„Random Key Distribution“ Verfahren in WSNs – Vergleich verschiedener probabilistischer Ansätze

Sebastian Bendeich
Betreuerin: Corinna Schmitt
Seminar Innovative Sensorknoten: Betrieb, Netze und Anwendungen SS2011
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitektur
Fakultät für Informatik, Technische Universität München
Email: sebastian.bendeich@mytum.de

KURZFASSUNG

Verschlüsselte paarweise Kommunikation und Authentifizierungsmöglichkeiten spielen eine wichtige Rolle in drahtlosen Sensornetzwerken (WSN), da die Sensoren meist ungeschützt im Einsatzgebiet verteilt sind und ggf. sensible Daten transportieren. Aufgrund von Einschränkungen durch zur Verfügung stehende Ressourcen wie Rechenleistung, Speichergröße und Energie, sowie der Resistenzanforderungen gegenüber Angriffen und Attacken, welche auf WSNs einwirken können, spielen die verwendeten Schlüsselverteilungsstrategien eine wichtige Rolle in der Umsetzung und Ermöglichung von Abhörsicherheit und Authentifizierung. Die Authentifizierung wird insbesondere benötigt, um Manipulationsversuche erkennen und Gegenmaßnahmen, wie Ausschluss des entsprechenden Sensors aus dem Netzwerk, ergreifen zu können. Chan et al [2] und Di Pietro et al [3] haben sich mit probabilistischen Verfahren beschäftigt, die hier erläutert und verglichen werden, sowie auf Ihre Resistenz bzw. Robustheit gegenüber Angriffen beleuchtet werden.

Schlüsselworte

WSN, Random Key Schemes, Probalisitc Key Schemes, Verschlüsselung, Verschlüsselungssicherheit, Authentifizierung, DoS-Attacken, Key Managment Protokolle

1. EINLEITUNG

Der Einsatz von drahtlosen Sensornetzwerken (WSN) wird immer häufiger und umfasst immer mehr Einsatzgebiete. Da es je nach Einsatzgebiet, zum Beispiel bei der Gebäude- oder Geländeüberwachung, aber nicht zuletzt auch im militärischen Bereich, bedeutsam ist, ob die übertragenen Daten abhörsicher und verlässlich (z.B. gegenüber Manipulation) sind, bekommt auch der Aspekt der Verschlüsselung und Authentifizierung in WSNs immer größere Bedeutung. Jedoch lassen sich nicht alle, der in den üblichen EDV-Einsatzgebieten bereits verbreiteten Verfahren, wie „Public Key“-Verschlüsselung, in WSNs einsetzen [1]. Dies ist der Tatsache geschuldet, dass die einzelnen Sensoren nur begrenzte Energie, Rechenleistung und insbesondere begrenzten Speicherplatz zur Verfügung haben, die nicht zuletzt das Speichern der teils langen Schlüssel für „Public Key“-Verfahren einschränken bis ganz unmöglich machen. Denn selbst bei der Verschlüsselung von Webseiten (SSL) wird der „Public Key“ nur zum Finden und Austauschen eines gemeinsamen, symmetrischen Schlüssel genutzt und dies, obwohl die Rechenleistung von Computern und die Bandbreite der Internetanbindungen im Vergleich zu denen der Sensoren in WSNs relativ hoch ist.

In diesem Artikel werden die von Chan et al im Artikel „Random Key Predistribution Schemes for Sensor Networks“ [2] beschriebenen probabilistischen Methoden zur Schlüsselverteilung und daraus folgenden Eigenschaften und Protokolle für die Kommunikation zwischen Sensoren mit denen von Di Petro et al im Artikel „Random Key-Assignment for Secure Wireless Sensor Networks“ [3] verglichen. Insbesondere wird auf die Tauglichkeit der jeweiligen Methoden zur Verschlüsselung und Authentifizierung von Sensoren innerhalb eines WSN wertgelegt und die mit den jeweiligen Verfahren bzw. Protokollen verbundenen Schwachstellen herausgearbeitet. Es ist also an der Verschlüsselung, dem Mithörer nicht die Chance zu geben, den mitgeschnitten Datenverkehr dechiffrieren zu können. Aufgrund der Zugänglichkeit zu den Sensoren, sollte man sich bewusst sein, dass Sensoren eingesammelt oder direkt vor Ort ausgelesen oder gar manipuliert werden können und somit ggf. im Speicher des Sensors liegende Schlüssel dem Angreifer bekannt werden können. Nicht zuletzt spielt auch die mögliche Absicht eines Angreifers, das Sensornetzwerk zu stören bzw. durch Erzwingen von vermehrter Funk- oder CPU-Aktivität, die Energiespeicher der Sensoren vorzeitig zum Erliegen zu bringen eine wichtige Rolle, sodass zum Einsatz kommende Verfahren versuchen sollten, den Einfluss auf die Kommunikationseigenschaften und die lokalen Ressourcen durch derartige Attacken zu minimieren. Detailliertere Beschreibungen der genannten und weiterer Angriffsmotive und –verfahren können u.a. dem Artikel „Security for wireless sensor networks“ von Avancha et al [4] und den Ausführungen in „A survey of security issues in mobile ad hoc and sensor networks“ von Djenouri et al [5] entnommen werden.

Zunächst nun ein paar einführende Definitionen, bevor mit der Vorstellung der Verfahren begonnen wird.

2. DEFINITION SYMMETRISCHE VERSCHLÜSSELUNG

Unter symmetrischer Verschlüsselung versteht man ein Kryptografieverfahren, bei dem die Nachricht mit demselben Schlüssel ver- und auch wieder entschlüsselt wird. Im Gegensatz zu asymmetrischen Verschlüsselung („Public Key“) bei der verschiedene Schlüssel zum Ver- und Entschlüsseln existieren,

doi: 10.2313/NET-2011-07-1_12

sodass der Verschlüsselungsschlüssel frei bekannt gegeben werden kann. Bei der symmetrischen Verschlüsselung ist es also von Vorteil einen Schlüssel nur für eine paarweise Kommunikation einzusetzen, falls sichergestellt werden soll, dass nur Sender und Empfänger in der Lage sind, die Nachricht entschlüsseln zu können. Dies führt auch schon zum Kernproblem von symmetrischen Verschlüsselungsverfahren: auf welche Weise sollen die Schlüssel den Sensoren zugewiesen werden. Dynamische Verfahren, die generisch einen Schlüssel generieren und diesen über einen asymmetrisch verschlüsselten Kanal zum Kommunikationspartner übertragen, entfallen, aufgrund der schon eingangs erwähnten Nichteignung von WSNs für „Public Key"-Verschlüsselungsverfahren.

3. DEFINITION PROBABILISTISCHE VERTEILUNG

Eine mögliche Alternative ist das Vorverteilen eines generierten Schlüsselpools auf die auszubringenden Sensoren, d.h. die Schlüssel werden zusammen mit der Software aufgespielt. Bei probabilistischen Verfahren, wird jedem Sensor zufällig eine Teilmenge der verfügbaren Schlüssel im Pool zugewiesen. Zwei Sensoren können nun direkt miteinander kommunizieren, falls sie mindestens einen Schlüssel gemeinsam haben. Da dies neben einigen Parametern wie Pool- und Teilmengengröße von zufälligen Ereignissen abhängt, spricht man von probabilistischen Verteilungsansätzen. Eschenauer und Gligor veröffentlichten eines der ersten probabilistischen Verfahren [6].

Generell unterscheidet man bei probabilistischen Verfahren drei Phasen (nach dem Aufteilungsschema von Di Pietro et al [3]): die erste Phase (Predeployment) bezeichnet die Zeit, in der (z.B. im Labor) der Speicher des Sensor vor Ausbringung in das eigentliche Einsatzgebiet mit Software oder anderweitigen Daten (z.B. Schlüsseln) bestückt werden kann. In der zweiten Phase (Key Discovery) erkunden die Sensoren, nachdem Sie im Einsatzgebiet ausgebracht wurden, ihre Umgebung. Dabei ermitteln sie mit welchen der Sensoren sie direkt kommunizieren können (Funkreichweite) und welche Schlüssel sie mit ihnen teilen. In der dritten Phase (Kanalaufbau) einigen sich zwei in der Phase zuvor gefundenen Sensoren auf einen gemeinsamen Schlüssel für eine sichere, paarweise Kommunikation.

4. VERFAHREN VON DI PIETRO ET AL

Di Pietro et al entwickelten zwei Protokolle für die Schlüsselverteilung [3], die aufeinander aufbauen. Das erste Verfahren „direktes Protokoll" genannt und die Erweiterung „kooperatives Protokoll", mit deren das Sicherheitslevel der Kommunikation innerhalb des WSN dynamisch verändert werden kann.

4.1 Das „direkte Protokoll"

In der Predeployment-Phase werden zunächst P zufällige Schlüssel generiert.

$$P = \{v_1, \ldots, v_P\}$$

Dabei ist jedem Schlüssel ein Index (z.B. fortlaufende Nummer) zugeordnet. Jeder Sensor a soll am Ende dieser Phase eine k Element große Teilmenge des Schlüsselpools P in sich tragen. Die Zuweisung der k Schlüssel zu einem Sensor erfolgt nun nicht vollständig zufällig, wie im Verfahren von Eschenauer und Gligor [6], sondern durch ein sog. Pseudo-zufälliges Verfahren, bei welchem ein, allen Sensoren bekannter Ausgangswert (Seed) genutzt wird, um den Zuteilungsgenerator zu initialisieren. Der Zuteilungsgenerator wählt nun anhand des Ausgangswerts und der Sensor ID k Indizes aus. Die Schlüssel mit den entsprechenden Indizes werden anschließend dem Sensor zugeteilt, dies wiederholt man mit allen n Sensoren. Die Wahl eines Pseudo-Zufallsgenerators stellt damit den einzigen wirklichen Unterschied zum von Eschenauer und Gligor [6] vorgestellten Verfahren dar. Jedoch macht sich dieser Unterschied in den Eigenschaften stark bemerkbar, wie im Folgenden ersichtlich wird.

Aufgrund dass jedem Sensor der Ausgangswert und der Zuteilungsgeneratoralgorithmus bekannt sind, kann jeder Sensor a berechnen, welche Schlüssel (bzw. vielmehr die Indizes der Schlüssel) ein Sensor mit ID b haben muss.

Der Vorteil der pseudozufälligen Zuteilung offenbart sich nun in der Key Discovery Phase, da nun zum Ermitteln der gemeinsamen Schlüssel nicht die Schlüssel selbst über einen Broadcast gesendet werden müssen (und damit Gefahr gelaufen wird, dass ein möglicher Mithörer, die Schlüssel abgreifen kann), sondern es ausreicht, die eigene Sensor ID zu senden, welche keine sicherheitskritische Information ist und somit auch unverschlüsselt übertragen werden kann. Dies wirkt sich unter anderem auch positiv auf den Overhead und damit auf den Energieverbrauch aus. Empfängt Sensor a nun den Broadcast von Sensor b, so weiß a, dass er sich in Sendereichweite von b befindet und damit, dass b höchstwahrscheinlich auch in der Sendereichweite von ihm liegt.

```
getSitzungsschlüssel(p: sensorID)
Input: ID des Kommunikationspartners (beim Empfänger
Sender-ID und vice versa)

k_a,b = 0;
partner_indizes = calculateIndizes(p);
found=false;
for each partner_index ∈ partner_indizes {
        if partner_index ∈ meine_indizes {
                k_a,b = k_a,b ⊕ meine_keys[partner_index];
                found = true;
        }
}
if( !found) {
        error();
} else {
        return k_a,b;
}
```

Tabelle 1 : Pseudocode der Schlüsselberechnung beim „direkten Protokoll"

Um eine verschlüsselte Nachricht von Sensor a an Sensor b zu senden (Kanalaufbauphase), wird der Sitzungsschlüssel $k_{a,b}$ wie folgt berechnet: aus den beiden vorherigen Schritten kennt a die Schlüsselindizes aller Schlüssel die b besitzt. Nach einem Durchsuchen der eigenen Schlüsselmenge, kennt a nun alle Schlüsselindizes (und Schlüssel!), die er mit b gemeinsam hat. An dieser Stelle könnte sich auch herausstellen, dass a und b keine Schlüssel teilen und somit eine direkte Kommunikation nicht möglich ist. Falls die gefundene Schnittmenge jedoch nicht leer war, so werden die gemeinsamen Schlüssel nun mittels der XOR-

doi: 10.2313/NET-2011-07-1_12

Operationen verbunden, das Ergebnis ist der Sitzungsschlüssel $k_{a,b}$.

$$k_{a,b} = \bigoplus_{Vi \in Pa \cap Pb} V_i$$

Dieser Vorgang ist in Pseudocode-Implementierung auch nochmals in Tabelle 1 veranschaulicht. Wenn Sensor *b* nun eine Nachricht von a empfängt, berechnet *b* nun auf die gleiche Art und Weise, welche Schlüssel er mit *a* teilt und daraus $k_{b,a}$, was offensichtlich identisch ist mit $k_{a,b}$. Um aber die Verlässlichkeit der Berechnung zu erhöhen, bzw. diese prüfen zu können. Sendet Sensor *a* als erste Nachricht zu *b* eine *key_estimate* Nachricht, welche mittels $k_{a,b}$ verschlüsselt ist. Falls *b* in der Lage ist, diese zu entschlüsseln, also $k_{a,b}$ durch Sensor *a* korrekt berechnet wurde, antwortet *b* mit einer *key_estimate_confirm* Nachricht.

4.2 Das „kooperative Protokoll“

Das oben beschriebene Verfahren hat den Nachteil, dass es nicht resistent gegenüber einer hohen Anzahl korrupter Sensoren ist, da sobald der Angreifer es geschafft hat, die Schnittmenge der Schlüssel von Sensor *a* und *b* in Erfahrung zu bringen (zum Beispiel durch Kapern und Auslesen von Sensoren), er nun in der Lage ist, die gesamte Kommunikation zwischen *a* und *b* abzuhören und zu manipulieren.

Im kooperativen Verfahren wird die Kanalaufbauphase modifiziert. Falls nun Sensor *a* mit Sensor *b* kommunizieren möchte, so wählt *a* eine Menge *C* der Kardinalität *m* von Sensoren

$$C = \{c_1, \ldots, c_m\} \text{ mit } m > 0 \text{ und } a,b \notin C$$

als kooperierende Partner aus, wobei *a* und *b* nicht in *C* enthalten sein dürfen. An jeden Sensor c_i wird eine dementsprechende Anfrage gesendet, wobei diese Kooperationsanfrage mittels $k_{a,ci}$ (gemäß dem direkten Verfahren) verschlüsselt ist und als Nutzdaten die ID von Sensor *b* enthält. Sensor c_i berechnet daraufhin zunächst den gemeinsamen Schlüssel $k_{ci,b}$ (gemäß dem direkten Verfahren) und sendet schließlich einen Hashwert (sei HMAC die gewählte Hashfunktion) aus der ID von *a* und dem Schlüssel $k_{ci,b}$ an den anfragenden Sensor *a* mittels $k_{a,c}$ zurück.

$$HMAC(\mathrm{ID}(a), k_{ci,b})$$

Sobald Sensor *a* alle angefragten Hashwerte vorliegen, bildet er den letztendlichen Sitzungsschlüssel $k^C_{a,b}$ durch Anwenden der XOR-Operation auf Schlüssel $k_{a,b}$ und die empfangenen Hashwerte (auch jeweils durch XOR verbunden).

$$k^C_{a,b} = k_{a,b} \oplus (\bigoplus_{c \in C} \mathrm{HMAC}(\mathrm{ID}(a), k_{ci,b}))$$

Sensor *a* sendet nun zu Beginn der Sitzung an *b* verschlüsselt mittels $k_{a,b}$ die Liste der gewählten Sensoren *C* zusammen mit dem Hashwert des Schlüssels $k^C_{a,b}$. Das Verfahren auf Seite von Sensor *a* ist nochmals in der Pseudocode-Implementierung in Tabelle 2 dargestellt.

Daraufhin kann Sensor *b* aus der übertragenen Sensorliste ohne Versenden weiterer Nachrichten, den Schlüssel $k^C_{a,b}$ berechnen und ihn mittels des mit gesendeten Hashwerts validieren, was in Tabelle 3 mittels einer Pseudocode-Implementierung dargestellt ist.

Da Sensor *b* hierzu mit keinem der Sensoren *C* kommunizieren muss, kann *a* auch Sensoren wählen, die nicht in Sende- und Empfangsreichweite von *b* liegen.

Es ist anzumerken, dass das kooperative Protokoll dem direkten entspricht, falls *m=0* gewählt wird. *m* kann insbesondere der jeweiligen Sicherheitslage entsprechend gewählt werden, sodass auch mit zunehmender Anzahl von korrumpierten Sensoren eine sichere Verschlüsselung gewährleistet werden kann.

```
prepareSend(b: sensorID, c[]: sensorID)
Input: ID des Zielsensors, Liste der kooperierenden Sensoren

k^C_a,b = getSitzungsschlüssel(b);
for each koop_id ∈ c {
        //sendCoopRequest Methode liefere die
        //empfangene Antwort zurück
        answer = sendCoopRequest(koop_id);
        k^C_a,b = k^C_a,b ⊕ answer;
}

//sendet Liste c mittels dem direkten Protokoll an Sensor b
sendViaDirectProtocol(b, getSitzungsschlüssel(b), c);

return k^C_a,b;
```

Tabelle 2 : Pseudocode der Sitzungsschlüsselberechnung auf Verbindungsinitiatorseite beim „kooperativen Protokoll“

```
prepareReceive(a: sensorID, c[]: sensorID)
Input: ID des Sendersensors, Empfange Liste der
kooperierenden Sensoren

k^C_a,b = getSitzungsschlüssel(a);
for each koop_id ∈ c {
        //Folgende zwei Kommandos wurden auch bei
        //jedem kooperierenden Sensor ausgeführt, um
        //die Antwort (answer) zu berechnen
        koop_key = getSitzungsschlüssel(koop_id);
        hashedValue = HMAC(a, koop_key);

        k^C_a,b = k^C_a,b ⊕ hashedValue;
}

return k^C_a,b;
```

Tabelle 3 : Pseudocode der Sitzungsschlüsselberechnung auf Verbindungszielseite beim „kooperativen Protokoll”

Abschließend kann man feststellen, dass sowohl im direkten als auch im kooperativen Protokoll eine sichere Authentifizierung von Sensor *b* durch *a* und umgekehrt gewährleistet werden kann, solange auch eine sichere paarweise Kommunikation zwischen beiden besteht. Da mithilfe des Pseudozufallsgenerators jeder Sensor *a* ermitteln kann, welche Schlüssel ein anderer Sensor mit ID *b* haben muss. Ein Angreifer kann also eine ID *b* nur fälschen und sich mit diesem falschen Sensor erfolgreich bei *a* authentifizieren, wenn dem Angreifer alle Schlüssel bekannt waren, die *a* und der echte Sensor *b* gemeinsam hatten.

doi: 10.2313/NET-2011-07-1_12

5. VERFAHREN VON CHAN ET AL

Chan et al [2] schlagen in ihrem Werk „Random Key Predistribution Schemes for Sensor Networks“ insgesamt drei Verfahren zur Predeployment-Schlüsselverteilung vor. Wobei die im Folgenden zuerst erläuterte „q-Composite Schlüsselverteilungsstrategie“ eine Möglichkeit für den Grundstock der „Multipfad-Schlüsselverstärkung“ bildet, die anschließend kurz erläutert wird. Schlussendlich wird ein drittes Verfahren, die „zufällige, paarweise Schlüsselstrategie“, vorgestellt, welche einen komplett verschiedenen Ansatz zu den vorherigen beiden aufweist, dafür jedoch einen Authentifizierungsmechanismus enthält.

5.1 q-Composite Schlüsselverteilungsstrategie

Das q-Composite Verfahren variiert das zugrundeliegende Verfahren von Eschenauer und Gligor [6] im Grunde nur in der Tatsache, dass für die Kommunikation zwischen zwei Sensoren, verlangt wird, dass diese mindestens *q* Schlüssel gemeinsam haben, d.h. das zugrundegelegte Verfahren erhält man durch Setzen von *q=1*.

Für das bestehende Sicherheitsproblem während der Key Discovery-Phase, dass ein böswilliger Mithörer, die gebroadcasteten Schlüsselindizes nutzen kann, um gezielt Sensoren einzusammeln und auszuwerten, sodass er den vollständigen Schlüsselpool in seinen Besitz bringen kann, wird ein Verfahren vorgeschlagen, dass von Merkle [7] erarbeitet wurde. Hierbei wird nicht der Schlüsselindex im Klartext gebroadcastet, sondern stattdessen wird für jeden Schlüssel *i,* den der Sensor *a* besitzt, ein „Puzzle“ generiert und gesendet. Ein Sensor *b* der in Empfangsreichweite ist und das Puzzle entschlüsseln kann (also durch Probieren seiner eigenen Schlüssel, die empfangene Nachricht dechiffrieren konnte und somit ebenfalls diesen Schlüssel besitzt), sendet daraufhin die passende Antwort zu dem Puzzle zurück. Sensor *a* weiß anschließend über alle Sensoren *B,* welche die richtige Antwort zurückgesendet haben, dass sie den verwendeten Schlüssel *i* kennen.

Nach abgeschlossenem „puzzeln“, kennt jeder Sensor *a* seine Nachbarn *B* mit denen er mindestens *q* Schlüssel gemeinsam hat und mit denen somit eine direkte Kommunikation möglich bzw. erlaubt ist.

Falls Sensor *a* nun eine Nachricht an einen der Sensoren *B* senden möchte, so bildet er aus den (gemeinsamen mit *b*) Schlüsseln einen Hashwert. Dieser Hashwert ist nun der Kanalschlüssel von Sensor *a* und Sensor *b* für alle zu sendenden Nachrichten, d.h. er bleibt über die gesamte Betriebszeit des WSN gleich.

Welche Hashfunktion angewendet wird, wurde von Chan et al nicht genauer spezifiziert und bleibt dem jeweiligen Anwender überlassen.

5.2 Multipfad-Schlüsselverstärkung

In diesem Verfahren nehmen wir an, dass bereits eine paarweise Kommunikation über das Verfahren von Eschenauer und Gligor [6] oder mittels der q-Composite Schlüsselverteilungsstrategie ermöglicht wurde. Allerdings möchte man nun in der Kanalaufbauphase, die eigentlichen Nachrichten besser verschlüsseln, um zum Beispiel sicherzustellen, dass falls ein Angreifer in Besitz der gemeinsamen Schlüssel von Sensor *a* und *b* gelangt ist, damit nicht automatisch die gesamte

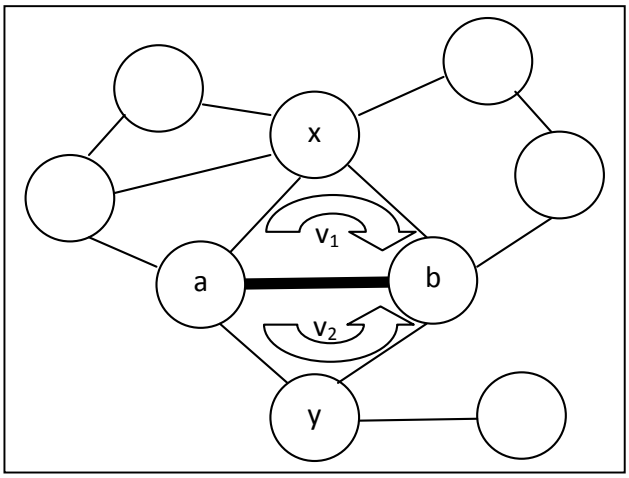

Abbildung 1 : Multipfad-Schlüsselverstärkung Beispiel mit *j=2* und *h=1*

Kommunikation zwischen diesen beiden Sensoren dechiffrieren kann.

Hierzu wird für jeden Kanalaufbau ein eigener Sitzungsschlüssel zwischen Sensor *a* und *b* verabredet. Offensichtlich kann dieser Sitzungsschlüssel nicht einfach direkt von *a* nach *b* gesendet werden, da sonst diese Information im oben genannten Fall für den Angreifer ebenfalls ersichtlich wäre und der Sitzungsschlüssel somit seine Bedeutung und Funktion verloren hätte. Wir nehmen an, es wären genug Routinginformationen bekannt, sodass Sensor *a* alle (Sensor-)disjunkten Pfade (welche in der Key Discovery-Phase entdeckt wurden) von sich zu Sensor *b* kennt, die aus *h* oder weniger Sprüngen (Hops) bestehen. Nehmen wir nun weiter an, es gäbe *j* solcher disjunkten Pfade, so werden *j* Zufallszahlen von Sensor *a* generiert, wobei jede Zufallszahl die gleiche Länge wie die Verschlüsselungsschlüssel aufweist. Jede dieser Zufallszahlen sendet *a* nun über einen anderen Pfad zu Sensor *b*. Wenn *b* alle Zahlen empfangen hat, können Sensor *a* und *b* den Sitzungsschlüssel durch XOR-Anwendung berechnen.

$$k' = k \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_j$$

Für das Finden der Pfade, schlagen Chan et al für den Spezialfall *h=2* die Vorgehensweise vor, dass nach erfolgter Key Discovery-Phase die Sensoren *a* und *b* eine Liste ihrer jeweiligen Nachbarn austauschen. Die Schnittmenge der beiden Listen, sind dann die zu nutzenden Intermediäre. Die Anzahl der Pfade entspricht dann der Kardinalität dieser Schnittmenge. Da bei nur zwei Sprüngen nur genau ein intermediärer Sensor auf dem Pfad liegt, ist es offensichtlich, dass alle so gefundenen Pfade automatisch disjunkt sind. Ein Beispielfall für die Anwendung der Multipfad-Schlüsselverstärkung ist in Abbildung 1 dargestellt, wobei die dick dargestellte Verbindung zwischen Sensor *a* und *b* durch die Teilschlüssel gesichert werden soll. Die zwei Teilschlüssel v_1 und v_2 werden hierbei über die disjunkten Pfade via Sensor *x* bzw. *y* übertragen. Wobei hier angenommen wurde, dass kein Teilschlüssel direkt von *a* nach *b* übertragen wird, da dies wider dem Sinn der Multipfad-Schlüsselverstärkung wäre, worauf im Kapitel „Vergleich“ noch eingegangen wird.

doi: 10.2313/NET-2011-07-1_12

5.3 Zufällige, paarweise Schlüsselstrategie

Die bisherigen Verfahren von Chan et al, besaßen noch nicht die Möglichkeit der Authentifizierung der jeweiligen in Kommunikation stehenden Sensoren. Da im Gegensatz zu dem Verfahren von Di Pietro et al [3] die Schlüssel durch einen, für die Sensoren unbekannten, Zufallsalgorithmus verteilt wurden, sodass es sein kann, dass die Schnittmenge der gemeinsamen Schlüssel von Sensor *a* und *b,* auch in Sensor *c* vorhanden sind und somit keine Möglichkeit für Sensor *a* besteht zu Prüfen, ob er nun wirklich mit Sensor *b* oder Sensor *c* kommuniziert.

Mit der zufälligen, paarweisen Schlüsselstrategie stellen Chan et al nun eine Variante vor, die auch eine Authentifizierung zulässt. Sie basiert auf der naiven Variante, zur Erinnerung: bei einem *n* Sensoren großen Netzwerk hält jeder Sensor *n-1* Schlüssel, wobei jeder Schlüssel in genau zwei Sensoren vorkommt. Das Verfahren führt nun für die einzelnen Sensoren IDs ein, mit jedem Schlüssel der lokal im Sensor gespeichert wird, wird auch die ID des anderen Sensors gespeichert, der denselben Schlüssel besitzt. Die Authentifizierung nutzt also die Tatsache, dass falls eine sichere Kommunikationsverbindung unter Nutzung des Schlüssels *k* zustande kam, dass es sich bei dem Gesprächspartner um den Sensor mit der entsprechenden ID handeln sollte. Aus der in Chan et als Werk zitierten Erdös und Rényis Formel geht hervor, dass eine vollständige Vernetzung des Sensornetzwerks der Größe *n* gegeben ist, in der die kleinste Wahrscheinlichkeit *p* für die Möglichkeit, dass zwei beliebige Sensoren kommunizieren können, sodass der gesamte Sensorknotengraph zusammenhängend ist mit Wahrscheinlichkeit *c*, jeder Sensor nur *np,* statt *n-1* paarweise Schlüssel speichern muss, was dem begrenzten Speicherplatz Rechnung trägt.

In der Predeployment-Phase wird nun ein *n* gewählt, welches der maximalen Größe (bei gleichbleibender Verbindungswahrscheinlichkeit) des Sensornetzwerks entspricht. Es müssen hierbei am Anfang nicht unbedingt auch genauso viele Sensoren ausgebracht werden, sodass Reserven für später hinzuzufügende Sensoren bleibt. Für jeden Sensor *a* wird nun eine ID generiert und mit *np* anderen IDs (Partnern) in Verbindung gebracht. Für jedes so entstandenes Paar aus Sensor a und einem Partnersensor wird nun ein Schlüssel generiert und zusammen mit der ID des Partners in Sensor *a* gespeichert, bzw. zusammen mit der ID von Sensor *a* im Partnersensor gespeichert.

In der Key Discovery-Phase broadcastet jeder Sensor *a* nun seine ID zu seinen unmittelbaren Nachbarn. Die Nachbarn prüfen nun, ob sie die empfangene ID in ihrem Speicher finden, wenn dies der Fall ist, wird der damit im Speicher aufgefundene Schlüssel genutzt, um einen verschlüsselten Handshake mit dem Sendersensor *a* auszuführen. Falls dieser Handshake erfolgreich war, ist sichergestellt, dass *a* auch wirklich der Sensor ist, für den er sich beim Broadcast ausgegeben hat, da er sonst nicht in Besitz des richtigen Schlüssels gewesen wäre, um den Handshake zu dechiffrieren bzw. die Antwort auf den Handshake zu chiffrieren.

6. VERGLEICH

Nachdem nun die von Di Pietro et al und Chan et al entwickelten Verfahren vorgestellt wurden, soll an dieser Stelle ein Vergleich der jeweiligen Verfahren erfolgen mit dem Schwerpunkt auf:

- Overhead im Datenverkehr
- Resistenz gegenüber DoS-Attacken
- Abhörsicherheit

6.1 Overhead im Datenverkehr

Aufgrund der begrenzten Ressourcen in typischen WSNs und des Energiehungers der Funkschnittstelle, sollte der Datenoverhead auf jeden Fall gering gehalten werden. Er ist aufgrund dieser Konstellation auch ein wichtiges Merkmal, der über die Lebensdauer eines einzelnen Sensorknotens entscheidet. Da Datenoverhead auf jeder Ebene der Kommunikation entsteht (OSI-Modell), in diesem Artikel aber nur verschiedene Verschlüsselungsstrategien behandelt wurden, wird hier auch nur der bis zum erfolgten Kanalaufbau entstehende Overhead berücksichtigt. Es ist jedoch anzumerken, dass natürlich die gewählte Schlüssellänge und die Verarbeitungsweise der Daten, also das Verfahren zur De-/Chiffrierung der Nutzdaten mithilfe des Schlüssels auch zum Gesamtoverhead beitragen.

6.1.1 Direktes Protokoll nach Di Pietro et al

Dieses Verfahren weißt den geringsten Overhead auf, da für die Key Discovery-Phase nur einmalig ein Broadcast gesendet werden muss, welches als Nutzdaten ausschließlich die Sensor-ID beinhaltet. Für den Kanalaufbau besteht der Overhead nur in der *key_estimate* und *key_estimate_confirm* Nachricht, welche aber prinzipiell auch weggelassen werden könnte, da sie nur zur Prüfung eingesetzt werden.

6.1.2 Kooperative Protokoll nach Di Pietro et al

Offensichtlich, steigt der Overhead in diesem Verfahren gegenüber dem „direkten Protokoll“ zugunsten der Sicherheit an. Je höher das Sicherheitslevel gewählt wird, also je mehr kooperierende Sensoren mit einbezogen werden, desto größer der Overhead, da an jeden kooperierenden Sensor eine Nachricht mit Aufforderung zur Kooperation und der ID des Zielsensors *b* gesendet werden muss, sowie die ausgewählten Sensoren das Ergebnis an *a* zurücksenden müssen. Folglich zwei Nachrichten pro kooperierenden Sensor plus der Nachricht mit der Liste der kooperierenden Sensoren an Zielsensor *b* zusätzlichen Overhead gegenüber dem „direkten Protokoll“. Hinzukommt, dass für verschiedene Sitzungen verschiedene, kooperierende Sensoren gewählt werden können, sodass das Prozedere dementsprechend pro Sitzung wiederholt werden muss.

6.1.3 q-Composite Strategie nach Chan et at

In diesem Verfahren muss jeder Sensor eine Liste mit seinen Schlüsselindizes senden (also *n* Nachrichten insgesamt), was identisch zur Anzahl bei der Key Discoveryphase im „direkten Protokoll“ nach Di Pietro et al wäre. Um aber die Sicherheit zu erhöhen (s.o.), ist es empfehlenswerter, für jeden Schlüssel den ein Sensor *a* besitzt ein Puzzle zu generieren und zu senden. Hierbei würde jeder Sensor also *K* (= Kardinalität der Schlüsselteilmenge pro Sensor) Nachrichten senden. Da nun aber auf das Puzzle geantwortet werden muss, kommen noch maximal *K* Antworten hinzu. Insgesamt also $(n * k^2)$ Nachrichten, die gesendet werden müssen.

6.1.4 Multipfad-Schlüsselverstärkung nach Chan et al

Dieses Verfahren baut auf einem erfolgten Kanalaufbau auf (deswegen der Name „Verstärkung“), es sind also die schon erzeugten Overheadkosten aufzuaddieren, welche vom gewählten Verfahren abhängig sind. Die zusätzlichen Kosten variieren nach

doi: 10.2313/NET-2011-07-1_12

der Anzahl der Sprünge bzw. der verwendeten Methode zum Finden der Pfade. Legen wir an dieser Stelle das von Chan et al vorgeschlagene Vorgehen zugrunde, so würden alle Sensoren eine Liste mit ihren Nachbarn verschlüsselt an jeden Nachbar senden. Dies wären also *n* Nachrichten. Im Maximalfall hat jeder Sensor *n-1* Nachbarn gefunden. Folglich gäbe es *j=n-2* disjunkte Pfade von Sensor *a* nach Sensor *b*. Damit sendet Sensor *a* folglich *j* Nachrichten und jeder der *j* Sensoren sendet wieder eine Nachricht an Sensor *b*. Insgesamt wären es also insgesamt $(n + (n-2)^2)$ zusätzliche Nachrichten. Damit ist der Overhead des Verfahrens für *h=2* vergleichbar mit dem des „kooperativen Protokolls" von Di Pietro et al. Wobei allerdings auch der eigentlich passive Empfangsknoten zu Beginn eine Liste der Nachbarn senden musste, wohin gegen beim „kooperativen" Protokoll der Zielknoten keine weitere Nachricht senden oder empfangen muss.

6.1.5 Zufällige, paarweise Schlüsselstrategie nach Chan et al

Bei diesem Verfahren haben wir zunächst *m* Nachrichten mit der ID des Sendesensors die gebroadcastet werden. Anschließend muss jeder Sensor für die maximal *m-1* Nachbarn noch einen Handshake ausführen, bestehend aus einer Nachricht plus Antwort. Insgesamt benötigt dieses Verfahren also $(m+ 2*(m-1))$ Nachrichten. Dieses Verfahren ähnelt im Punkt des Datenoverheads folglich am stärksten dem des „direkten Protokolls" von Di Pietro et al.

6.2 Resistenz gegenüber DoS-Attacken

Ein Angreifer kann als Ziel nicht nur das Ausspionieren oder Manipulieren von Daten haben, sondern auch daran interessiert sein, dass Sensornetzwerk lahm zu legen, wie im Artikel von Sen et al [1] ausführlich erläutert wurde, z.B. indem er ein vorzeitiges Erschöpfen der Energieressourcen vorursacht. Eine DoS-Attacke kann zum einen die Luftschnittstelle betreffen (Jamming) oder Angriffsmöglichkeiten auf der Anwendungsschicht adressieren. Da in diesem Paper nur ein Schwerpunkt auf die Verschlüsselungsstrategie bzw. den Kanalaufbau gelegt wird, werden nur DoS-Angriffsmöglichkeiten berücksichtigt, die genau daran ansetzen. Wie wir sehen werden, spielt der Netzwerkoverhead, den die verschiedenen Strategien mit sich bringen, eine nicht unbedeutende Rolle.

6.2.1 Direktes Protokoll nach Di Pietro et al

Während der Key Discovery-Phase sendet jeder Sensor nur einmalig seine ID, d.h. die Häufigkeit mit der eine Key Discovery-Phase von außen eingeleitet werden kann, gibt die obere Schranke für erzwungenen, unnötigen Netzwerkoverhead an. Während der Kanalaufbauphase kann nur durch Senden einer *key_estimate* Nachricht Aktivität erzwungen werden, wobei eine Antwort nur erfolgt, falls es sich um den konformen Schlüssel gehandelt hat. Folglich kann der attackierte Sensor in dieser Phase nur dann zum Senden gezwungen werden, falls eine verschlüsselte Verbindung mit diesem Sensor vorliegt, dann bietet zugleich die Anwendungsschicht meist das effizientere Ziel.

6.2.2 Kooperative Protokoll nach Di Pietro et al

Aufgrund, dass das kooperative auf dem direkten Protokoll aufbaut, erbt dieses Verfahren die soeben vorgestellten Schwachstellen. Nehmen wir hier an, dass in der Menge der gewählten, kooperierenden Sensoren, ein oder mehr Sensoren korrumpiert wären. Der korrumpierte Sensor *c* hat nun drei Möglichkeit auf die Kooperationsanfrage zu reagieren: ignorieren (1), mit dem richtigen Schlüssel $k_{c,b}$ (2) oder dem falschen Schlüssel $k'_{c,b}$ (3) zu antworten, wie bereits Di Pietro et al in Ihrem Artikel [3] ausführten.

Fall (1) würde der Situation entsprechen, falls ein zur Kooperation hinzugezogener Sensor nicht mehr erreichbar wäre (z.B. zerstört oder Energievorrat erschöpft), sodass nach einem gewissen Timeout Sensor *a* einfach einen anderen Sensor wählt. Folglich kann gerade einmal eine Nachricht als unnötigen Overhead erzwungen werden, da sich *a* merken kann, welcher Sensor nicht mehr zur Verfügung stand und ihn für weitere Kooperationsanfragen nicht mehr berücksichtigen muss. Fall (2) würde ein Angreifer wählen, falls er beabsichtigt, die Kanalverschlüsselungssicherheit herabzusetzen, da ihm in diesem Fall ggf. bereits genug Informationen (über andere bei der Kooperation beteiligten Schlüssel) vorliegen, um den Kanal zwischen *a* und *b* nun abzuhören, dazu später mehr. Im Fall (3) würde es im Anschluss Sensor *a* nicht gelingen einen Kanal zu *b* aufzubauen, da der Sitzungsschlüssel von *a* nun nicht mehr mit dem von *b* berechneten übereinstimmt. Gegenmaßnahmen kann auch hier das erneute wählen von anderen Kooperationspartnern sein. Mit diesem Verhalten würde es jedoch für Sensor *a* ersichtlich, dass das Netzwerk angegriffen wird und es können ggf. vorgesehene Gegenmaßnahmen eingeleitet werden, die ggf. die Basisstation mit einbeziehen, um den detektieren, böswilligen Sensor zu isolieren, wie Di Pietro et al in Ihrem Artikel [3] kurz skizzieren.

6.2.3 q-Composite Strategie nach Chan et at

In diesem Verfahren besteht ähnlich zum „direkten Protkoll" nach Di Pietro et al nur die Angriffsmöglichkeit, eine Key Discovery-Phase auszulösen.

6.2.4 Multipfad-Schlüsselverstärkung nach Chan et al

In diesem Verfahren wird ähnlich zum „kooperativen Protokoll" nach Di Pietro et al ein Sitzungsschlüssel vereinbart. Hierbei nehmen wir nun an, dass ein oder mehrere Pfade korrumpierte Sensoren beinhalten. Diese Sensoren haben wieder die bereits oben genannten drei Möglichkeiten: Fall (1) entspricht zwar wieder dem Fall, dass ein gewählter Sensor ausgefallen ist, jedoch kann je nach Pfadlänge, der erzeugte Overhead deutlich größer sein, als beim kooperativen Verfahren, da Sensor *a* nicht direkt mitbekommt, dass ein Teil des Schlüssels nicht bei *b* ankam und somit das Scheitern erst nach Senden einer chiffrierten Nachricht an *b* (welche *b* nun nicht entschlüsseln kann) bemerkt. Desweiteren ist im von Chan et al vorgeschlagenen Verfahren zum Finden der Intermediäre nicht vorgesehen, was passieren soll, wenn einer der Sensoren aus der Schnittmenge sich zwar durch Nachbarschaftserkundung auffinden lässt, aber keine Nachrichten weiterleitet. Sodass dieses Verfahren im Gegensatz zum vergleichbaren „kooperativen Protokoll" von Di Pietro et al, noch weitere Spezifikationen in der Implementierung erfordert, um gegen Attacken geschützt zu sein. Das gleiche gilt für Fall (3), wenn ein falscher Wert weitergeleitet wird. Im Fall (2), d.h. der Sensor hat den korrekten Wert weitergeleitet, besteht die gleiche Auswirkung auf die Abhörsicherheit wie beim „kooperativen Protokoll", mehr dazu später.

doi: 10.2313/NET-2011-07-1_12

6.2.5 Zufällige, paarweise Schlüsselstrategie nach Chan et al

Da dieses Verfahren die Paarbildung bereits in der Predeploymentphase festlegt, gibt es auch praktisch keine Angriffspunkte für eine DoS-Attacke auf der in diesem Artikel behandelten Ebene.

6.3 Abhörsicherheit

Das wohl wichtigste Merkmal einer Verschlüsselungsstrategie, stellt die Abhörsicherheit dar. In Sensornetzwerken, bei welchen die Sensoren meist physisch zugänglich sind und der (teilweisen) Verwendung von Pre-Shared Keys ist dies nochmals eine besondere Herausforderung, da einzelne Sensoren korrumpiert sein können oder durch Einsammeln und Auslesen des Speichers, der Angreifer in Besitz der Pre-Shared Keys gelangen kann.

6.3.1 Direktes Protokoll nach Di Pietro et al

Aufgrund der Bedeutsamkeit der Abhörsicherheit für dieses Thema, wurden im Artikel von Di Pietro et al [3] auch die dort vorgestellten Verfahren auf diesen Punkt hin untersucht. Es ist leicht einzusehen, dass eine Verbindung zwischen Sensor *a* und *b* nicht mehr abhör- und damit auch manipulationssicher, sobald der Angreifer in den Besitz aller Schlüssel, die *a* und *b* gemeinsam haben, gekommen ist. Da dieses Verfahren im Gegensatz zu dem q-Composite Verfahren von Chan et al [2] keine Mindestanzahl an gemeinsamen Schlüsseln vorschreibt, kann es auch sein, dass der Kanal nur durch ein einzelnen Pre-Shared Key verschlüsselt ist.

6.3.2 Kooperative Protokoll nach Di Pietro et al

Im „kooperativen Protokoll“ erhöht sich die Anzahl der verwendeten Schlüssel durch die kooperierenden Sensoren. Nimmt man an, dass die kooperierenden Sensoren nur teilnehmen, falls sie tatsächlich auch einen Schlüssel mit Sensor *b* teilen, so stellt die Anzahl der kooperierenden Sensoren eine untere Schranke für die zusätzlich hinzukommenden Schlüssel dar. Wobei diese untere Schranke nur gilt, falls Sensor *a* eine intelligente Auswahl an Kooperationssensoren getroffen hat. Wobei intelligent hierbei bedeutet, dass der Kooperationssensor mindestens ein Schlüssel mit *b* gemeinsam hat, welcher weder in Sensor *a* noch in einem anderen kooperierenden Sensor vorhanden ist. Das hierfür nötige Wissen kann Sensor *a* aus der Anwendung des Pseudo-Zufallsgenerators beziehen. Di Pietro et al erwähnen für Ihr „kooperatives Protokoll“ noch die Möglichkeit als kooperierende Sensoren nicht nur lokale Nachbarn hinzuziehen, sondern ggf. nur über mehrere Sprünge zu erreichende Sensoren mit einzubeziehen. Dies schwächt die Auswirkungen einer lokal begrenzten Infiltrierung durch korrumpierte Sensoren ein. Da Sensor *b* bei diesem Verfahren weder aktiv noch passiv direkt in Kommunikation mit den Kooperationssensoren treten muss, steigt hierbei auch der Overhead nicht so stark an, wie es bei der „Multipfad-Schlüsselverstärkung“ von Chan et al [2] der Fall wäre. Folglich kann das „kooperative Protokoll“ bei der gleichen Anzahl an Sprüngen weiter entferntere Sensoren mit einbeziehen, als dies bei der „Multipfad-Schlüsselverstärkung“ möglich wäre.

6.3.3 q-Composite Strategie nach Chan et at

Durch die Tatsache, dass zunächst eine gewisse Mindestanzahl *q* an Schlüsseln bei beiden Sensoren gemeinsam vorhanden sein müssen, erhöht zunächst die Sicherheit, wie bereits oben erwähnt. Jedoch muss bei diesem Verfahren die Größe des Schlüsselpools (Predeploymentphase) bei angenommener gleichbleibender Speichergröße in den Sensoren verringert werden. Das hat die Konsequenz, dass durch Einsammeln und Auswerten von Sensoren, der Angreifer bei der gleiche Anzahl an ausgewerteten Sensoren einen größeren Teil des Schlüsselpools in Erfahrung bringen konnte, worauf hin bereits Chan et al in Ihrem Artikel [2] selbst hinweisen. Es läuft hierbei auf den folgenden Trade-Off hinaus: je größer *q* gewählt wird, desto Resistenter ist das Netzwerk gegenüber das Auswerten/Korrumpieren einer kleinen Anzahl an Sensoren, wird jedoch Anfälliger gegen Angriffe/Sabotageakte auf eine Vielzahl von Sensoren.

6.3.4 Multipfad-Schlüsselverstärkung nach Chan et al

Aufgrund dass bei diesem Verfahren, ein völlig von den Pre-Shared Keys unabhängiger Sitzungsschlüssel generiert wird, der aus *j* Teilschlüsseln besteht, wobei jeder Teilschlüssel auf einem Sensordisjunkten Pfad zu Sensor *b* übertragen wird, kann ein Angreifer die Verbindung zwischen Sensor *a* und *b* nur dann erfolgreich abhören, wenn er in Besitz aller *j* Teilschlüssel gelangen ist. Dies erhöht also die Sicherheit auf den ersten Blick enorm, gegenüber allen oben genannten Verfahren, da der Angreifer, selbst wenn er im Besitz des gesamten Schlüsselpools gelangt ist, den Sitzungsschlüssel immer noch nicht vorhersagen kann. Allerdings wäre er somit in der Lage, die generieten Teilschlüssel bei der Übertragung von Sensor *a* erfolgreich abzuhören und somit den Sitzungsschlüssel zu erhalten. Auch wenn der Angreifer nicht alle Pre-Shared Keys des Schlüsselpools kennt, er jedoch in der Umgebung von Sensor *a* und *b* ausreichend viele korrumpierte Sensoren hat, kann er ebenfalls Kenntnis über einen Großteil oder gar den gesamten Sitzungsschlüssel erhalten.

6.3.5 Zufällige, paarweise Schlüsselstrategie nach Chan et al

Diese Strategie bietet die größte Abhörsicherheit an, da die Paarbildung bereits in der Predeploymentphase festgelegt wurde und damit keine Schlüsselinformationen ausgetauscht werden müssen. Selbst wenn ein Angreifer Sensoren auswertet, fallen ihm dabei nur Schlüssel in die Hände, die ausschließlich von diesem Sensor genutzt werden und in keinem weiteren Verbindungspaar zweier Sensoren eine Rolle spielen.

7. ZUSAMMENFASSUNG

Zusammenfassend fällt auf, dass die Verfahren von Di Pietro et al [3] dynamischer wirken als die von Chan et al [2], da aufgrund der Idee der Nutzung eines Pseudozufalls-Zuteilungsgenerator jeder Sensor weiß (bzw. vielmehr berechnen kann), welche Schlüssel (anhand der Indizes) ein anderer Sensorknoten aufweisen muss. Wobei hierfür nur der Algorithmus und der Initialisierungswert gespeichert werden müssen. Ein weiterer großer Vorteil der Verfahren von Di Pietro et al, betrifft die Authentifizierung, welche einem durch das Verfahren praktisch geschenkt wird. Allerdings könnte man das „direkte Protokoll“ von Di Pietro um eine Mindestanzahl an gemeinsamen Schlüsseln erweitern, indem man die Forderung aus dem „q-Composite“-Verfahren von Chan et al mit übernehmen würde. Gegebenenfalls erweitern könnte man das direkte oder kooperative Protokoll, um die Nutzung eines Sitzungsschlüssels, der unabhängig von den Pre-Shared Keys ist, z.B. unter Einbeziehung des

doi: 10.2313/NET-2011-07-1_12

Schlüsselverstärkung nach Chan et al auf einer höheren Protokollebene über dem direkten/kooperativen Protokoll. Die Dynamik der Protokolle von Di Pietro et al zeichnet sich auch gerade in mobilen WSN aus, bei der sich die Nachbarn der Sensoren ständig ändern und abwechseln, nicht zuletzt auch, da die Key Disovery-Phase aus nur einem einzigen Broadcast (pro Sensor) besteht.

Für statische Sensornetzwerke dürfte die „zufällige, paarweise Strategie“ von Chan et al, klar im Vorteil sein, da sie für diesen Einsatzzweck, den geringsten Overhead mit sich bringen würde (gut für die Lebensdauer der Energievorräte) und auch beim Thema Abhörsicherheit durch ihre Effizienz besticht.

Generell gilt jedoch für probabilistische Verfahren, dass es mit einer gewissen Wahrscheinlichkeit passieren kann, dass einzelne Sensoren, obwohl sie in Funkreichweite mit ein oder mehreren Nachbarn sind, nicht in das Netzwerk integriert werden können, da sie mit ihren unmittelbaren Nachbarn keine Schlüssel teilen (bzw. nicht ausreichend viele, je nach Verfahren). Dies sollte bei der Planung der Funkreichweite bzw. der Sensordichte, also der Abschätzung der durchschnittlichen Anzahl an Nachbarn, für die Ausbringung berücksichtigt werden.

8. LITERATUR

[1] Jaydip Sen, „A Survey on Wireless Sensor Network Security”, International Journal of Communication Networks and Information Security (IJCNIS), Vol. 1, No. 2, August 2009

[2] Haowen Chan, Adrian Perrig und Dawn Song, „ Random Key Predistribution Schemes for Sensor Networks”, Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP’03), 2003

[3] Roberto Di Pietro, Luigi V. Mancini und Alessandro Mei, „Random KeyAssignment for Secure Wireless Sensor Networks”, Proceedings of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks Fairfax, Virginia, 2003

[4] Sasikanth Avancha, Jeffrey Undercoffer, Anupam Joshi und John Pinkston, „Security for Wireless Sensor Networks”, Kluwer Academic Publishers Norwell, MA, USA, 2004

[5] Djamel Djenouri und Lyes Khelladi, „A survey of security issues in mobile ad hoc and Sensor Networks“, IEEE Communications Surveys & Tutorials, Fourth Quarter, 2005

[6] Laurent Eschenauer undVirgil D. Gligor, „A key-management scheme for distributed sensor networks“, Proceedings of the 9th Computer Communication Security (CCS 2002), Washington, MA, November 2002

[7] R. Merkle, „Secure communication over insecure channels“, Communications of the ACM, 21(4):294–299, 1978

doi: 10.2313/NET-2011-07-1_12