Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust

Marc Ströbel
Betreuerin: Corinna Schmitt
Seminar Sensorknoten: Betrieb, Netze und Anwendungen SS2010
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitektur
Fakultät für Informatik, Technische Universität München
Email: stroebem@in.tum.de

KURZFASSUNG

In vielen Anwendungsgebieten von Sensornetzwerken ist die Authentizität, Integrität und Vertraulichkeit erfasster Daten eine wichtige Voraussetzung für einen erfolgreichen Einsatz. Um diesen Anspruch zu gewährleisten, muss in erster Linie die drahtlose Kommunikation innerhalb eines Sensornetzwerkes verschlüsselt werden. Da die Rechenleistung und Speicherkapazität einzelner Knoten in Sensornetzwerken jedoch stark begrenzt ist, kann dies im Allgemeinen nicht mit konventionell benutzten Verfahren zum Aufbau sicherer Verbindungen, wie etwa Diffie-Hellman, realisiert werden.
Das Paper *Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust* stellt ein günstiges Verfahren zum Schlüsselaustausch zwischen Sensorknoten vor, welches nur auf einer geringen Menge von vorweg ausgetauschten Schlüsseln basiert. Zudem begnügt es sich mit dem Gebrauch symmetrischer Kryptographie und benötigt keine vertrauenswürdige Basisstation.

Schlüsselworte

Lightweight Key Management, Wireless Sensor Networks, Initial Trust

1. EINLEITUNG

Ursprünglich entwickelt in der Rüstungsindustrie für den militärischen Kontext, haben Sensornetzwerke heute bereits Einzug in viele Anwendungen in unserem alltäglichen Leben gewonnen. Im Vergleich zu traditionell eingesetzten Sensoren, besitzen Sensoren in drahtlosen Sensornetzwerken aufgrund ihrer Autonomie nur sehr begrenzte Ressourcen. Ihre unabhängige Energiequelle und drahtlose Datenübertragung ermöglicht jedoch den ausdrücklich gewünschten Einsatz in schwer erreichbaren Zielgebieten, oder in mobilen Szenarien [1].
Dass Sicherheitsaspekte in Sensornetzwerken, auch außerhalb des militärischen Bereichs, von kritischer Bedeutung sind, zeigt ein Einsatzszenario von Sensornetzwerken als tragbare Technologie im Gesundheitswesen: Ein Patient wird mit mehreren tragbaren Geräten ausgestattet, welche bestimmte Vitalparameter überwachen und deren Werte per Funk in das Netzwerk des Krankenhauses übermitteln. Diese Anwendung von Sensornetzwerken ermöglichen einem Patienten mehr Freiräume während seiner Genesung, in manchen Fällen sogar einen kürzeren Klinikaufenthalt mit einer anschließenden elektronischen Überwachung zu Hause [9]. Offensichtlich kann eine ungesicherte Kommunikation in dieser Anwendung nicht nur zu einer empfindlichen Verletzung der Privatsphäre des Patienten führen, sondern im schlimmsten Fall auch zu einer falschen Behandlung, aufgrund kompromittierter Datensätze.
Um eine sichere Kommunikation zwischen den Sensorknoten zu gewährleisten, und somit Authentizität, Integrität und Vertraulichkeit erfasster Daten sicherzustellen, wird ein *effizienter* Schlüsselaustauschmechanismus benötigt. *Effizienz* ist erforderlich, da Sensorknoten im Allgemeinen nur sehr limitierte Rechenleistung besitzen und daher aufwändige Verfahren, wie Public-Key-Verschlüsselung oder das Diffie-Hellman Schlüsselaustauschprotokoll, zu teuer sind [4]. Ein möglicher Lösungsansatz umgeht diese Problematik durch den Einsatz vertrauenswürdiger Basisstationen. Eine Basisstation ist ein zentraler Knoten im Sensornetzwerk, welcher sowohl über unterbrechungsfreie Energie, als auch leistungsstarke Hardware verfügt. Neben der Speicherung und Weiterleitung eingehender Daten der Sensorknoten, wird eine Basisstation oftmals auch als verauswürdiger Mittler eingesetzt, der Sensorknoten im Netzwerk gegenseitig authentifiziert und paarweise Schlüssel bereitstellt. So können sich Sensoren gegenüber der Basisstation durch ein langfristiges Geheimnis authentifizieren, welche mit ihrer Rechenleistung, starke Verfahren zur Schlüsselerzeugung einsetzten kann.
Dutertre et al. identifizieren folgende Schwächen in diesen Verfahren:

- Zusätzlicher administrativer Aufwand: Noch vor dem Aufstellen des Sensornetzwerkes im entsprechenden Zielgebiet, muss ein Austausch eines langfristigen, individuellen Geheimnisses zwischen Basisstation und jedem Knoten des Netzwerkes erfolgen.
- Die Eigenschaft der Basisstation als Single-Point-of-Failure: Die Basisstation bietet ein ideales Ziel für einen Denial-of-Service Angriff und im Falle einer Kompromittierung verliert jeglicher Schlüssel im Sensornetzwerk seine Gültigkeit.

Darüber hinaus existieren Netztopologien für Sensornetzwerke, welche bewusst auf eine Basisstation verzichten, und somit auf ein Schlüsselaustauschverfahren für limitierte Rechenleistung angewiesen sind.
In [7] entwerfen Dutertre et al. ein Schlüsselverwaltungsverfahren mit minimalen administrativen Aufwand, welches basierend auf einen minimalen Anteil an bereits vorweg ausgetauschten Schlüsseln, sichere Verbindungen für Untermen-

gen eines Sensornetzwerkes aufbauen kann – ohne aufwändige Verschlüsselungsverfahren oder vertrauenswürdigen Basisstationen.

Im Kapitel 2 werden zunächst einige Zeichen und Abkürzungen als Notation eingeführt. Anschließend behandelt Kapitel 3 den Ablauf und die Funktionsweise Dutertre et al.'s Schlüsselverwaltungsverfahrens. Kapitel 4 beinhaltet eine objektive Bewertung der Sicherheit und Ausfallsicherheit des Verfahrens, und zieht einen Vergleich mit ausgewählten Protokollen zur Schlüsselverwaltung. Nach einer kurzen Vorstellung der Implementation des Lightweight Key Management für das Betriebssystem TinyOS in Kapitel 5, stellt Kapitel 6 eine Zusammenfassung der Arbeit dar.

2. NOTATION

Um das Schlüsselaustauschverfahren von Dutertre et al. [7] im Folgenden genauer beschreiben zu können, werden zunächst einige Abkürzungen für einzelne Schlüssel und kryptographische Funktionen einführt:

A, B, C, ...	einzelne Sensorknoten
N_a	eine Zufallszahl, generiert vom Sensorknoten A
R_a	eine Zufallszahl, vor dem Aufstellen abgelegt im Sensorknoten A
$G_k(m)$	Hashfunktion mit Schlüssel k über Zeichenkette m
$MAC_k(m)$	Message-Authentication-Code für Nachricht m erzeugt mit Schlüssel k
bk_1	Authentifikationschlüssel für das Bootstrapping
bk_2	Schlüssel zur Schlüsselgenerierung für das Bootstrapping
gk_i	Authentifikations Schlüssel für Generation i
K_{ab}	Gemeinsamer Schlüssel zwischen den Sensorknoten A und B
$ENC_{K_{ab}}(m)$	Verschlüsselung der Nachricht m mit dem Schlüssel K_{ab}

Die Hashfunktion G mit Schlüssel k hat die Eigenschaft, dass der mit ihr berechnete Schlüssel r unter der Eingabe von Schlüssel k und der Zeichenkette m ($G_k(m) = r$) effizient berechenbar ist, jedoch ohne die Kenntnis von k keine Aussage über r getroffen werden kann.
Der Message-Authentication-Code $MAC_k(m)$ ist ebenfalls eine Hashfunktion mit Schlüssel k, welche zur Authentifizierung von Nachrichten verwendet wird: Wenn ein Knoten A einen gemeinsamen Schlüssel K_{ab} mit Knoten B teilt, kann A den Hashwert $MAC_{K_{ab}}(m)$ seiner Nachricht m zu B mitschicken. B kann nun überprüfen, ob die Nachricht m wirklich von A versendet wurde, indem es den Hashwert selbst berechnet und mit dem angehängten Hashwert von A vergleicht. Dieses Verfahren sichert sowohl die Authentizität als auch die Integrität einer Nachricht [8, Kap. 7].
Die Abkürzung $ENC_{K_{ab}}(m)$ steht für die Verschlüsselung der Nachricht m mit dem Schlüssel K_{ab} und ermöglicht den Knoten A und B, wenn sie im Besitz dieses Schlüssels sind, vertrauliche Kommunikation.
Des Weiteren betrachten wir im folgenden Kapitel das Sensornetzwerk S mit den Knoten $\{A, B, C, D\}$.

3. DAS PROTOKOLL

Das Schlüsselaustauschverfahren erfolgt in zwei Phasen: Zunächst baut jeder Knoten sichere Kanäle (Secure Local Links) zu allen Nachbarknoten in seiner Reichweite auf. Im zweiten Schritt können mit Hilfe dieser sicheren Kanäle, welche das gesamte Sensornetzwerk verbinden, gemeinsame Schlüssel für Gruppen von Knoten vereinbart werden.

3.1 Secure Local Link

Das Initialisieren des sicheren Kanals – von Dutertre et al. auch als Bootstrapping ihres Protokolls bezeichnet – beruht im Wesentlichen auf den gemeinsamen Schlüsseln bk_1 und bk_2, welche vor dem Aufstellen auf allen Knoten von S abgelegt werden.
Betrachten wir die Initialisierung eines Kanals zwischen den Knoten A und B (siehe Abbildung 1): Nach der Aktivierung des Sensornetzwerkes, beginnt Knoten A mit dem Broadcast der Nachricht $< Hello, A, N_a, MAC_{bk1}(Hello, A, N_a) >$, welche neben seiner Identität A, eine Nonce N_a sowie den Message-Authentication-Code $MAC_{bk1}(Hello, A, N_a)$ über der gesamten Nachricht enthält. Der Message-Authentication-Code erzeugt mit dem gemeinsamen Schlüssel bk_1 authentifiziert A bei dem jeweiligen Empfänger als Mitglied des Sensornetzwerkes S. Wird die Nachricht von B oder einem anderen Mitglied von S empfangen, sendet dieses in einer Empfangsbestätigung die eigene Identität B, eine Nonce N_b und einen Message-Authentication-Code zurück. Da auch B seiner Nachricht einen MAC über m und der Nonce N_a, erzeugt mit bk_1 anhängt, haben sich A und B gegenseitig authentifiziert.
Im nächsten Schritt berechnen beide Knoten den gemeinsamen Schlüssel K_{ab} aus der Funktion $G_k(m)$, den beiden Nonces N_a, N_b und dem gemeinsamen Schlüssel bk_2: $K_{ab} = G_{bk_2}(N_a N_b)$. K_{ab} ermöglicht den Knoten A und B zukünftig, sicher und effizient durch symmetrische Verschlüsselung zu kommunizieren.

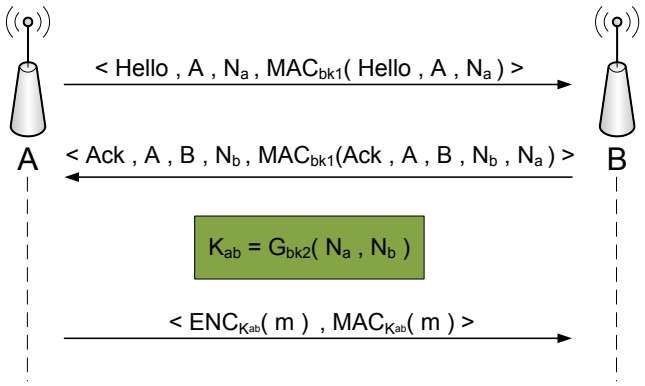

Abbildung 1: Aufbau eines Secure Local Links

Hat ein Knoten sichere Kanäle zu den in seiner Reichweite liegenden Nachbarknoten aufgebaut, werden die gemeinsamen Schlüssel bk_1 und bk_2 gelöscht. Dieses Verhalten stellt sicher, dass eine spätere Kompromittierung des Knotens, nicht jeglichen ausgehandelten Schlüssel gefährdet (siehe auch Kapitel 4.1).

3.2 Group-Key Distribution

Die Secure Local Links können mit sogenanntem *chaining* – dem Weiterleiten einer Nachricht von Knoten zu Knoten, auch zum Kommunizieren zwischen zwei entfernten Knoten verwendet werden:

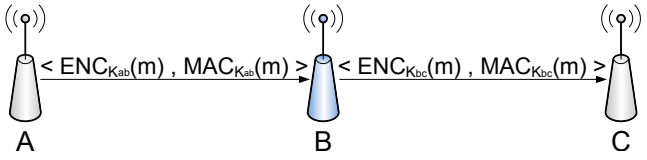

Abbildung 2: Verkettung einer Nachricht vom Sender A zum Empfänger C

Um eine teurere Neuverschlüsselung an jedem Knoten für Multicast- oder Broadcastnachrichten zu verhindern, kann über die aufgebauten lokalen Kanäle ein Gruppenschlüssel ausgehandelt werden, welcher Message Chaining für Multicast durch eine einmalige Verschlüsselung ersetzt. Dazu besitzt Dutertre et al.'s Protokoll den Mechanismus *KeyRefresh*, welcher einen Gruppenschlüssel K_g an alle gewünschten Knoten im Sensornetzwerk versendet. Ein KeyRefresh von A, gesendet an B, hat die Form:

$$< KeyRefresh, B, A, O, N, ENC_{K_{ab}}(K_g), L, \\ MAC_{K_{ab}}(KeyRefresh \dots) >$$

Während der Wert O für den Herkunftsknoten des neuen Schlüssels steht, bezeichnet L eine Liste auszuschließender Knoten. L kann dual genutzt werden, einerseits um einen gemeinsamen Schlüssel nur zwischen einer Untermenge des Sensornetzwerkes zu instantiieren, andererseits um mögliche kompromittierte Knoten aus der Gruppe auszuschließen (siehe auch Kapitel 4.1). Für letzteren Fall existiert auch das Feld N, welches die Sequenznummer des Gruppenschlüssels angibt: mit ihm wird sichergestellt, dass ein bereits ausgeschlossener Knoten keine neuen Schlüssel über eine KeyRefresh-Nachricht verbreitet. Der zu verteilende neue Gruppenschlüssel K_g wird für Dritte unlesbar ($ENC_{K_{ab}}(K_g)$), verschlüsselt mit dem paarweisen Schlüssel von A und B, in die Nachricht eingefügt. Der MAC wird analog zu den anderen Protokollnachrichten wieder über den Inhalt der gesamten Nachricht berechnet.
Wenn Knoten A also einen Schlüssel K_{g1} zwischen ihm und den Knoten B und C instantiieren möchte, versendet er die Nachricht $< KeyRefresh, B, A, A, 0, ENC_{K_{ab}}(K_{g1}), \{D\}, MAC_{K_{ab}}(KeyRef \dots) >$ an seinen einzigen Nachbarn B, welcher die Nachricht an C weiterleitet: $< KeyRefresh, C, B, A, 0, E_{K_{bc}}(K_{g1}), \{D\}, MAC_{K_{bc}}(KeyRef \dots) >$.
Multicast-Nachrichten von A können nun per Broadcast im Sensornetzwerk verteilt werden (siehe Abbildung 3), wobei der Schlüssel K_{g1} sicherstellt, dass nur B und C die Nachrichten lesen können.

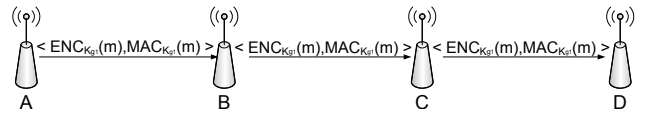

Abbildung 3: Senden einer Multicast-Nachricht von A nach B und C

3.3 Multiphase Deployment

Oftmals werden in Sensornetzwerken, zu einem späteren Zeitpunkt, neue Sensoren hinzugefügt oder defekte Sensoren ausgetauscht. Diese benötigte Flexibilität muss auch vom Schlüsselverwaltungsprotokoll unterstützt werden. Dutertre et al. haben daher das *Multiphase Deployment* in ihr Protokoll implementiert. Das Multiphase Deployment sieht vor, dass verschiedene Generationen von Sensorknoten, zum Beispiel die Generationen j bis n, nacheinander eingesetzt werden können. Untereinander etablieren die Knoten einer Generation paarweise Schlüssel mit den gemeinsamen Geheimnissen bk_1 und bk_2 ihrer Generation (siehe Kapitel 3.1). Weil generationsübergreifend bk_1 und bk_2 verschieden sind, wird jeder Knoten X einer Generation i mit einer Zufallszahl R_x und einer Menge an Schlüsseln $S_{x,(i+1)}, .., S_{x,n}$ initialisiert. $S_{x,i}$ ist zwar mit der Hashfunktion G berechenbar, es gilt jedoch, dass $G_{gk_i}(R_x) = S_{x,i}$ nur mit Hilfe des Generationenschlüssels gk_i effizient berechenbar ist. Der Generationenschlüssel gk_i der Generation i steht jedoch nur Sensoren der Generation i zur Verfügung.
Zurück zum Beispiel des Sensornetzwerkes S: Sei S für den Einsatz von zwei Generationen angelegt, so besitzt jeder der Knoten $\{A, B, C, D\}$ aus der ersten Generation eine Zufallszahl $R_a, .., R_d$, sowie ihren zugehörigen Schlüssel für die zweite Generation $S_{a,2}, .., S_{d,2}$. Besteht nun die zweite Generation aus dem Sensorknoten E, welcher in unmittelbarer Nähe von A platziert wird, dann hat der Schlüsselaustausch zwischen E und A folgende Form (siehe Abbildung 4): E sendet die Nachricht $< Hello, E, 2, N_E >$, welche neben dem Absender, die Kennzahl 2 für E's Generation und eine Zufallszahl enthält. Diese Nachricht besitzt keinen Message-Authentication-Code, da A und E bisher keinen gemeinsamen Schlüssel besitzen. Erhält A die Nachricht und befindet sich, wie in unserem Falle, im Besitz eines Schlüssels für die zweite Generation, schickt A eine Empfangsbestätigung $< Ack, A, E, R_a, MAC_{S_{a,2}}(Ack, A, E, R_a, N_E) >$. Diese Empfangsbestätigung enthält die Zufallszahl R_a und einen MAC erzeugt mit $S_{a,2}$. A authentifiziert sich somit als Teil einer vorausgegangenen Generation im Sensornetzwerk S. E kann den Message-Authentication-Code überprüfen, indem es $S_{a,2}$ aus G, gk_2 und R_a ($G_{gk_2}(R_a) = S_{a,2}$) berechnet. Anschließend authentifiziert E sich gegenüber A in einer weiteren Empfangsbestätigung und schließt somit den Schlüsselaustausch zwischen ihnen ab. Im weiteren Verlauf können die beiden Knoten über den Schlüssel $S_{a,2}$ sicher kommunizieren.

4. DISKUSSION

Grundsätzlich unterliegt ein Schlüsselverwaltungsverfahren gewissen Annahmen und Einschränkungen, welche Sicherheit und Ausfallsicherheit des Verfahrens betreffen, im Besonderen im Verfahren von Dutertre et al. aufgrund der Absicht, ein leichtgewichtiges Protokoll zu entwerfen. Das folgende Kapitel wirft eine Blick auf diese Eigenschaften und vergleicht das Protokoll mit ähnlichen Arbeiten.

4.1 Sicherheit und Ausfallsicherheit

4.1.1 Bootstrapping

Die paarweise Authentifizierung und Erzeugung eines gemeinsamen Schlüssels beruht auf einem Verfahren, das bereits 1993 von Bellare und Rogaway vorgestellt wurde [2]. Bellare und Rogaway haben bewiesen, dass ihr Verfahren

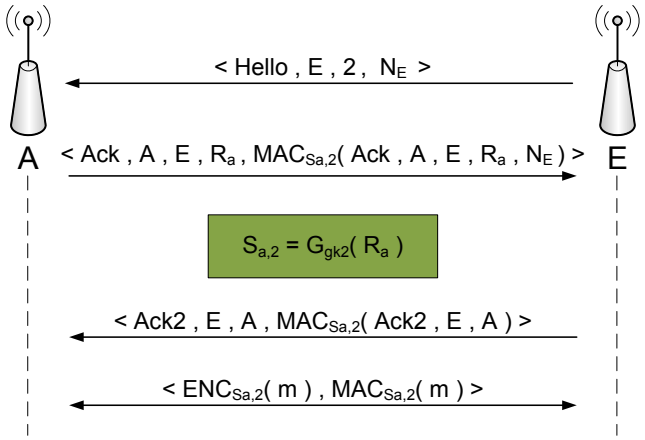

Abbildung 4: Schlüsselaustausch zwischen verschiedenen Generationen von Sensorknoten

ausreichende Sicherheit gegen einen Angreifer, der sowohl den Netzverkehr abhören, als auch Nachrichten in das Netz einspeisen kann, bietet. Die Voraussetzung für diese Sicherheit ist jedoch, dass der Angreifer weder Kenntnis des Schlüssel bk_1 noch des Schlüssels bk_2 besitzt. Eine größere Gefahr geht im Szenario der Sensornetzwerke von der Kompromittierung einzelner Knoten aus: Sollte es einem Angreifer gelingen, einen Knoten zu übernehmen und die Bootstrapping-Schlüssel bk_1 und bk_2 abzurufen, kann er an jegliche Schlüssel durch Abhören der Nonces gelangen. Darüber hinaus kann er sich selbst als Teil des Sensornetzwerkes gegenüber den anderen Knoten authentifizieren. Aus diesem Grund sollte das Zeitfenster des Bootstrappings so kurz wie möglich gehalten werden, um die Schlüssel bk_1 und bk_2 anschließend zu löschen.
Im Bezug auf die Ausfallsicherheit besitzt das Bootstrapping mehrere negative Bedingungen:

- Das Bootstrapping soll auf einen möglichst kurzen Zeitraum beschränkt sein, um die Sicherheit der Schlüssel bk_1 und bk_2 zu gewährleisten.
- Die Funkverbindung der Sensorknoten ist unzuverlässig.
- Das Erreichen von Nachbarknoten basiert auf Broadcastnachrichten, was dazu führen kann, dass Kollisionen mehrerer Nachrichten auftreten.

Um trotzdem eine robuste Ausfallsicherheit zu gewährleisten, initiieren die Sensoren ihre Nachrichten zu zufällig gewählten Zeitpunkten, um die Wahrscheinlichkeit von Kollisionen gering zu halten. Zusätzlich ist es möglich, die Sensoren auf das Versenden mehrerer *Hello*-Nachrichten zu konfigurieren, so dass im Falle von Paket-Kollisionen eine zweite Chance zur Erzeugung eines gemeinsamen Schlüssels besteht.

4.1.2 Key-Refresh

Solange der Initiator und alle Empfänger eines Key-Refresh vertrauenswürdig sind, besteht sicherlich keine Gefahr für eine Kompromittierung des gemeinsamen Schlüssels. Unmittelbar problematisch wird es, wenn es einem Angreifer gelingt, einen Knoten der Gruppe zu übernehmen und eventuell sogar durch das Initiieren einer eigenen Key-Refresh-Nachricht andere Knoten aus der Gruppe auszuschließen. Letztlich stellt sich die Frage wie ein Knoten die Übernahme eines anderen Gruppenknotens entdecken kann, um folglich einen neuen Schlüsselaustausch unter Ausschluss dieses Knotens durchzuführen. Diese Problematik führt in den Bereich der *Intrusion Detection* Systeme für Sensornetzwerke. Als weiterführende Literatur kann zum Beispiel die Arbeit [5] betrachtet werden.

4.1.3 Multiphase Deployment

Analog zum Bootstrapping muss auch im Multiphase Deployment darauf geachtet werden, den Schlüsselaustausch auf einen möglichst kurzen Zeitraum zu beschränken, da der Verlust des Schlüssels gk_i der Generation i, alle paarweisen Schlüssel zwischen Generation i und Knoten aus vorausgegangenen Generationen kompromittiert.
Darüber hinaus muss beachtet werden, dass ein Schlüssel $S_{a,i}$ eines Knotens A zu einem Sensor der Generation i identisch zu weiteren Schlüsseln zwischen A und anderen Sensoren der Generation i sind, da alle auf dem Geheimnis der Nonce R_a beruhen. So ist es einem Angreifer möglich, durch die Übernahme eines Knotens, eine Vielzahl von paarweisen – sicher geglaubten – Verbindungen zu kompromittieren.

4.2 Das Protokoll im Vergleich

Die große Schwachstelle von Dutertre et al.'s Protokoll ist die Tatsache, dass ein Angreifer im Besitz der Schlüssel bk_1, bk_2 und der Generationenschlüssel $gk_j, .., gk_n$ alle Schlüssel in einem Sensornetzwerk kompromittieren könnte.
Andere Verfahren wie zum Beispiel das *q-Composite Random Key Predistribution Scheme*, vorgestellt in [6], oder das *Location-based pairwise Key Establishment* [10], bieten eine höhere Sicherheit, bringen jedoch den, von Dutertre et al. missbilligten, großen administrativen Aufwand mit sich:
Das *q-Composite Random Key Predistribution Scheme* verteilt auf jedem Knoten des Sensornetzwerkes eine zufällige Teilmenge, mit der Kardinalität n, aus einer großen Menge B von Schlüsseln. Haben nun zwei Nachbarknoten mehr als q ($q > 1$) gemeinsame Schlüssel, benutzen sie einen Hash all ihrer gemeinsamen Schlüssel als Schlüssel für ihren sicheren Kanal. Die n Schlüssel sind mit einer bestimmten Wahrscheinlichkeit p aus B gewählt, um eine Mindest-Konnektivität im Sensornetzwerk zu garantieren. Dieses komplexe Verfahren verhindert dass ein Angreifer mit der Übernahme eines Knotens mehr als die paarweisen Verbindungen dieses Knotens kompromittieren kann.
Das *Location-based pairwise Key Establishment* verteilt ähnlich, wie das *q-Composite Random Key Predistribution Scheme*, vorweg Schlüssel auf die einzelnen Knoten. Da es zusätzlich die Annahme trifft, dass die Position der einzelnen Knoten vorweg bekannt ist, ermöglicht es eine effizientere Verteilung der Schlüssel. Diese Effizient führt zu einer bedeutenden Verringerung des Speicherbedarfs.
Andere Schlüsselverwaltungsprotokolle basieren auf dem Konzept von Blundo et al., vorgestellt in [3], welches symmetrische Polynome ($P(x, y) = P(y, x)$) vom Grad λ einsetzt: Jeder Knoten i speichert ein Polynom mit $\lambda + 1$ Koeffizienten, welches einem symmetrischen Polynom $P(x, y)$ an der Stelle $P(i, y)$ entspricht. Ohne zu tief in die Zahlentheorie

abzusteigen (für die genaue Funktionsweise siehe [3]) sei gesagt, dass jeder Knoten i nun mit seinem Polynom $f_i(y)$, welches dem Polynom $P(x, y)$ an der Stelle (i, y) entspricht $(f_i(y) = P(i, y))$, einen gemeinsamen Schlüssel mit einem beliebigen Knoten j berechnen kann. Zur Berechnung des gemeinsamen Schlüssels wertet i sein Polynom $f_i(y)$ an der Stelle j aus. Dementsprechend berechnet j den gemeinsamen Schlüssel aus $f_j(i)$. Somit erzeugen beide Knoten den gemeinsamen Schlüssel $K_{ij} = f_i(j) = P(i, j) = f_j(i)$ [4]. Solange ein Angreifer nicht mehr als λ Knoten kompromittiert, und somit nicht mehr als λ Polynome verschiedener Knoten erlangt hat, sind alle erzeugten Schlüssel sicher. Hat ein Angreifer jedoch $\lambda + 1$ Knoten kompromittiert, ist er in der Lage aus diesen, alle Koeffizienten des ursprünglichen Polynoms $P(x, y)$ zu berechnen und somit jeden beliebigen Schlüssel im Sensornetzwerk zu bestimmen.
Auch diese Verfahren ziehen mehr Verwaltungsaufwand als Dutertre et al.'s Schema auf sich, jedoch können sie als wesentlich sicherer eingeschätzt werden, da ein Angreifer im Zweifelsfall *sehr viele* Sensorknoten in seine Gewalt bringen muss.

5. IMPLEMENTIERUNG

Dutertre et al. stellen in ihrem Paper eine Implementierung ihres Protokolls für das TinyOS Operating System [12] vor, welche neben der Protokolllogik hauptsächlich einen zusätzlichen Netzwerkstack in das TinyOS integriert. Ein neuer Netzwerkstack ist notwendig, da weder der Standard-Stack noch die sicherheitsspezifische Erweiterung des alten Stacks, TinySec [11], die nötige Funktionalität für das *Bootstrapping* und *Key-Refresh* bietet: Der ursprüngliche Stack lässt jegliche Sicherheitsfeatures vermissen, aber auch TinySec bietet nur Verschlüsselung und Signierung auf Basis *eines* Schlüssels an, nicht jedoch die Möglichkeit, verschiedene Schlüssel für verschiedene Empfänger zu benutzen. Des Weiteren wird das Signieren von unverschlüsselten Nachrichten, wie beim Bootstrapping angewandt, nicht von TinySec unterstützt.
Dutertre et al.'s Netzwerkstack verwendet viele Komponenten beider Stacks, erweitert diese aber auch durch ein *raw Messages*-Format, in welchem der Stack die Nachrichten ausschließlich versendet, Verschlüsselung, Signierung und jegliche Überprüfungen aber einem Programm überlässt. Auch beim Empfang werden *raw Messages* ohne Verarbeitung direkt wieder an die zugehörige Applikation hochgegeben. Eine andere Erweiterung ist die flexible Schlüsselwahl für das Ver- und Entschlüsseln ausgehender bzw. eingehender Nachrichten.
Das Bootstrapping implementieren Sie durch einen *SecureLinkManager*, der mit raw Messages das Bootstrappingprotokoll realisiert und eine Tabelle von authentifizierten Nachbarknoten, mit paarweisen Schlüsseln, aufbaut. Für die Verschlüsselung wird der Advanced Encryption Standard verwendet (für detaillierte Information siehe [8, Kap. 6]).
Auch der Mechanismus Key-Refresh wurde bereits als Prototyp realisiert. Zur Verbreitung von Key-Refresh-Nachrichten greift der Prototyp auf die Tabelle des SecureLinkManagers zu und versendet die Nachrichten im *raw Message* Format.

6. ZUSAMMENFASSUNG

Das Verfahren des *Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust* bietet für den Fall, dass weder die Bootstrapping Schlüssel bk_1 und bk_2, noch die Generationenschlüssel gk_i in die Hand eines Angreifers gelangen, gute Sicherheit, bei minimalem Verwaltungsaufwand. Auch die Implementierung für das TinyOS zeigt, dass die Autoren ihre Ziele erreicht haben, und ihre Schlüsselverwaltung mit limitierten Ressourcen einsatzfähig ist. Doch es sollte darauf hingewiesen werden, dass Protokolle, wie das *Random pairwise Key Scheme* deutlich stärkere Sicherheit mit höherem, aber automatisierbarem Verwaltungsaufwand besitzen und daher Dutertre et al.'s Arbeit vorgezogen werden sollten – sofern diese mit ihrem deutlich höheren Speicheraufwand auf den zugrunde liegenden Sensoren realisierbar sind.

7. LITERATUR

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[2] M. Bellare and P. Rogaway. Springer-verlag. this is the full version. entity authentication and key distribution, 1993.

[3] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 471–486, London, UK, 1993. Springer-Verlag.

[4] S. A. Camtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical report, 2005.

[5] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong. Decentralized intrusion detection in wireless sensor networks. In *Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pages 16–23, New York, NY, USA, 2005. ACM.

[6] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[7] B. Dutertre, S. Cheung, and J. Levy. Lightweight key management in wireless sensor networks by leveraging initial trust, sdl. Technical report.

[8] C. Eckert. *IT-Sicherheit : Konzepte - Verfahren - Protokolle*. Oldenbourg, 2001.

[9] J. M. Ferro, L. M. Borges, O. J. Velez, and A. S. Lebres. Applications of wireless sensor networks.

[10] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 29–42, New York, NY, USA, 2004. ACM.

[11] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In *in Ambient Intelligence*. Springer Verlag, 2004.