchtigen.

WMN-Sicherheit

Bevor ich auf den Untersuchungsgegenstand näher eingehen werde, möchte ich nun noch etwas genauer auf die WMN-Sicherheit eingehen. Der erste Schritt, um sich mit der Sicherheit von WMNs zu beschäftigen, ist, die Charaktermerkmale von WMNs herauszustellen. Erst dadurch lassen sich systematisch Schwachstellen, Bedrohungen und Angriffe ableiten, aus denen Gegenmaßnahmen entwickelt werden können. Zu den besonderen Merkmalen von WMNs gehört die Infrastrukturlosigkeit, das Multi-hopping, die Benutzung eines offenen Übertragungsmediums und (je nach Szenario) freie Zugänglichkeit zu den Maschen-Routern. Genauer gesagt sind dies alles Merkmale, die hauptsächlich auf Maschen-Router zu treffen. Für die Maschen-Klienten hingegen gilt zudem, dass sie bestimmten Energie-, Rechen- und Speicherbeschränkungen unterliegen können.

Daraus ergeben sich jedoch besondere Herausforderungen für die WMN-Sicherheit. Die WMN-Sicherheit muss nicht nur überdacht sondern möglicherweise vollständig neu konzipiert werden. Der Forschungsbereich der WMN-Sicherheit steht noch relativ am Anfang seiner Entwicklung. Redwan und Kim (2008, S. 1) schrieben dazu noch 2008: »The state-of-the-art work is still insuficient for deploying sizeable WMNs because important aspects such as network radio range,

[6] Siehe zum Beispiel Martignon et al. (2010) oder Gennaro et al. (2008).

[7] Dazu Zhou und Haas (1999, S. 3): „[...] to achieve high survivability, ad hoc networks should have a distributed architecture with no central entities. Introducing any central entity into our security solution could lead to significant vulnerability; that is, if this centralized entity is compromised, then the entire network is subverted."

network capacity, scalabilty, manageability, and security still remains open.«

Schwachstellen

Dies sind Eigenschaften, die im besonderen Maße WMNs auszeichnen und insofern auf potentielle Schwachstellen hinweisen. Auf einige dieser Schwachstellen möchte ich kurz eingehen. Salem und Hubaux (2006, S. 3) stellen fest, dass das Multi-hopping Detektierungs- und Reaktionsmechanismen von Angriffen verzögert; Die physikalische Schutzlosigkeit von Knoten ermöglicht das Einfangen, Klonen oder Manipulieren dieser Geräte; Redwan und Kim (2008, S. 3) weisen auf das geteilte drahtlose Medium als Schwachstelle hin, welches das Abhören, Manipulieren oder Stören der Kommunikation möglich macht; Die Infrastrukturlosigkeit, die durch Mechanismen der Selbstorganisation kompensiert werden soll, schließt die Existenz einer zentrale Vertrauenspartei (Central Authority, Trusted Third Party, Key-Server) ohne weiteres Zutun vorerst aus. Hinzu treten Energie- und Speicherbeschränkungen auf Seiten der mobilen Maschen-Klienten, die bei unsachgemäßer Nutzung zu Geräteausfällen oder Dienstverweigerung führen kann. Ein weiterer Punkt ist die dynamische Veränderung der Netzwerktopologie, wenn man zum Beispiel das Mobility für mobile Endgeräte zulässt. Sind die Veränderungen zu stark, können Schutz-, Detektierungs- und Reaktionsmechanismen nur noch suboptimal funktionieren. Als letzten Punkt sei die freie Zugänglichkeit von Netzwerkknoten als Schwachstelle aufgeführt. Um die Kosten gering zu halten, bleibt der Einsatz von manipulationsresistente Hardware aus. Dies erleichtert nicht nur einen Zugriff auf den Knoten, sondern gefährdet, bspw. mit der Möglichkeit der Einflussnahme auf das Routing-Protokoll, auch die Sicherheit des gesamten Netzwerkes.

Angriffe

Auf eine ausführliche Angriffsmodellierung (engl. Threat-Modelling), wie sie sich für eine systematische Analyse der Angriffe gehört, wird an dieser Stelle verzichtet, da dies nicht das Hauptthema dieser Arbeit ist. Ich möchte daher nur auf einige ausgewählte aktive Angriffe und Gegenmaßnahmen verweisen, und zwar hauptsächlich im Bezug auf das OSI-Schichtenmodell. Wie schon oben angedeutet zählen gerade die physikalische und die Sicherungsschicht zu den kritischen Bereichen der WMN-Protokolle. Auf der Ebene der physikalischen Schicht lassen sich Jamming-Angriffe festhalten, die zu der Klasse von Denial-of-Service-Angriffen (kurz: DoS) gehören und eine grundsätzliche Kommunikation verhin-

dern können. Auf der Ebene der Sicherungsschicht sind DoS-Angriffe bspw. auf das MAC-Protokoll möglich und damit eine Dienstverweigerung eines Knotens oder eine topologische Verzerrung erwirken können. Wurmloch-Angriffe wiederum ermöglichen (in einigen Fällen ohne jegliche Authentisierung und Autorisierung) unveränderte Pakete über kürzere Wege als den normalen Wegen, nämlich über legitime Knoten, weiterzuleiten. Dies führt üblicherweise zu Konsistenzprobleme im Routing-Protokoll. Auf der Ebene der Vermittlungsschicht finden sich aktive Angriffe auf das Routing-Protokoll, bspw. Veränderung der Routing-Tabellen oder der Routing-Packete. Newsome et al. (2004) behandeln eine Angriffsklasse, die sich Sybil-Angriffe nennt, bei denen ein bösartiger Knoten unberechtigt mehrere Identitäten annimmt.

Neben den genannten aktiven Angriffen zählt das Abhören der Kommunikation als passiver Angriff auf die Vertraulichkeit der übertragenen Informationen. Ich komme nun zum Untersuchungsgegenstand.

Untersuchungsgegenstand

Um dieser Arbeit eine vernünftige Arbeitsperspektive zu geben, ist der Untersuchungsgegenstand wie folgt festgelegt: Der Gegenstand dieser Arbeit ist die Integration von MobiSEC in das Testbed der Humboldt-Universität zu Berlin (Humboldt-Wireless-Lab-Testbed, kurz: HWL-Testbed[8]), sowie die Überprüfung und Beurteilung (Evaluation) ihrer praktischen Eignung. Zwei Einschränkungen habe ich allerdings vorgenommen: (a) Das Protokoll gliedert sich in zwei Sicherheitsbereiche, Zugangssicherheit und Backbone-Sicherheit, wobei ich mich in der Arbeit auf letzteres beschränke; (b) Anstatt das echte Testbed zu verwenden benutze ich den Network Simulator 2 (kurz: NS2[9]), der üblicherweise eine Bewährungshilfe für die Migration in das echte Testbed darstellt.

Erkenntniswert

Die oben genannte Frage wurden von Anfang an ergebnisoffen behandelt. Der Erkenntniswert dieser Arbeit besteht darin, die Diskussion über das MobiSEC-Konzept um weitere Aspekte zu bereichern, und zwar in der Theorie, Implementierung und Simulation. Weiterhin werde ich auch versuchen, Erfahrungen über Problemvorkommen und Unzulänglichkeiten darzulegen.

Aufbau der Arbeit

Meine Arbeit gliedert sich in fünf Kapitel. Das zweite Kapitel ist der theoretische

[8] Vergleiche Zubow et al. (2012).

[9] Die Projektseite befindet sich unter http://nsnam.isi.edu/nsnam/index.php/User_Information.

Teil. Hier werde ich versuchen das Sicherheitsprotokoll zusammenzufassen, in dem ich das zugrunde liegende Netzwerkmodell und die Sicherheitsannahmen explizit darstelle. In dem ich im dritten Kapitel analytisch zeige, dass der Overhead nur sehr gering ausfällt, habe ich Grund zur Annahme, dass sich dies auch empirisch in einer Simulation zeigen lässt. Im vierten Kapitel werde ich auf den Aufbau und die Funktionsweise meines Software-Entwurfs eingehen. Das fünfte Kapitel beinhaltet die Darstellung der Simulationsergebnisse einschließlich ihrer Bewertung. Im sechsten und letzten Kapitel werde ich die Arbeit noch einmal zusammenfassen und die wichtigsten Erkenntnisse herausarbeiten.

2 MobiSEC-Sicherheitsprotokoll

Ich möchte in diesem Kapitel näher auf das Sicherheitsprotokoll eingehen. Zu erst wird es mir darum gehen, das Netzwerkmodell, das dem Sicherheitsprotokoll zugrunde liegt, klar zu formulieren. Danach werde ich die Schwachstellen bzw. das Angreifermodell im Bezug auf das Netzwerkmodell rekonstruieren, die als explizite und implizite Überlegungen in das Sicherheitsprotokoll von MobiSEC eingegangen sind. Aufbauend auf diesen Vorüberlegungen werde ich das System von Gegenmaßnahmen erklären, die das Sicherheitsprotokoll konstituieren.

2.1 Netzwerkmodell

Der Entwurf eines Sicherheitsprotokolls setzt das Wissen um die Schwachstellen und Angriffsmöglichkeiten voraus. Diese ergeben sich wiederum erst aus einem klar formulierten Netzwerkmodell. Daher möchte ich in diesem Abschnitt kurz auf das zugrunde liegende Netzwerkmodell eingehen.

Das Netzwerk setzt sich – wie schon in der Arbeitsdefinition erwähnt – aus Maschen-Klienten und Maschen-Routern zusammen. Kommuniziert wird einzig und allein über das drahtlose Medium. Über dieses Medium etablieren die Maschen-Router im Ad-hoc-Modus das Multi-hop-Netzwerk. Die Maschen-Router übernehmen dabei die Routing-Aufgabe eines Backbone-Netzes, während die Maschen-Klienten über die Maschen-Router Zugang zu allen Netzwerkdiensten haben. Da-

raus ergibt sich eine doppelte Funktion der Maschen-Router, für die sie mit zwei drahtlosen Schnittstellen ausgerüstet sind: während die eine Schnittstelle die Kommunikation innerhalb des Backbone-Netzes ermöglicht, dient die andere Schnittstelle dem Zugang für die Maschen-Klienten. Mit anderen Worten, ein Maschen-Router nimmt sowohl die Rolle eines Routers als auch eines Access Points (AP) ein.

Zusätzlich existiert ein dedizierter Maschen-Knoten, der eine besondere Rolle in dem Maschen-Netzwerk einnimmt: Er ist größtenteils für die Sicherheit zuständig und übernimmt die zentralen Aufgaben der Schlüsselverwaltung, Schlüsselverteilung und Authentifikation innerhalb und gegenüber dem gesamten Netzwerk. Ich werde ich fortführend als Schlüsselverwaltungs-/Authentifizierungsserver (kurz: SA-Server) bezeichnen.

Die Autoren von MobiSEC gehen weiterhin davon aus, dass (a) die Links zwischen den Knoten symmetrisch sind, (b) eine sehr geringe Mobilität vorliegt, (c) eine ausreichende Energieversorgung vorhanden ist, und (d) innerhalb des Netzwerkes eine Zeitsynchronisation vorgenommen wird (zum Beispiel durch die Verwendung des Network-Time-Protokolls).

2.2 Schwachstellenanalyse/Angreifermodell

Um das Sicherheitsmodell hinter MobiSEC genauer begreifen zu können, möchte ich nun näher auf die von den Autoren formulierten Schwachstellen bzw. Angriffsmöglichkeiten eingehen. Es sind im Wesentlichen Überlegungen aus dem Bereich der Netzwerksicherheit.

Die erste Überlegung betrifft das drahtlose geteilte Medium, welches grundsätzlich eine offene Angriffsfläche bietet. Selbst unter Anwendung kryptographischer Verfahren zur Kanalverschlüsselung sind prinzipiell Angriffe auf das Medium, und dadurch auch auf die Verkehrsdaten bzw. das eingesetzte Protokoll möglich. Diese beinhalten zum einen passive Angriffe durch bloßes Abhören, Aufzeichnen und Analysieren von Verkehrsdaten und zum anderen aktive Angriffe wie zum Beispiel Replikation, Modifikation, Fälschung von Verkehrsdaten. Und schließlich ist auch die Verhinderung übermittelter Verkehrsdaten ein aktiver Angriff. Lässt man physikalische Angriffe in Sinne einer Störung des Mediums (z. B. Jamming) oder

Hardware-Manipulation außen vor, bleiben im Grunde vier große Angriffsklassen übrig, die über das drahtlose Medium möglich sind: (i) DoS-Angriffe, (ii) kryptoanalytische Angriffe, die auf die verschlüsselten Inhalte oder die verwendeten Schlüssel abzielen, (iii) Protokoll-Angriffe und (iv) Angriffe auf die Implementierung.

Zwei Vorwegnahmen: Die erste Angriffsklasse gehört bis heute zu den härtesten Problemen der Netzwerksicherheit. Dies ist vermutlich auch der Grund, warum die Autoren diese Klasse nicht weiter beachtet haben. Auch die vierte Angriffsklasse bleibt bei den Autoren außen vor, da es in erster Linie um Sicherheitsaspekte auf der Stufe des Protokoll-Designs geht.

Die Absicherung des Mediums führt zu der nächsten sicherheitstheoretischen Überlegung, nämlich der Sicherstellung, dass nur authentifizierte und gleichermaßen autorisierte Geräte am Netzwerk teilnehmen dürfen (bzw. über ein abgesichertes Medium mit den anderen Teilnehmern kommunizieren dürfen). Es darf nicht vergessen werden, dass das Netzwerkmodell von MobiSEC zwei getrennte Funktionsbereiche vorsieht, also auch zwei Zugangsmechanismen zu erwarten sind. Die Wahl und Vergabe der Credentials (dt. Nachweise) sowie der eingesetzten Authentisierungs-, Autorisierungs- und Accounting-Mechanismen, mittels derer die digitalen Identitäten der Geräte verifiziert werden, ist hier eher als eine Herausforderung zu sehen als eine Schwachstelle.

Wie schon angesprochen, existiert in dem Netzwerk ein dedizierter Knoten, der zentrale Sicherheitsaufgaben übernimmt. In den Sicherheitsdiskursen ist auch die Rede vom „Single-Point-of-Failure“, womit die Abhängigkeit und Verwundbarkeit des Netzwerks zum Ausdruck gebracht wird. Dieser zentralen Position ist es zuzuschreiben, dass (a) sowohl die Kommunikationsfähigkeit als auch (b) die Vertraulichkeit und Integrität der Kommunikation des gesamten Netzwerkes von ihm abhängen. Beide Aspekte haben jedoch eine unterschiedliche Qualität. Im Bezug auf (a) ist zum Beispiel denkbar, dass der SA-Server vom Rest des Netzwerkes durch Jamming-Angriffe isoliert werden kann. Dies hat zur Folge, dass die Maschen-Router kein neues Schlüsselmaterial erhalten und damit das Routing verweigern. Auf der einen Seite bedeutet dieser Angriff eine bestenfalls temporäre Kommunikationsunfähigkeit, aber auf der anderen Seite werden die Kommunikationsdaten nicht Preis gegeben.

Im Bezug auf (b) ist es denkbar, dass ein Angreifer den SA-Server manipulieren kann. In diesem Fall ist anzunehmen, dass der gesamte Netzwerkverkehr kompromittiert ist, da der Angreifer potentiell Zugriff auf den Generator für das Schlüsselmaterial besitzt. Im Vergleich zu (a) ist dieser Angriff subtiler und damit ungleich schwerer zu detektieren. Der Schaden wird jedoch gleich hoch, wenn nicht sogar höher sein.

Es soll im Folgenden nun darum gehen, welche Gegenmaßnahme die Autoren für die hier angesprochenen Schwachstellen getroffen haben.

2.3 Sicherheitsmodell und -Protokoll

Vor dem Hintergrund des vorgestellten Netzwerkmodells und der rekonstruierten Schwachstellenanalyse ist es vor allem die Netzwerksicherheit, welche die Autoren im Auge haben. Wie schon im Abschnitt zuvor angedeutet, muss zum einen die Zugangskontrolle (für den Eintritt in das Netzwerk) und zum anderen die Absicherung des Mediums (Kommunikationskanals) behandelt werden. Generell verfahren die Autoren nun so, dass in ihrem Sicherheitsprotokoll fast nur offene Standardtechniken der drahtlosen Netzwerksicherheit zum Einsatz kommen. Dies ist insofern günstig, da Standards wie z. B. 802.1X in der Theorie als auch in der Praxis bereits einigen ernsthaften Prüfungen unterzogen worden sind.[10]

Das Sicherheitsmodell von MobiSEC ist folgendermaßen aufgebaut: Entsprechend dem oben genannten Netzwerkmodell werden zwei logische Sicherheitsbereiche eingeführt, nämlich die Backbone-Sicherheit zwischen den Maschen-Router und die Zugangssicherheit, die zwischen einem einzelnen Maschen-Router (in seiner Rolle als AP) und den mit ihm kommunizierenden Maschen-Klienten besteht. Für die Zugangssicherheit wird das 802.11i-Protokoll verwendet, wohingegen für die Backbone-Sicherheit eine Kombination aus 802.11i[11], TLS[12] und WEP[13] verwendet wird.

Genauer gesagt wird das 802.11i im Verbund mit dem TLS-Protokoll für eine

[10] Siehe zum Beispiel den ernstzunehmenden Angriff von Duckwall (2011).
[11] Siehe IEEE (2004a).
[12] Siehe Dierks und Allen (1999).
[13] Nach 802.11i. Andere Protkolle zur Verschlüsselung des Link-Layers sind möglich, z. B. TKIP oder CCMP.

sichere Authentifikation verwendet und hinsichtlich der Kanalverschlüsselung das WEP-Protokoll für bereits authentisierter Maschen-Router.

Um die große Menge von Terminals von autorisierten Maschenknoten (Maschen-Klienten und Maschen-Routern) zu unterscheiden, erhält jeder Maschenknoten ein bzw. zwei Zertifikate innerhalb einer Public-Key-Infrastruktur (PKI). Das erste Zertifikat dient der Authentisierung für das Zugangsnetz. Mit einem zweiten Zertifikat erhält der Maschen-Knoten Zugang zum Backbone-Netz. Mit anderen Worten heißt das, dass ein Maschen-Klient nur *eine* Phase der Authentisierung durchläuft, während ein Maschen-Router *zwei* Phasen der Authentisierung durchläuft. Nun die beiden Bereiche im Einzelnen.

2.3.1 Zugangssicherheit

Da das 802.11i-Protokoll sehr komplex ist, werde ich in diesem Unterabschnitt versuchen, ihre wichtigsten Grundzüge zu beschreiben. 802.11i trägt den stattlichen Namen „Robust Security Network“ (kurz: RSN), dessen Ziel es ist, eine starke Authentifikation sowie die Schlüsselverwaltung im Bezug auf den WLAN-Zugang zu organisieren. Das Kernelement vom 802.11i-Protokoll bildet nach wie vor das 802.1X-Protokoll,[14] welches die Authentifikation und Schlüsselverwaltung organisiert. Darin werden drei Rollen unterschieden: Suplikant, Authentisierungsserver und Authenticator.[15]

1. Der Suplikanten ist das Endgerät, der Netzwerkdienste in Anspruch nehmen möchte, die über einen Port des Authenticators angeboten werden.

2. Der Authenticator kann ein Switch oder Access Point sein. Er vermittelt zwischen dem Suplikanten und dem Authentisierungsserver. Der Authenticator ist auch vergleichbar mit einem Einwahlknoten, der eine portbasierte Zugangskontrolle implementiert. An dieser Stelle kommt ein sogenannter Dualport-Mechanismus zum Einsatz: Während ein *unkontrollierter* Port für die Abwicklung des Authentisierungsprotokolls verwendet wird, lässt der andere *kontrollierte* Port nur Pakete von authentifizierten Partnern passieren.

[14] Siehe IEEE (2004b).
[15] Vergleiche Eckert (2008, S. 347).

3. Der Authentisierungsserver authentifiziert den Suplikanten und teilt dem Authenticator mit, ob der Suplikant berechtigt ist, Zugang zu den Netzwerkdiensten zu erhalten. In der Regel handelt es sich bei dem Authentisierungsserver um einen RADIUS-Server.

Wenn die Authentisierung des Suplikanten gegenüber dem Authentisierungsserver erfolgreich war, wird der kontrollierte Port für den authentifizierten Suplikanten freigeschaltet und eine verschlüsselte Punkt-zu-Punkt-Verbindung mittels des TKIP-Verfahren (oder alternativ das CCMP-Verfahren) zwischen dem Suplikanten und dem Authenticator hergestellt.

Im Sinne der Rollen des MobiSECs Protokolls entspricht der Suplikant einem Maschen-Klienten, der Authenticator einem Maschen-Router (in seiner Rolle als AP) und der Authentisierungsserver dem SA-Server.

Das 802.11i-Protokoll sichert somit sowohl den Zugang (Authentifikation) zum Netzwerk als auch den Link-Layer (Integrität und Vertraulichkeit) zwischen Maschen-Klienten und Maschen-Router als AP ab. Doch die weitere drahtlose Kommunikation neben dem hier behandelten Authentifikationsprozesses ist dadurch nicht abgesichert. Dies ist nun Gegenstand der MobiSEC-Backbone-Sicherheit.

2.3.2 Backbone-Sicherheit

Die Backbone-Sicherheit wird durch ein 2-Phasensystem erreicht: Die erste Phase entspricht dem Authentifikationsprozess eines Maschen-Knotens (nach 802.11i), um Zugang zu dem Netzwerk zu erhalten. Nach Vollendung der ersten Phase wird ein zweiter Authentifikationsprozess mittels TLS eingeleitet, der dazu dient, den Maschen-Knoten das Schlüsselmaterial für die Partizipation am Backbone zukommen zu lassen. Genauer gesagt führen der Maschen-Knoten und der SA-Server mit Hilfe des TLS-Protokolls eine gegenseitige Authentifikation auf Basis eines Zertifikataustausches und der gegenseitigen Verifizierung der privaten Schlüssel durch, so dass sie daraufhin einen sicheren Kanal aufbauen. Das Resultat ist eine sichere Ende-zu-Ende-Verbindung. Über diese gesicherte TLS-Verbindung übermittelt der SA-Server (der auch die Aufgabe der Schlüsselverwaltung übernimmt) das Schlüsselmaterial, das im gesamten Backbone verwendet wird.

Jeder Maschen-Router muss also diese beiden Phasen durchschritten haben, um in Besitz des gültigen Schlüsselmaterials zu kommen. Nun ist die Grundidee, dass für das gesamte Backbone-Netz ein gemeinsamer Gruppenschlüssel verwendet wird. Das heißt, dass jede Punkt-zu-Punkt-Verbindung zwischen zwei Maschen-Routern auf dem Link-Layer mit dem gleichen Schlüssel verschlüsselt wird. Damit dieser allerdings geschützt bleibt gegen kryptographische Angriffe mittels Kryptoanalysen, haben die Autoren eine zusätzliche Gegenmaßnahme getroffen: Ein Schlüsselverteilungsprotokoll (Key-Distribution-Protocol, kurz: KDP) sorgt dafür, dass alle Maschen-Router periodisch vom SA-Server mit einer Schlüsselliste versorgt werden. Die Schlüssel dieser Liste werden dazu verwendet, den Gruppenschlüssel synchron in regelmäßigen Abständen auszutauschen. Der periodische Schlüsselaustausch begrenzt einerseits die Anzahl der Kryptotexte, die mit einem gleichen Schlüssel erzeugt werden und andererseits die Gültigkeit der verwendeten Schlüssel selbst. Letztes trägt zu der Sicherheit künftiger Kommunikation bei. Wie ist nun das KDP aufgebaut?

2.4 Schlüsselverteilungsprotokoll (KDP)

Bei dem Schlüsselverteilungsprotokoll handelt es sich um ein einfaches Request-and-Response-Protokoll mit genau einem Round Trip. Das Ziel des Protokolls ist es, dass jeder Maschen-Router eine valide Schlüsselliste besitzt und die darin enthaltenen Schlüssel zum richtigen Zeitpunkt einsetzt. Nur wenn alle Maschen-Router zu jedem Zeitpunkt ihre Sitzungsschlüssel synchron verwenden, ist eine problemlose Kommunikation möglich. Ich werde nun als erstes das Standardverfahren („Server-Driven“) erläutern und danach die optimierte Variante („Client-Driven“). Ein detaillierter Aufbau der KDP-Pakete, wie ich sie in meiner softwaretechnischen Implementierung verwendet habe, ist den beiden Erläuterungen nachgestellt.

„Server-Driven“-Verfahren

In diesem Modus erhält der Maschen-Router durch eine einfache Anfrage beim Schlüsselverwaltungsserver ein Liste von Schlüsseln. Die Antwort auf die Anfrage enthält jedoch nicht nur die Liste sondern auch einen sogenannten Zeitstempel. Durch den Zeitstempel erfährt der Knoten, zu welchem Zeitpunkt ein bestimmter

Schlüssel der Liste im synchronisierten Netzwerk eingesetzt werden soll. Das bedeutet aber, dass auch der Maschen-Router eine Art Schlüsselverwaltung haben muss. Die Auswahl des passenden Schlüssels aus der vorgehaltenen Liste wird durch eine Funktion in Abhängigkeit von der Zeit und dem Zeitstempel bestimmt.

„Client-Driven"-Verfahren

Anders als beim „Server-Driven"-Verfahren ist dieses Verfahren dafür gedacht, den Netzwerkverkehr zu reduzieren. Anstatt einer Liste von Schlüsseln wird daher nur ein Seed übertragen, aus dem der Maschen-Router selbst eine bestimmte Anzahl von Schlüsseln generieren darf. Die Berechnung dieser Schlüsselliste erfolgt auf Basis einer Hash-Funktion und dem übermittelten Seed.

Spezifikation des Nachrichtentyps „KDP-Request"

KDP-Request werden vom Client an den Server verschickt. Die Nachricht beinhaltet ein Typen-Feld, mit dem der Client die Art des Schlüsselmaterials bestimmt, und ein Feld mit der eigenen MAC-Adresse als Identifikator.

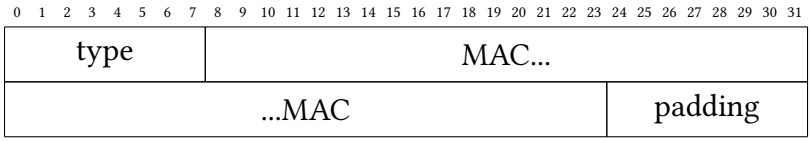

Abbildung 1: Aufbau einer KDP-Request-Nachricht

Spezifikation des Nachrichtentyps „KDP-Response"

Das Typ-Feld aus dem KDP-Request bestimmt die Form des KDP-Response. In einer simplifizierten Form existieren KDP-Responses für die beiden Modi „Client-Driven" und „Server-Driven".

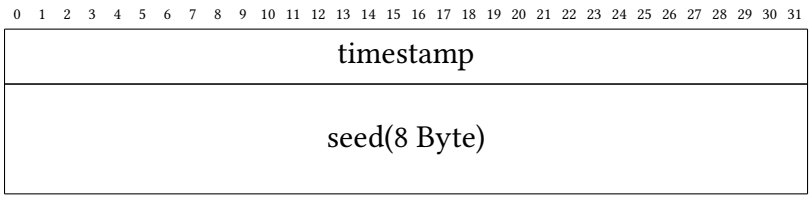

Abbildung 2: Aufbau einer KDP-Response-Nachrichten für die „Client-Driven"-Variante

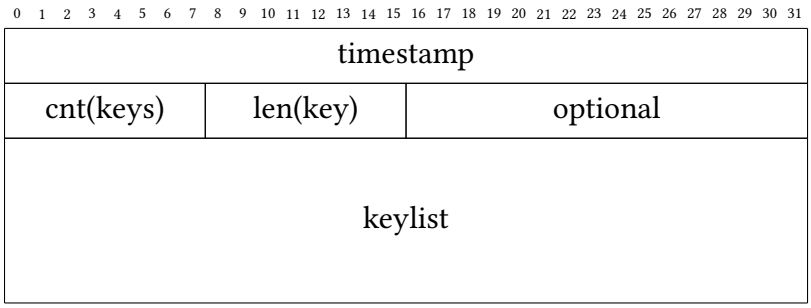

Abbildung 3: Aufbau einer KDP-Response-Nachrichten für die „Server-Driven“-Variante

2.5 Sicherheitsannahmen

Zuvor habe ich versucht die konstruktiven Elemente des Sicherheitsmodells zu erläutern. Ich möchte nun einige Sicherheitsannahmen der Autoren explizit machen, ohne die das Sicherheitsmodell nicht auskommt.

Ich beziehe mich als erstes auf das in Abschnitt 2.2 auf Seite 10 angesprochene Problem des „Single-Point-of-Failure“. Dort habe ich bereits zwei Angriffe auf den SA-Server gegenübergestellt. Dieses Problem wird von den Autoren nur insofern behandelt, als dass sie die folgende Sicherheitsannahme treffen:

Annahme 1 Es gibt mindestens einen manipulationssicheren Maschenknoten, der sich aus Sicht der anderen Knoten erwartungsgemäß korrekt verhält. Das bedeutet, dass er nur die festgelegten Dienste gemäß den zugrunde liegenden Protokollen ausführt. Das ist der SA-Server.

Der nächste Punkt betrifft den in Abschnitt 2.3.2 auf Seite 14 angesprochene Zertifikataustausch. Damit Zertifikate zum Einsatz kommen können, muss es ein Verwaltungssystem für Zertifikate geben; im Fall von MobiSEC handelt es sich um eine PKI, bestehend aus Zertifizierungsstelle, Zertifikaten (CA) und Menschen. Daraus ergeben sich für die Autoren zweierlei Sicherheitsannahmen:

Annahme 2 Sofern Annahme 1 zutrifft, basiert die Authentizität der Maschen-Knoten in der Theorie nur noch auf der Unterschrift und Einzigartigkeit der Zertifizierungsstelle (CA). In dem die CA die Zertifikate der Maschen-Knoten

unterschreibt, etabliert sie eine nachweisbare und überprüfbare Identität für die Netzwerkteilnehmer und wird zum Vertrauensanker des gesamten Netzwerkes.

Annahme 3 Die PKI wird von einem Menschen oder einer Organisation aufgebaut und gewartet, welche die Verantwortung dafür übernimmt.

Das Besondere an diesen drei Aussagen ist, dass sie nicht aus dem Protokoll ableitbar sind, sondern durch eine sogenannte Sicherheits-Policy von außen, im Vorhinein und währenddessen durchgesetzt werden müssen.

3 Theoretische Analyse des Protokoll-Overheads

Mit dem Beginn dieses Kapitels werde ich nun auf die von mir untersuchte zweite Phase des Sicherheitsprotokolls eingehen. Diese zweite Phase ist besonders interessant, da sie auf Grund der Funktionsweise des KDPs in periodischen Abständen wiederholt wird, um letzten Endes die Übertragung der eigentlichen Kommunikationsdaten zwischen den Netzwerkteilnehmern zu gewährleisten. Daher bezeichne ich in diesem Abschnitt die dabei anfallenden Daten, im Gegensatz zu den Kommunikationsdaten, als Overhead.

Meine erste hier vorgenommene Untersuchung soll eine Abschätzung des Overhead zwischen zwei direkt miteinander verbundenen Knoten erlauben (One-Hop-Szenario). Ich verwende dabei die Prämisse, dass ein kleiner Overhead weniger Kollisionen auf dem Medium hervorruft und sich nur geringfügig auf den Datendurchsatz auswirkt. Indem ich nun analytisch zeige, dass der Overhead nur sehr gering ausfällt, habe ich Grund zur Annahme, dass sich dies auch empirisch zeigen lässt.

Die Phase zwei des Sicherheitsprotokolls verwendet zwei Protokolle: Zum einen das TLS-Protokoll, um eine sichere Verbindung aufzubauen und zum zweiten das KDP für die Versorgung des Backbone-Netzes mit Schlüsseln. TLS, oder genauer gesagt das TLS Record Protokoll, verwendet für den Betrieb drei Sub-Protokolle – Change-Cipher-Spec-Protokoll, Alert-Protokoll, Handshake-Protokoll, Application-Data-Protokoll – von denen allein das Handshake-Protokoll eine signifikante

Netzwerklast generiert und deshalb hier untersucht wird. In meiner Analyse habe ich für TLS zwischen *Session Setup Overhead* und *Session Resume Overhead* unterschieden. Der letzte Fall stellt eine Optimierung des Session Setups dar. Dabei arbeiten die Kommunikationspartner mit Session-IDs und sparen damit den Austausch von Zertifikaten ein.

Aufbauend auf dieser TLS-Unterscheidung habe ich die Netzwerklast des KDP untersucht und dabei zwischen „Server-Driven“ und „Client-Driven“ unterschieden.

Zur Berechnung der Netzwerklast des Schlüsselverteilungsprotokolls habe ich die folgenden Formeln verwendet:

$$\begin{aligned} kdpOverhead(serverDriven) &= (macLength + modeLength + keysInList \cdot \\ &\quad keyLength + timestampLength) \\ kdpOverhead(clientDriven) &= (macLength + modeLength + seedLength + \\ &\quad timestampLength) \end{aligned}$$

Für das KDP aus Martignon et al. (2009) habe ich gängige WEP-Schlüssellängen für $keyLength$ und UNIX-Timestamps für $timestampLength$ verwendet. Die verwendeten Werte für die TLS-Records stammen sowohl aus der Spezifikation von Dierks und Allen (1999) für TLSv1.0 als auch aus den Analysen und Messungen von Apostolopoulos et al. (1999). Das TLS-Hand-Shake-Protokoll schlägt mit zwei Round Trips zu Buche, das KDP mit nur einem, in der Summe also drei Roundtrips.

Das Ergebnis des One-Hop-Szenarios ist nachfolgend in einer Tabelle zusammenfassen. Weitere Protokolle wie bspw. TCP und 802.11 wurden für die Abschätzung außen vorgelassen.

Ausgangsparameter

Schlüssel in Liste:	12
Timeout für einen Schlüssel:	15 Sekunden
Dauer der Session:	180 Sekunden

Netzwerklast des KDP über TLS (One-Hop-Overhead)

	KDP Server-Driven	KDP Client-Driven
TLS-Session Setup	2190 Byte	2134 Byte
TLS-Session Resume	1228 Byte	1172 Byte
Datenrate (pro Session)[16]	13 B/s	12 B/s

Diese Ergebnisse stellen je nach Konfiguration von TLS eine untere Grenze dar. Übertragungsfehler können daher zu einem Anstieg dieser Werte führen. Gemessen an der Dauer einer Session (siehe Datenrate), fallen diese Werte jedoch kaum ins Gewicht. Doch wie steht es um größere Netzwerke, in denen alle Maschen-Router innerhalb eines bestimmten Zeitraums eine Anfrage an den SA-Server stellen?

Ausgehend von einer Verkehrsanalyse zwischen zwei direkt miteinander verbundenen Knoten, ist die Netzwerklast für komplexere Topologien auch analytisch bestimmbar. Da das Ziel jeder Anfrage der SA-Server ist, kann man sich vorstellen, dass durch das Routing-Protokoll graphentheoretisch ein gerichteter Baum entsteht. Dazu folgende Definition:

Sei $G = (V, E)$ ein gerichteter Baum mit dem SA-Server als Wurzel, in dem die Kanten in Richtung Wurzel zeigen, $r \in V$ ein dedizierter Relay-Knoten (Maschen-Router oder SA-Server) und P die Menge aller in G vorhandenen Pfade.

$$NodeOverhead(r) = oneHopOverhead \cdot \sum_{\substack{\exists q \in V: \\ (q,r) \in P, \\ q \neq r}} |(q, r)|$$

Setzt man für r den SA-Server ein und für den $oneHopOverhead$ einen der Werte aus der oberen Tabelle, so erhält man den gesamten Overhead, der für die Verteilung des Schlüsselmaterials einer Session entsteht. In Abbildung 4 ist ein Graph mit neun Knoten dargestellt. In einem Szenario mit diesen neun Knoten, bei dem das TLS-Protokoll im Session-Setup-Modus und der KDP im Server-Driven-Modus betrieben wird, beträgt die Gesamtnetzwerklast am SA-Server ca. 41 Kilo-

[16] Die Datenrate ist auf Basis des TLS-Session Setups berechnet worden.

bytes pro Session. Gemessen am üblichen Netzwerkverkehr bspw. in Bürogemeinschaften ist dieses Ergebnis durchaus akzeptabel.

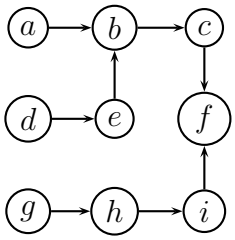

Abbildung 4: Gerichteter Baum mit f als Wurzelknoten.

4 Software-Entwurf: Aufbau und Funktionsweise

In diesem Kapitel soll es darum gehen, den Aufbau und die Funktionsweise meines Software-Entwurfs zu erläutern. Es handelt sich dabei um die konkrete Umsetzung der zweiten Phase des MobiSEC-Sicherheitsprotokolls bzw. der Backbone-Sicherheit. Im Einzelnen möchte ich dabei einerseits Entwurfsprinzipien und Entwurfsentscheidungen zur Geltung bringen und andererseits Erfahrungen über Problemvorkommen und Unzulänglichkeiten darlegen.

Die Grundlage für die Implementierung bietet das Click-Framework, eine Software-Architektur für das Bauen von flexiblen und konfigurierbaren Routern.[17]

4.1 Click-Netzwerkarchitektur

Im Click-Framework wird jeder Netzwerkstack eines Netzwerkteilnehmers durch ein Click-Skript repräsentiert. Ich erinnere noch einmal daran, dass es in dem hier behandelten Sicherheitsprotokoll drei Akteure gibt: Maschen-Klient, Maschen-Router und den SA-Server. Ich werde als ersten auf den Maschen-Router eingehen.

[17] Siehe Kohler et al. (2000).

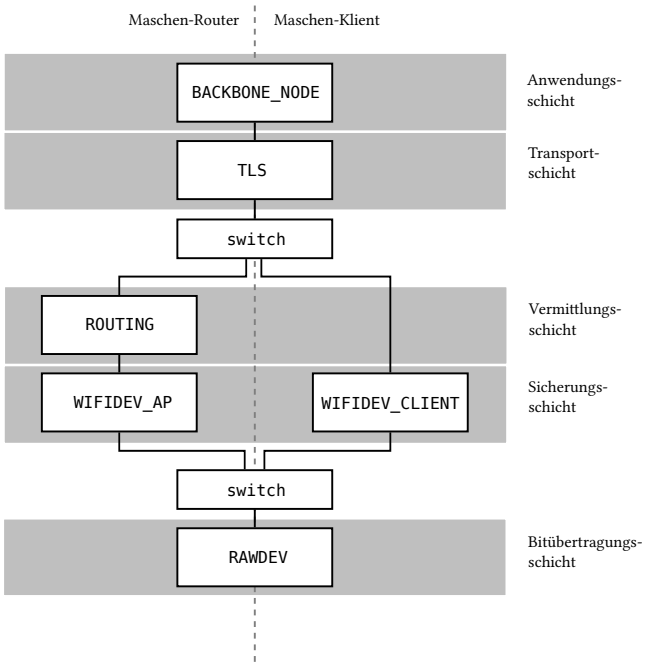

Abbildung 5: Netzwerkstack des Maschen-Knoten

Maschen-Router

Wie schon in 2.3 beschrieben muss jeder Maschen-Router, bevor er als solcher seine Rolle im Backbone-Netz ausüben kann, sich als Maschen-Klient gegenüber dem SA-Server authentisieren. Diese Doppelrolle spiegelt sich im Netzwerkstack eines Maschen-Routers wider. Um diese Doppelrolle besser diskutieren zu können, werde ich im Rahmen dieses Unterabschnitts von einem Maschen-Knoten sprechen, der je nach Protokollzustand entweder die Rolle eines Maschen-Routers oder eines Maschen-Klienten ausübt. Abbildung 5 zeigt in Anlehnung an das OSI-Schichtenmodell, wie diese Doppelrolle in Click umgesetzt ist.

Bedingt durch die HWL-Netzwerkarchitektur und die zum Einsatz kommenden Geräte verfügt der Maschen-Knoten nur über ein Interface, das in der Abbildung auf der Bitübertragungsschicht durch das Click-Element `RAWDEV` dargestellt ist.

Augenfällig ist nun die Konstruktion in der Sicherungsschicht und Vermittlungsschicht, die von zwei `switch`-Elementen umschlossen ist. In dieser Konstruktion spiegelt sich im Wesentlichen die Doppelrolle wider. Die `switch`-Elemente

aus dem Click-Framework erlauben es mir mittels einer Kontrollfunktion, Paket-Ströme innerhalb des Netzwerkstacks zu beeinflussen. Je nach Schaltung nehmen die Pakete entweder den Weg über das `WIFIDEV_CLIENT`-Element oder den anderen Weg über die Elemente `WIFIDEV_AP` und `ROUTING`. Zu keinem Zeitpunkt jedoch wird das `switch`-Element beide Wege zulassen. Diese Konstruktion eines zweigeteilten Netzwerkstacks bezeichne ich fortlaufend als „Switch-Stack“, da entweder der eine oder der andere Teil des Stacks zum Einsatz kommt. Kontrolliert wird dieser Switch-Stack vom Click-Element in der Anwendungsschicht.

Als Maschen-Klient muss der Maschen-Knoten über eine Netzwerkschnittstelle verfügen, die im Infrastrukturmodus gemäß dem IEEE 802.11-Protokoll arbeitet und zudem WEP-verschlüsseln kann. Dies erledigt das Click-Element `WIFIDEV_CLIENT`.

Da die Adressierung im HWL-Click aus Gründen der Effizienz ausschließlich über die Ethernet-Adresse erfolgt, ist im Netzwerkstack des Maschen-Klienten kein Element für die Vermittlungsschicht notwendig.

Die Transportschicht enthält zur Zeit lediglich das Click-Element `TLS`, da zum Zeitpunkt der Entwicklung ein TCP-Modul im Click-Framework des HWL noch nicht vorhanden war. Während der Simulationen zeigte sich jedoch, dass eine zuverlässige Transportschicht unerlässlich ist, und zwar aus zwei gewichtigen Gründen: Erstens verändert jeder Rollenwechsel eines Maschen-Knotens – wenn auch minimal – die Netzwerktopologie und beeinflusst damit das Routing-Protokoll. Zweitens ist das geteilte Medium inhärent störanfällig (z. B. auf Grund des Hidden-Node-Problems[18]).

Das `TLS`-Element ist ein wichtiger Bestandteil des MobiSEC-Sicherheitsprotokolls, ohne jedoch Teil des Click-Frameworks zu sein. Zwar lässt sich OpenSSL in der Version 0.9.8 mit einem TCP-Socket mehr oder weniger leicht betreiben, allerdings wäre eine Integration in den Click-Netzwerkstack nicht möglich gewesen. Ich entschied mich daher für den Aufwand, ein eigenes Click-Element als Wrapper zu entwickeln. Anstatt eines TCP-Sockets habe ich einen von mir kontrollierten Puffer als OpenSSL-BIOs verwendet, um die TLS-Records selbst zu verpacken und für den Click-Netzwerkstack verarbeitbar zu machen.

[18] Vergleiche zum Beispiel Goldsmith (2005).

Das Click-Element `BACKBONE_NODE` ist das Herzstück des MobiSEC-Maschen-Routers. `BACKBONE_NODE` nutzt nun das Click-Element `TLS`, um das KDP über eine sichere Verbindung ausführen zu können.

Betrachten wir nun den Maschen-Knoten in seiner Rolle als Maschen-Router. Im Switch-Stack treten an die Stelle der Netzwerkschnittstelle `WIFIDEV_CLIENT` nun die beiden Click-Elemente `WIFIDEV_AP` (Sicherungsschicht) und `ROUTING` (Vermittlungsschicht) ein. Erst durch sie ist der Maschen-Knoten in der Lage, die Funktionen eines APs auszuführen, einerseits, und andererseits den Routing-Dienst innerhalb des Backbone-Netzes zu übernehmen. Als Routing-Protokoll verwende ich das Dynamic Source Routing (kurz: DSR).

SA-Server

Der SA-Server besitzt im Grunde genommen den gleichen Aufbau wie der Maschen-Knoten, sofern er sich als Maschen-Router verhält. Statt dem `BACKBONE_NODE` kommt nun das Click-Element `KEYSERVER` zum Einsatz, dessen wichtigste Aufgabe es ist, die Maschen-Router mit dem notwendigen Schlüsselmaterial zu versorgen.

4.2 Protokollablauf

Kommen wir nun zur technischen Realisierung des Protokollablaufs. Eingangs eine kleine Vorbemerkung bezüglich der ausgelassenen ersten Phase des Sicherheitsprotokolls.

Vorbemerkung

In dieser Arbeit nehme ich explizit an, dass das 802.11i-Protokoll transparent ausgeführt wird. Das heißt, dass eine Netzwerkschicht nichts von den Vorgängen auf der darunterliegenden Schicht mitbekommt. Doch kann ich das wirklich tun? Zwei Situationen sind in dieser Hinsicht interessant: (1.) Unter welchen Bedingungen wird die zweite Phase in Gang gesetzt (Bootstrapping) und (2.) was passiert bei einem Kommunikationsabbruch, der im schlimmsten Fall mit einer daran anschließenden neuen *Assoziierung* bzw. *Reassoziierung* auf Medium-Access-Control-Ebene verbunden ist? Auf beide Fragen möchte ich eine Antwort geben.

1. Wie in IEEE (2004a) Abschnitt 5.4.2.2 beschrieben, baut 802.11i im Zuge einer Assoziierung eine *Robust Security Network Association* (kurz: RSNA) auf. Es handelt sich bei dem RSNA-Konzept um einen speziellen Typ von Assoziierung, dem bestimmte Sicherheitsprozeduren vorausgegangen sind (wie in 2.3.1 beschrieben). Über diese RSNA bestimmt ein Maschen-Router als AP, ob ein bestimmter Maschen-Klient für die Benutzung des unkontrollierten Port autorisiert ist und damit Zugang zum Netzwerk erhält.
   Diese Eigenschaft nutze ich in meiner Arbeit, indem ich von jeder 802.11i-RSNA auf eine normale 802.11-Assoziierung abstrahiere.

2. Längere Kommunikationsabbrüche können zur Folge haben, dass Teile des Backbone-Netzes in vereinzelte Maschen-Knoten zerfallen, von denen einige Maschen-Router verbleiben, andere jedoch in den unprivilegierten Status eines Maschen-Klienten zurückwechseln. Für eben jene Maschen-Klienten können zwei Fälle eintreten. Entweder erreicht der Maschen-Klient seinen ehemaligen AP und hat die Möglichkeit, sich mit ihm zu reassoziieren. Oder erreicht einen neuen AP des MobiSEC-Backbone-Netzes und beginnt eine neue Assoziation. In beiden Fällen jedoch schreibt der IEEE 802.11i-Standard vor, dass eine neue RSNA erstellt werden muss. Damit aber ist die zweite Frage auf die erste Frage reduzierbar.

Innerhalb meines Software-Entwurfs steht die Assoziierung auf MAC-Ebene also für eine erfolgreiche Abwicklung des 802.11i-Protokoll und ermöglicht es mir, mich auf die zweite Phase zu konzentrieren.

4.2.1 Ablauf eines Rollenwechsels

Wenden wir uns nun dem Aspekt des Rollenwechsels genauer zu, der ja in gewisse Weise konstitutiv für das Backbone-Netz ist. Im letzten Abschnitt 4.1 habe ich den Aufbau des Netzwerkstacks beschrieben. Hier soll es unter anderem um ihre Benutzung gehen. Der Rollenwechsel ist nun folgendermaßen umgesetzt:

1. Ein KDP-Request wird von der Client-Anwendung (`BACKBONE_NODE`) vorbereitet, welches eine Request-ID und die Ethernet-Adresse des Absenders

enthält. Die Server-Anwendung kann über diese beiden Daten später eine rudimentäre Accounting-Funktion ausüben. Als nächstes übergibt die Client-Anwendung den KDP-Request an das TLS-Click-Element.

2. Falls keine TLS-Session vorhanden ist, führt das Click-Element `TLS` eine gegenseitigen Authentifizierung aus. Die Pakete aus der Anwendungsschicht werden solange gepuffert, bis die TLS-Verbindung erfolgreich aufgebaut ist.

3. Nachdem der KDP-Request abgeschickt worden ist, erwartet der Maschen-Router ein KDP-Response. Dieses Response-Paket enthält einen Zeitstempel, Schlüsselanzahl und wahlweise eine Schüsselliste oder einen Seed. Diese Informationen werden in einer Puffer-Struktur abgespeichert und ggf. eine Schlüsselliste aus dem Seed erstellt.

4. Der Maschen-Router führt nun eine Stack-Switch-Routine aus. Diese beinhaltet, dass der Maschen-Router sich ordnungsgemäß in seiner alten Rolle als Maschen-Klient am AP abmeldet (Disassoziierung) und dann den Netzwerk-Stack umschaltet.

5. Ein zeitgesteuerter Mechanismus verwendet den Zeitstempel und die Schlüsselliste und berechnet den Zeitpunkt ihrer Einsetzung (siehe dazu den nachfolgenden Abschnitt 4.2.3). Der Maschen-Router ist nun bereit für die Teilnahme am Backbone-Netz.

Sobald sich die Information über den neuen Maschen-Router durch das Routing-Protokoll im ganzen Backbone-Netz verbreitet hat, kann er sein vollständiges Potential als AP und Router entfalten.

Als problematisch hat sich im vierten Schritt vor allem die korrekte Ausführung des TLS-Protokolls erwiesen. Die im Click-Element `TLS` verwendete OpenSSL-Engine benutzt einen Zustandsautomaten, um die Protokollsicherheit zu gewährleisten. Möchte man eine TLS-Verbindung beenden, muss eine „Shutdown“-Routine gestartet werden, die üblicherweise einen Informationsaustausch zwischen Klient und Server erfordert. Ein Stack-Switch kann jedoch den Empfang dieser Shutdown-Nachrichten konterkarieren und damit die OpenSSL-Engine des Clienten und des Servers desynchronisieren.

Um dieses Problem zu umgehen, verwende ich die von der OpenSSL-Engine angebotene Möglichkeit des „Quiet Shutdowns“: Die OpenSSL-Engine wird mittels eines Flags in den finalen Zustand gesetzt, ohne dabei weitere Nachrichten zu versenden. Das setzten dieses Flags geschieht im Software-Entwurf (in den meisten Fällen) auf Seiten des Servers genau nach dem Erhalt des KDP-Request und auf Seiten des Klienten nach dem Erhalt des KDP-Response.

4.2.2 Sicherheitsparameter

Der vom Sicherheitsprotokoll geleitete Kommunikationsprozess wird von einigen Sicherheitsparametern bestimmt, die vor der Inbetriebnahme des WMNs festgelegt werden müssen. Die Auslese dieser Parameter unter der Vielzahl automatisierbarer (bzw. voreinstellbarer) Konfigurationsmöglichkeiten erfolgte aufgrund der Überlegung, dass die Routing-Leistung einerseits und der Sicherheitsgrad andererseits sich gegenseitig (negativ) bedingen. Das optimale Verhältnis dieser beiden Dimensionen zu bestimmen, je nach Umweltbedingungen, Netzwerkszenario usw., ist daher dem abwägenden Systemadministrator anheim gestellt.

Client-Driven vs. Server-Driven
: Die gegenseitige Bedingtheit tritt bei diesem Parameter am offensichtlichsten zu Tage, da die Autoren bewusst das Protokoll so entworfen haben, dass die Schlüsselliste im Server-Driven-Modus eine starke informationstheoretische Unabhängigkeit aufweisen sollen, andererseits jedoch zu einer höheren Netzwerklast führen können. Im anderen Modus, so die Vermutung, kehrt sich dieses Verhältnis tendenziell um.

Schlüsselanzahl Die Schlüsselanzahl ist ein Maß für den Vorrat mit gültigem Schlüsselmaterial, über den ein Maschen-Router verfügen soll. Je größer die Zahl tendenziell ist, desto unabhängiger ist er vom SA-Server.

Sitzungslänge Die Sitzungslänge entspricht dem Zeitraum, in dem ein bestimmter Schlüssel seine Gültigkeit besitzt. Je niedriger ihr Wert tendenziell ist, desto unwahrscheinlicher ist zwar ein erfolgreicher Kyptoangriff, aber desto höher ist auch der Verbrauch an Schlüsselmaterial. Letzteres hat unmittelbar eine höhere Netzwerklast zur Folge.

4.2.3 Epochen- und Sitzungsmechanismus

Die Teilnahme am Backbone-Netz setzt voraus, dass erstens alle Maschen-Knoten zeitlich synchronisiert sind, zweitens gleichzeitig den gleichen Gruppenschlüssel verwenden und drittens nach einer bestimmten Zeit den alten Gruppenschlüssel durch einen neuen ersetzten. Wie diese Übergänge von statten gehen, soll nun hier erklärt werden.

Üblicherweise erhält jeder Maschen-Router nach einem KDP-Request das nötige kryptographische Material, um eine Schlüsselliste abzuspeichern. Aus dem Zeitstempel, der Länge einer Sitzung und der Anzahl der Schlüssel in dieser Liste ergibt sich die zeitliche Gültigkeit der Schlüsselliste. Diesen Zeitraum bezeichne ich als *Epoche*. Diese Gliederung spiegelt sich in Abbildung 6 wider, auf die ich nun näher eingehen möchte.

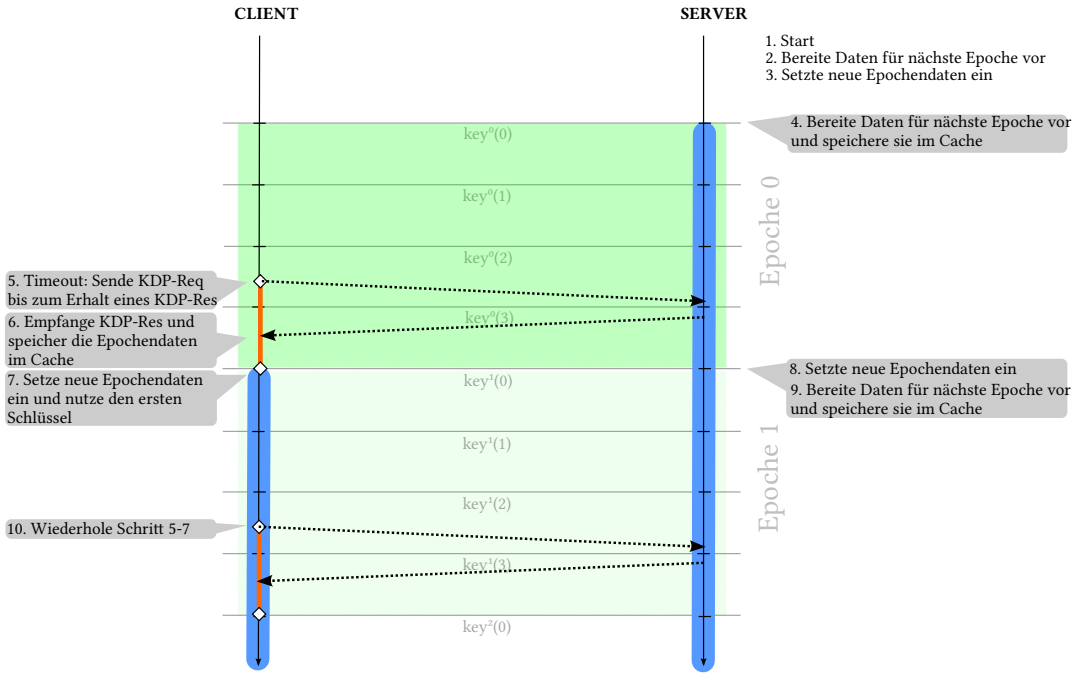

Abbildung 6: Epochen- und Sitzungsmechanismus

Die Schritte 1-3 in der Abbildung beschreiben den Bootstrapping-Prozess, bei dem der SA-Server das kryptographische Material für die folgende Epoche 0 er-

zeugt und es kurz vor Beginn dieser Epoche bei sich selber einsetzt. In Schritt 4 bereitet er das kryptographische Material für die darauf folgende Epoche 1 vor.

In meinem Software-Entwurf habe ich mich dafür entschieden, das zeitgesteuerte Einholen der Schlüsselliste (KDP) zu trennen von der zeitgesteuerten Einsetzung der Schlüssel. Das bedeutet, dass es auf der einen Seite zwei zeitgesteuerte Funktionen gibt, die sich um den Epochenwechsel und den Sitzungswechsel kümmern, und zwar unabhängig davon, ob nun eine gültige Schlüsselliste vorhanden ist oder nicht. Ist eine gültige Schlüsselliste vorhanden, so wird diese sofort für die nächste Sitzung verwendet. Die grünen und hellgrünen Flächen markieren die fortlaufenden Epochen- und Sitzungswechsel.

Auf der anderen Seite erwirkt eine zeitgesteuerte Funktion die Ausführung des KDP, um die Schlüsselliste rechtzeitig vor dem Beginn der nächsten Epoche empfangen zu können (siehe Schritt 5-7). Ansonsten kann es passieren, dass der betroffene Maschen-Router den Netzwerkverkehr nicht mehr entschlüsseln kann und erneut die beiden Phasen durchlaufen muss. Dadurch entstehen hohe Kosten: Zum einen der Wechsel im Switch-Stack im Maschen-Router und zum anderen die Veränderung der Netzwerktopologie, die mit weiteren Rechenaufwand für das Routing-Protokoll behaftet ist. Schritt 8-9 laufen analog zu Schritt 3-4.

Der orange Bereich zeigt den Vorlauf zur nächsten Epoche an, in der die Maschen-Router Zeit haben, das zukünftige Schlüsselmaterial rechtzeitig einzuholen. Danach sind die Maschen-Router gezwungen einen Rollenwechsel zum Maschen-Klienten zu vollziehen und das Sicherheitsprotokoll von vorn zu beginnen.

4.3 Innerer Aufbau & Zusammenwirken

Im letzten Abschnitt dieses Kapitels möchte ich den Fokus auf die beiden Click-Elemente `BACKBONE_NODE` und `KEYSERVER` richten. Sie bilden gewissermaßen das Herzstück des Sicherheitsprotokolls, da sie nicht nur das KDP ausführen, sondern auch einige weitere Elemente koordinieren und in sich integrieren. Ich beginne wieder mit dem `BACKBONE_NODE` und zeige im Anschluss die Unterschiede auf.

4.3.1 Click-Element BACKBONE_NODE

Das Click-Element BACKBONE_NODE besteht aus drei Komponenten: backbone_node, keymanagement und *kdp*, die jeder für sich eine bestimmten Zweck erfüllen. Eine wichtige Entwurfsentscheidung bestand darin, dem backbone_node die zentrale Koordinierungs- bzw. Kontrollfunktion zuzuweisen, da ein autonomes Schlüsselverteilungsprotokoll und ein autonomes System für die Schlüsselverwaltung Synchronisationsprobleme hervorrufen würden, die ich für schwerer zu lösen hielt. Aus diesem Grund bieten die Module keymanagement und kdp lediglich Speicherstrukturen und Schnittstellen mit synchronen Return-Funktionen an. Nun zu den Modulen im einzelnen.

Das keymanagement-Modul stellt eine instanziierbare Klasse dar, die dazu da ist,

- das Schlüsselmaterial für eine Epoche in einer entsprechenden Datenstruktur vorzuhalten,
- optional aus dem Seed eine Schlüsselliste zu generieren und
- den richtigen Schlüssel zum entsprechenden Zeitpunkt zu installieren.

Das kdp-Modul liefert die für das Protokoll wichtigen Protokollpakete KDP-Request und KDP-Response. Das backbone_node-Modul integriert diese beiden Module und kümmert sich

- um den Empfang und den zeitgesteuerten Versand der KDP-Pakete,
- das Verarbeiten der KDP-Pakete,
- die zeitgesteuerten Übergänge von Sitzungen und Epochen mittels Timeouts und Scheduling-Algorithmen und
- um den Stack-Switch.

Zusätzlich kommuniziert das backbone_node-Modul mittels sogenannten Handlern[19] aus den verschiedensten Gründen mit den Click-Elementen TLS, WIFIDEV_CLIENT, WIFIDEV_AP, SWITCH und WEP. Die Verknüpfung zum WEP-Element ist

[19] Siehe http://www.read.cs.ucla.edu/click/element?s[]=handler.

notwendig, um die WEP-Gruppenschlüssel auszutauschen. Die Verknüpfung zum TLS-Element ist maßgeblich durch die fehlende Flusskontrolle bedingt (siehe 4.2.3). Der Stack-Switch-Mechanismus benötigt wiederum eine Verbindung mit den WIFIDEV_CLIENT, WIFIDEV_AP, SWITCH.

4.3.2 Click-Element KEYSERVER

Das Click-Element KEYSERVER besteht auch aus drei Komponenten: keyserver, keymanagement und kdp. Die Funktionsweise des keyserver-Modul ist die folgende: Es kümmert sich

- um den Empfang und den zeitgesteuerten Versand der KDP-Pakete,
- das Verarbeiten der KDP-Pakete,
- die zeitgesteuerten Übergänge von Sitzungen und Epochen mittels Timeouts und Scheduling-Algorithmen und
- berechnet das Schlüsselmaterial für die kommenden Epochen.

Zwar kommuniziert das keyserver-Modul, ähnlich wie das backbone_node-Modul, mit den Click-Elementen TLS und WEP. Ein Switch-Stack ist jedoch nicht nötig, da er von vornherein als AP aufgestellt ist und damit das Bootstrapping besorgt.

5 Simulation & Evaluation

In ihrem Paper haben die MobiSEC-Autoren eine Simulationsstudie gemacht, auf die ich mich im Folgenden beziehen werde. Relevant für meine Arbeit ist das in ihrer Simulationsstudie betrachtete Netzwerk mit Grid-Topologie, bei dem 30 Knoten auf einem Feld von 1000 m $\times$1000 m angeordnet sind. Die Knoten haben jeweils einen horizontalen Abstand von 200 m und einen vertikalen Abstand von 250 m. Die Kanalkapazität wurde auf 54 Mbit/s gesetzt. Alle Knoten verwenden, bedingt durch den beschränkten Funktionsumfang des NS2 (kein Multi-Channel oder Multi-Interface möglich), eine WLAN-Schnittstelle und überall den gleichen Kanal. Als Routing-Protokoll wird das UM-OLSR verwendet.

Der Zweck ihrer Simulation ist eine Durchsatzmessung einer TCP-Verbindung zwischen den zwei am weitesten entfernten Knoten. Die Experimente zeigen zwar gute Durchsatzraten, aber sie verraten wenig über die gesamte Verfügbarkeit des Backbone-Netzes. Man kann zumindest annehmen, dass das Sicherheitsprotokoll von den dazwischenliegenden Maschen-Routern vollständig durchgeführt wurde.

Um die Frage der Verfügbarkeit und Skalierbarkeit des Backbone-Netzes soll es in diesem Kapitel gehen.

Für die Evaluierung der von mir implementierten zweiten Phase des Sicherheitsprotokolls habe ich ebenfalls den NS2 verwendet. Alle Simulationen erstreckten sich über einen Zeitraum vom 1600 Sekunden. Die Maschen-Router verwendeten in jeder Epoche 12 Schlüssel mit einer Gültigkeit von 15 Sekunden.

Als Radiomodell verwende ich das von NS2 bereitgestellte „Two-ray ground reflection model“.[20] Die Knoten senden mit 1 Mbit/s und benutzten ebenfalls eine WLAN-Schnittstelle und den gleichen Kanal. Für das Routing verwende ich das Dynamic-Source-Routing-Protokoll (kurz: DSR-Protokoll). Aufgrund des Determinismus des NS2 habe ich auf eine statistische Auswertung verzichtet und lediglich einzelne Experimente durchgeführt.

Meine Betrachtungen beschränken sich im Grunde auf drei Szenarien mit Netzwerktopologien vom Typ Grid, und zwar mit verschiedenen quadratischen Extensionen: 9, 16 und 25 Knoten. In jeder dieser Szenarien nimmt der SA-Server eine zentrumsnahe Position ein (siehe beispielhaft Abbildung 7 mit 9 Knoten und Knoten 0 als SA-Server). Alle Knoten habe dabei einen horizontalen und vertikalen Abstand von ca. 180 m gegenüber ihren Nachbarn.

Die Verfügbarkeit des Backbone-Netzes bestimmt sich durch die Anzahl der Knoten, die zu einem bestimmten Zeitpunkt einen gültigen Schlüssel besitzen. Besitzen zum Beispiel fünf von zehn Konten zu einem bestimmten Zeitpunkt einen gültigen Schlüssel, so beträgt die Verfügbarkeit des Backbone-Netzes zu diesem Zeitpunkt 50%.

Mit dem Start im NS2 sind bereits alle Knoten zeitlich synchronisiert. Nachdem jeder Knoten den Bootvorgang hinter sich gelassen hat, senden die Maschen-Router ihre ersten Anfragen an den SA-Server und der Aufbau des Backbone-

[20] Fall und Varadhan (2008, S. 190)

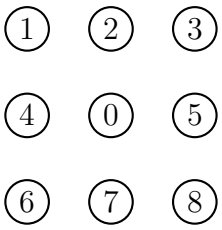

Abbildung 7: Die im NS2 verwendete Grid-Topologie

Netzes nimmt seinen Lauf.

Evaluation

Die Ergebnisse des ersten Experimentes kommen im ersten Diagramm von Abbildung 8 zum Ausdruck (links oben). In diesem Diagramm ist ein Koordinatensystem abgebildet, an dessen x-Achse die Zeit und an dessen y-Achse das Verhältnis der verfügbaren Knoten zur Gesamtzahl abgetragen ist. Darin sind zwei Stufenfunktion als Graphen abgebildet, welche den gemessen Verlauf der Verfügbarkeit über die Zeit hinweg widerspiegeln. Beide Graphen stehen jeweils für die unterschiedlichen Betriebsarten des Sicherheitsprotokolls (Client-Driven, Server-Driven). Sie zeigen, dass die Verfügbarkeit des Protokolls in beiden Betriebsarten über die Zeit hinweg knapp 88% beträgt, da 8 von 9 Knoten am Backbone-Netz teilnehmen.

Die Experimente, die zum zweiten und dritten Diagramm der ersten Reihe geführt haben, liefen analog zum ersten Experiment ab und unterschieden sich nur im Parameter der Knotenanzahl. Dabei ist folgendes zu beobachten: Die Verfügbarkeit des Backbone-Netzes sinkt der Tendenz nach mit steigender Knotenanzahl. Auffällig ist auch, dass die Verfügbarkeit bei einem bestimmten Schwellwert in allen drei Diagrammen scheinbar verharrt.

Dies zeigt erst einmal, dass das implementierte Sicherheitsprotokoll, wenn auch unter starken Einschränkungen der Verfügbarkeit, lauffähig ist. Verglichen mit den zuvor erwähnten Ergebnissen der MobiSEC-Autoren sowie meinen analytischen Betrachtungen aus dem dritten Kapitel, stellt sich die Frage, ob noch andere Fak-

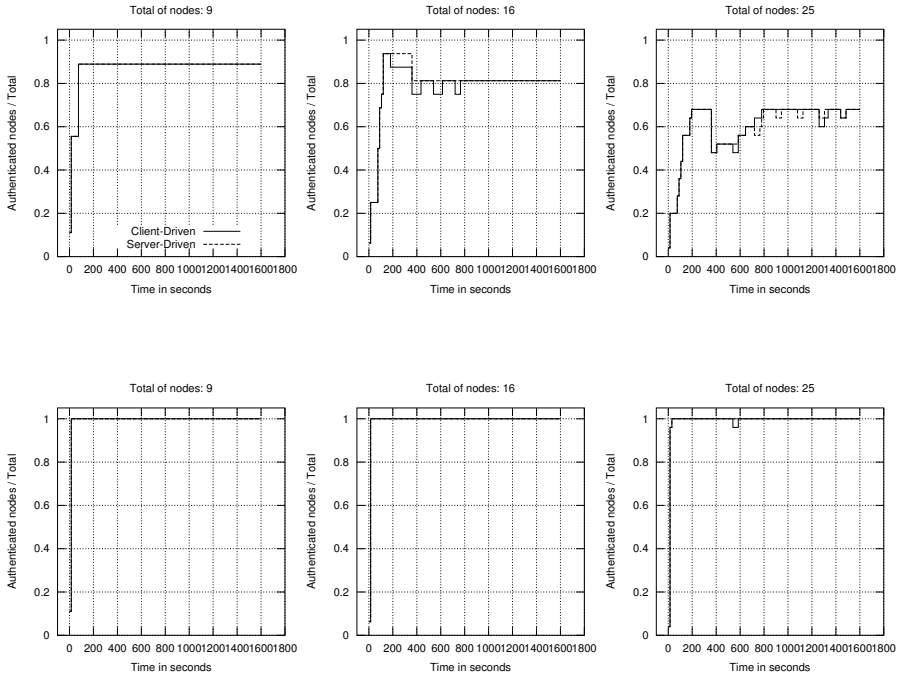

Abbildung 8: Erste Reihe: mit Switch-Stack. Zweite Reihe: ohne Switch-Stack.

toren neben den protokollinhärenten Eigenschaften einen besonderen Einfluss auf die Ergebnisse hatten.

Daher möchte ich an dieser Stelle zwei Faktoren diskutieren, die für die Verbesserung der Ergebnisse noch ausschlaggebend sein könnten: Erstens, eine fehlende Flusskontrolle (um die Kollisionen und Paketverluste zu vermeiden) und zweitens die Eigenheiten der DSR-Implementierung im HWL-Click-Framework.

Fehlenden Flusskontrolle

Üblicherweise gehen Pakete in drahtlosen Maschennetzwerken aufgrund von Kollisionen oder fehlerhaftem Routing verloren. Sowohl das TLS-Protokoll als auch das darauf aufbauende KDP benötigen aber ein zuverlässiges Transportprotokoll: Ein einziges fehlendes oder fehlerhaftes TLS-Paket hat eine Desynchronisierung

der OpenSSL-Engine von Maschen-Router und SA-Server zur Folge.

In Ermangelung eines HWL-Click-Moduls für die Flusskontrolle verfügt der Maschen-Router stattdessen über einen KDP-Retry-Mechanismus. Dieser läuft folgendermaßen ab: Sofern der Maschen-Router keine Antwort auf seine erste Anfrage bekommen hat, startet er eine bestimmte Anzahl von wiederholten Anfragen. Erhält er auch auf diese Anfragen keine Antwort, führt er einen Wechsel des Switch-Stacks aus. Letzteres geschieht in der Annahme, dass der Paketverlust über einen längeren Zeitraum einen Mangel an gültigen Schlüsselmaterial nach sich zieht und in der Folge keine Nachrichten mehr entschlüsselt werden können.

Dieser KDP-Retry-Mechanismus kann jedoch nicht als voller Ersatz für die fehlende Flusskontrolle herhalten, da seine Ausführung (aufgrund der Paketverlust-Annahme) auch einen TLS-Neustart erfordert und folglich mit viel Overhead verbunden ist.

Eine zusätzliche Maßnahme, um die fehlende Flusskontrolle zu kompensieren und Paket-Kollisionen zu vermeiden, besteht darin, die KDP-Requests der einzelnen Knoten zeitversetzt zu versenden.

Anhand der Log-Ausschriften des NS2 konnte ich beobachten, dass diese Maßnahmen ihre Wirkung zeigten. Trotzdem erreichten noch immer nicht alle Paketsendungen ihr Ziel, obwohl das Medium unbelegt erschien. Damit ergaben sich neue Hinweise im Bezug auf die Implementierung des DSR-Protokolls.

DSR-Protokoll

Während der Tests und Experimentvorbereitungen konnte ich zudem beobachten, dass der durch das Sicherheitsprotokoll hervorgerufene Wechsel des Switch-Stacks problematisch für das DSR-Protokoll sein könnte, da dieses keine vollständige Implementierung des Standards darstellte und damit gewisse Eigenheiten besaß. Ohne eine genauere Analyse angestellt zu haben, war bereits aufgrund einiger Log-Ausschriften erkennbar, dass bestimmte Kommunikationsprozesse, wie die Verbreitung von neuen Routing-Informationen, nicht korrekt durchgeführt wurden.

In der Folge habe ich in einer zweiten Experimentreihe den Switch-Stack vereinfacht, indem ich alle Maschen-Router nur noch als AP senden und empfangen ließ. Der Zweck dieser Maßnahme war es, den Wechsel zwischen Ad-Hoc- und

Infrastrukturmodus zu vermeiden und dem Routing-Protokoll zusätzliche Rechenaufwand zu ersparen.

Damit war die gesamte Kommunikation WEP-verschlüsselt. Zusätzlich wurden die TLS-Packete in gesonderter Weise verschickt, und zwar weder über das `WIFIDEV_AP` noch über das `WIFIDEV_CLIENT`-Element, sondern direkt über das `RAWDEV`-Element. Das Ganze entspricht – mit einigem Vorbehalt – in etwa einem Netzwerkmodell, bei dem die Maschen-Router zwei WLAN-Schnittstellen besitzen, und zwar eines für das Backbone-Netz und ein anderes für das Zugangsnetz.

Das Ergebnis lässt sich anhand der zweiten Reihe von Abbildung 8 ablesen. Bei diesen Experimenten bauen die Maschen-Router anfänglich relativ schnell ein Ad-Hoc-Netz und dann ein Backbone-Netz auf und schaffen es, das Schlüsselmaterial rechtzeitig einzuholen und die Epochen- und Sitzungsübergänge erfolgreich zu bestreiten. Die Verfügbarkeit beträgt mit steigender Knotenanzahl fast durchgängig 100%. Dies ist eine deutliche Verbesserung der Verfügbarkeit gegenüber der ersten Konfiguration.

Man kann daher vermuten, dass die aktuelle Implementierung des DSR-Protokolls tatsächlich Schwierigkeiten hat mit dem Wechsel zwischen Ad-Hoc-Modus (Maschen-Router) und Infrastrukturmodus (Maschen-Klient) .

Bezüglich des Experimentaufbaus mit Switch-Stack ist es sogar gut möglich, dass der TLS-Verkehr und der Routing-Verkehr sich aufgrund von Paketkollisionen gegenseitig negativ bedingen. Wird beispielsweise der TLS-Verkehr beeinträchtigt, kann dies einen Rollenwechsel vom Maschen-Router zum Maschen-Klienten verursachen. Dieser Wechsel bedeutet eine veränderte Netzwerktopologie, so dass neue Routing-Informationen gesammelt und berechnet werden müssen. Umgekehrt bedeutet eine veränderte Netzwerktopologie ein Risiko für die korrekte Zustellung von TLS-Paketen.

Fraglich bleibt insgesamt, ob dies ein protokollinhärentes Problem von DSR ist oder die Flusskontrolle das Zusammenspiel der verschiedenen Netzwerkprotokolle verbessern kann. Wenn man bedenkt, dass das DSR-Protokoll für den Einsatz in dynamischen Multi-Hop-Netzwerken entworfen wurde, kann man jedoch recht

zuversichtlich sein.[21]

Nichtsdestotrotz muss diese Vermutung über das Zusammenspiel mit dem DSR-Protokoll durch weitere empirische Untersuchungen erhärtet werden.

6 Schlussfolgerungen und Ausblick

Blicken wir zurück auf die in der Einleitung gestellte Frage, ob klassische Ansätze zentralisierter Sicherheitsarchitekturen und -Protokolle für WMNs zu Gunsten von dezentralisierten aufgegeben werden müssen, so ist folgendes zu ihrer Beantwortung zu sagen.

Mein Untersuchungsgegenstand bestand darin, das MobiSEC-Sicherheitsprotokoll in das HWL-Testbed zu integrieren und ihre praktische Eignung zu überprüfen und zu beurteilen. Damit bestand ein großer Teil des wissenschaftlichen Beitrags darin, die Experimentergebnisse der Autoren in einer anderen Umgebung zu reproduzieren und sie damit zu verifizieren, vor allem im Hinblick auf die Verfügbarkeit und Skalierbarkeit des Backbone-Netzes. Die Ergebnisse und Erkenntnisse sollen hier noch einmal kurz zusammengefasst werden.

Anhand der Ergebnisse der ersten Experimentreihe konnte ich zeigen, dass das umgesetzte Sicherheitsprotokoll in der derzeitigen Implementierung des HWL-Click-Frameworks nur unter starken Einschränkungen der Verfügbarkeit verwendbar ist. Wird der Wechsel zwischen Ad-Hoc-Modus und Infrastrukturmodus vermieden (d. h. kein Switch-Stack), zeigen die Ergebnisse des zweiten Experiments eine deutliche Verbesserung der Verfügbarkeit. Das Backbone-Netz konnte ohne größere Problem bis zu 25 Knoten skalieren. Zu den technischen Ursachen bezüglich der unterschiedlichen Messergebnisse zwischen der ersten und zweiten Experimentreihe habe ich zwei Vermutungen diskutiert:

1. Das Sicherheitsprotokoll ist bis zu einem gewissen Maß von einer zuverlässigen Transportschicht (Flusskontrolle) abhängig. Dies hat wiederum Auswirkungen auf die Netzwerktopologie.

[21] Dazu Johnson et al. (2007, S. 1):„The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes and is designed to work well even with very high rates of mobility.“

2. Das DSR-Protokoll besitzt in seiner derzeitigen Implementierung einige kleine Schwächen, die bei geringen Veränderungen der Netzwerktopologie zum Vorschein kommen. Gleichfalls ist anzunehmen, dass sich das DSR- und das TLS-Protokoll gegenseitig negativ bedingen, und zwar aufgrund der ersten Vermutung.

Eine zuverlässige Transportschicht für das TLS-Protokoll einzuführen (wie es auch der TLS-Standard verlangt), ist meines Erachtens ein vielversprechender Ansatz. Denkbar wäre auch der Einsatz einer weiterentwickelten Form von TLS, das sogenannte Datagram Transport Layer Security Protocol.[22] Dieses kann im Gegensatz zu TLS auch über unzuverlässige Transportprotokolle wie bspw. UDP übertragen werden.

Weitere Anknüpfungspunkte bestehen zum Beispiel in der Verbesserung des Routing-Protokolls und dem Ausbau der ersten Phase des Sicherheitsprotokolls (802.1X).

Hinsichtlich der Eingangs formulierte Problemstellung, welche Konzepte im Bezug auf die Sicherheitsarchitektur und -Protokolle von WMNs geeignet sind, um als notwendige Grundlage für weitere Netzwerkdienste zu dienen, so wäre für die weiterführende Forschung zudem die Frage interessant, inwiefern eine selbstorganisierte Auswahl des vertrauenswürdigen Knotens, also des SA-Servers, sinnvoll und möglich ist.

[22] Erste Vorarbeiten für die Integration in OpenSSL finden sich bei Modadugu und Rescorla (2004).

Literatur

Akyildiz, Ian F.; Xudong Wang und Weilin Wang (2005): Wireless mesh networks: a survey. *Comput. Netw.*, 47(4):445–487.

Anderson, Ross (2001): *Security Engineering*. Wiley & Sons.

Apostolopoulos, George; Vinod Peris und Debanjan Saha (1999): Transport Layer Security: How much does it really cost? In: *Proceedings of The IEEE INFOCOM*. S. 717–725.

Dierks, Tim und Christopher Allen (1999): *The TLS protocol. Version 1.0*. Techn. Ber.

Duckwall, Alva Skip (2011): A Bridge Too Far: Defeating Wired 8021x with a Transparent Bridge Using Linux. Vortrag auf der DefCon 19.

Eckert, Claudia (2008): *IT-Sicherheit. Konzepte, Verfahren, Protokolle*. Oldenburg Verlag München, 5. Aufl.

Fall, Kevin und Kannan Varadhan (2008): *The ns Manual. The VINT Project*. UC Berkeley, LBL, USC/ISI, and Xerox PARC.

Freifunk (2013): Was ist Freifunk? Internet: http://start.freifunk.net/, Stand: 28.1.2013.

Gennaro, Rosario; Shai Halevi; Hugo Krawczyk und Tal Rabin (2008): Threshold RSA for dynamic and ad-hoc groups. In: *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*. Springer-Verlag, Berlin, Heidelberg, EUROCRYPT'08, S. 88–107.

Goldsmith, Andreas (2005): *Wireless Communication*. Cambridge University Press.

IEEE (2004a): 802.11i, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 6: Medium Access Control (MAC) Security Enhancements.

IEEE (2004b): IEEE Standard 802.1X, Port-based Network Access Control.

Johnson, D.; Y. Hu und D. Maltz (2007): The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728.

Kohler, Eddie; Robert Morris; Benjie Chen; John Jannotti und M. Frans Kaashoek (Aug. 2000): The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297.

Martignon, Fabio; Stefano Paris und Antonio Capone (2009): Design and implementation of MobiSEC: A complete security architecture for wireless mesh networks. *Comput. Netw.*, 53(12):2192–2207.

Martignon, Fabio; Stefano Paris und Antonio Capone (2010): DSA-Mesh: a Distributed Security Architecture for Wireless Mesh Networks. Wiley InterScience.

Modadugu, Nagendra und Eric Rescorla (2004): The Design and Implementation of Datagram TLS.

Newsome, James; Elaine Shi; Dawn Song und Adrian Perrig (2004): The sybil attack in sensor networks: analysis & defenses. In: *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, New York, NY, USA, IPSN '04, S. 259–268.

Redwan, Hassen und Ki-Hyung Kim (2008): Survey of Security Requirements, Attacks and Network Integration in Wireless Mesh Networks. In: *Proceedings of the 2008 Japan-China Joint Workshop on Frontier of Computer Science and Technology*. IEEE Computer Society, Washington, DC, USA, FCST '08, S. 3–9.

Salem, Naouel Ben und Jean-Pierre Hubaux (2006): Securing Wireless Mash Networks.

Zhou, Lidong und Z. J. Haas (1999): Securing ad hoc networks. *Netwrk. Mag. of Global Internetwkg.*, 13(6):24–30.

Zubow, Anatolij; Robert Sombrutzki und Markus Scheidgen (2012): A Low-cost MIMO Mesh Testbed based on 802.11n. *IEEE Wireless Communications and Networking Conference, France.*