Sicherheit durch Hardwareeinbindung - secFleck

Thomas Riedmaier
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: riedmaie@in.tum.de

KURZFASSUNG

Drahtlose Sensornetzwerke wie das Fleck™-WSN halten Einzug in immer mehr Anwendungsgebiete. Im Generellen befinden sich die Sensorknoten hierbei in einer unbewachten Umgebung und sind deshalb zahlreichen Angriffsszenarien ausgesetzt. Das in dieser Arbeit vorgestellte secFleck-System begegnet dieser Problematik, indem es das Standard-Fleck™ -System um einen TPM-Chip erweitert. Dank diesem ist es möglich sowohl asymmetrische als auch symmetrische Verschlüsselungsroutinen effektiv auf einem Sensorknoten zu berechnen. Diese Kombination ermöglicht eine Reihe von Sicherheitsfeatures, von denen einige - wenn auch bei weitem nicht alle - in der secFleck-API realisiert wurden. Viele Features, die der TPM-Chip zur Verfügung stellen würde, bleiben jedoch ungenutzt.

Schlüsselworte

Fleck, SecFleck, Trusted Platform Module (TPM), Wireless Sensor Network (WSN)

1. EINLEITUNG

Drahtlose Sensornetzwerke (WSN) haben schon lange ihre militärischen Wurzeln hinter sich gelassen und werden inzwischen auch zu rein zivilen Zwecken genutzt. Zu diesen gehören unter Anderen wissenschaftliche Forschungsprojekte wie das sog. „Habitat monitoring“ [25], welches Ökologen hilft Einflüsse auf Lebensräume besser verstehen zu lernen. Überdies kommen WSNs auch in kommerziellen Anwendungen wie der Überwachung von Bergwerken zum Einsatz [18].

Da die Sensorknoten eines Netzwerkes typischerweise nur über sehr beschränkte Energieressourcen und Rechenkapazitäten verfügen, wurde bisher dem Sicherheitsaspekt von WSNs nur eine geringe Bedeutung beigemessen. Gerade in den kommerziellen Anwendungen ist die Sicherheit der Systeme jedoch ein verkaufskritisches Feature. Hierbei geht es nicht nur um die Vertraulichkeit eventuell geschäftskritischer Daten, sondern auch um die Glaubwürdigkeit der vom Sensornetzwerk ermittelten Daten. Ein von einem Angreifer absichtlich ausgelöster falscher Alarm könnte beispielsweise mehr Kosten verursachen als der Einsatz des WSNs an Kosten einspart.

Es liegt jedoch in der Natur der drahtlosen Sensornetzwerke, dass die Sensorknoten weit verteilt und deshalb oft unbewacht installiert sind. Dies setzt das Netzwerk einer Reihe von Angriffen aus, die sich nicht nur auf das Eindringen in den Funkverkehr beschränken. Auch der Fall, dass einzelne Knoten unauthorisiert umprogrammiert werden ist nicht auszuschließen.

Diese Arbeit stellt mit secFleck eine auf dem TPM-Chip basierende Technologie vor, die Lösungen für eine Reihe von brennenden Sicherheitsproblemen der WSN-Technologie verspricht. Sie gliedert sich dabei wie folgt:

Zunächst wird in Kapitel 2 mit Fleck™ diejenige Drahtlos-Netzwerk-Technologie vorgestellt, aus der das secFleck-System hervorgegangen ist. Anschließend wird in Kapitel 3 eine generelle kurze Einführung in die Sicherheit verteilter Systeme gegeben, welche in der Vorstellung der TPM-Technologie mündet. Auf diesen Grundlagen aufbauend wird daraufhin in Kapitel 4 secFleck vorgestellt und evaluiert. Abgeschlossen wird mit einem kurzen Überblick über verwandte Arbeiten in Kapitel 5 sowie einem Ausblick in die Zukunft in Kapitel 6.

2. FLECK

Fleck™ [22] bezeichnet eine aus der Zusammenarbeit zwischen dem australischen CSIRO [1] und der Datacall® Company [2] hervorgegangene Drahtlos-Sensor-Netzwerk-Technologie, deren Platine in Abb. 1 zu sehen ist. Sie ermöglicht das Sammeln von Informationen über große Gebiete hinweg, und bietet so beispielsweise für große Agrar- oder Minengesellschaften eine einfache aber dennoch verlässliche Möglichkeit ihren Betrieb zu überwachen [26].

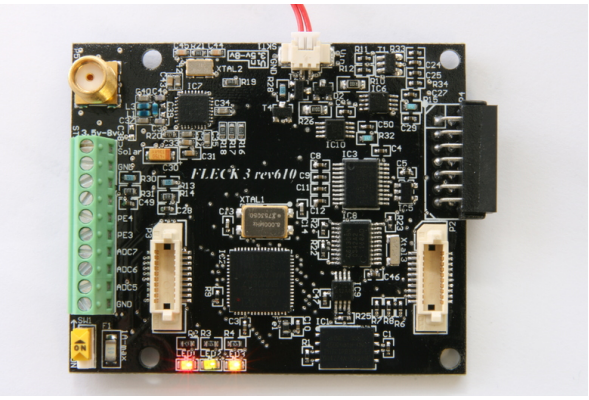

Abbildung 1: Fleck3 Platine mit Atmega128 Microcontroller und nRF905 Multiband Transceiver [27].

Gegenüber klassischen telemetrischen Sensorsystemen bietet Fleck™ zahlreiche Vorteile: Kosteneffizienz, einfache „Plug-and-Play“(PnP)-Installation des Systems sowie eine gesteigerte Zuverlässigkeit durch Selbstheilungsfähigkeiten des Netzwerkes. Der Aspekt mit dem das Fleck™-System jedoch am meisten gegenüber seinen Konkurrenzprodukten punkten kann, ist die geringe Stromaufnahme seiner Sensorknoten. Diese sind so gefertigt, dass sie mit drei AA-Akkus betrieben werden können, welche tagsüber durch ein kleines Solarpanel aufgeladen werden [26].

Damit Anwendungsprogrammierer sich nicht direkt mit der Hardware auseinandersetzen müssen, sind die Fleck™-Sensorknoten mit dem Fleck Operating System (FOS) [5] ausgestattet. Anders als das weitverbreitete TinyOS setzt FOS nicht auf Eventhandler um die Ressourcen auf verschiedene Programme aufzuteilen, sondern auf Threads. Dies ermöglicht eine einfache, intuitive Programmierung mit einem POSIX-ähnlichen Interface. FOS stellt zudem sicher, dass der Energieverbrauch des Systems auf ein Minimum beschränkt bleibt [5].

3. SICHERHEIT IN DRAHTLOSEN SENSORNETZWERKEN

Wie bei den meisten technischen Systemen gilt auch bei WSN-Systemen die Faustregel: Je komfortabler ein System zu benutzen ist, desto unsicherer wird es. Da es sich beim Fleck™-System um ein kommerzielles Produkt handelt, waren seine Entwickler bemüht, ein möglichst komfortabel benutzbares System zu erschaffen. Dies zeigt sich u. a. im Fokus auf die PnP-Funktionalität, welche es sehr einfach macht, ein Fleck™-System aufzubauen bzw. zu erweitern. Diese Einfachheit bedeutet jedoch auch, dass es einem Angreifer sehr einfach möglich ist in das System einzudringen, indem er beispielsweise seinen eigenen Sensorknoten in das System einschleust oder den Datenverkehr mitschneidet [12].

Im Gegensatz dazu haben die meisten Systembetreiber jedoch ein nicht zu vernachlässigendes Sicherheitsbedürfnis. Im Falle eines WSN lässt sich dieses Bedürfnis meist auf folgende Schutzziele zusammenfassen [10]:

- Informationsvertraulichkeit
- Verbindlichkeit von Nachrichten
- Datenintegrität

Klassischerweise wird die Einhaltung dieser Schutzziele durch die Verwendung kryptographischer Methoden sichergestellt. Im Nachfolgenden seinen diese kurz vorgestellt:

3.1 Symmetrische Verschlüsselungsverfahren

Charakterisierend für symmetrische Verschlüsselungsverfahren ist die Tatsache, dass die zur Ver- bzw. zur Entschlüsselung von Nachrichten verwendeten Schlüssel gleich, oder zumindest leicht voneinander ableitbar sind. Um den Inhalt verschlüsselter Nachrichten vor Angreifern zu schützen ist es deshalb nötig, beide Schlüssel geheim zu halten. Daraus ergibt sich jedoch auch gleich das größte Problem symmetrischer Verschlüsselungsverfahren: die Schlüsselverteilung, oder anders formuliert: „Wie ist es dem Sender möglich, dem berechtigten Empfänger (und nur diesem) den Entschlüsselungsschlüssel zukommen zu lassen?“ [10].

Im Falle von eingebetteten Systemen wird das Schlüsselverteilungsproblem oft sehr pragmatisch dadurch gelöst, dass den einzelnen Komponenten noch vor ihrem Einsatz die benötigten Schlüssel eingespeichert werden. Wurde auf diese oder eine andere Weise das Problem gelöst, ist es allen beteiligten Komponenten möglich eine vertrauliche Kommunikation zu führen.

Populäre Vertreter der symmetrischen Verschlüsselungsverfahren sind beispielsweise der „Data Encryption Standard“ (DES) [23], der „Advanced Encryption Standard“ (AES) [4] oder das speziell für eingebettete Systeme geeignete „Extended Tiny Encryption Algorithm“ (XTEA) [20].

3.2 Asymmetrische Verschlüsselungsverfahren und Signaturen

Anders als die symmetrischen Verschlüsselungsverfahren verwenden die asymmetrischen Verschlüsselungsverfahren separate Ver- und Entschlüsselungsschlüssel.

Kernidee der asymmetrischen Verschlüsselung ist es, dass sich jeder Kommunikationspartner ein Schlüsselpaar, bestehend aus einem privaten und einem öffentlichen Schlüssel generiert. Die Schlüsselpaare werden auf eine Weise generiert, die sicherstellt, dass sich Informationen, welche mit dem öffentlichen Schlüssel verschlüsselt wurden, ausschließlich mit dem privaten Schlüssel entschlüsseln lassen und umgekehrt. Der öffentliche Schlüssel wird nach der Erzeugung mit dem Namen des Erstellers versehen und der Allgemeinheit zur Verfügung gestellt. Dies stellt kein Sicherheitsrisiko dar, da es bei sicheren asymmetrischen Verschlüsselungsverfahrens nicht möglich ist, lediglich auf Grund der Kenntnis des öffentlichen Schlüssels den privaten Schlüssel effizient zu berechnen. Vielmehr stellt dieses Vorgehen eine sehr elegante Möglichkeit dar das Schlüsselverteilungsproblem zu lösen [10].

Im direkten Vergleich mit den symmetrischen Verfahren schneiden die asymmetrischen Verfahren (wie z.B. RSA [21]) eher schlecht ab: Sie lassen sich bei gleicher Schlüssellänge leichter brechen, können oft nicht so effizient in Hardware umgesetzt werden und benötigen mehr Zeit für Berechnungen [10]. Im Falle von eingebetteten Systemen sind die beiden letzten Faktoren kritisch, da Rechenzeit mit Stromverbrauch einhergeht und somit möglichst zu minimieren ist.

Die Stärken der asymmetrischen Verfahren liegen auf den Gebieten, auf denen die symmetrischen Verfahren keine Lösung anbieten, wie es z.B. beim der Schlüsselverteilungsproblem der Fall ist. Gerade in WSN-Systemen, in denen dynamisch Knoten hinzukommen oder entfernt werden, kann so ein Grad an Sicherheit erreicht werden, der durch das einmalige Einspeichern von Schlüsseln in die Hardware niemals erreicht werden könnte.

Zudem stellt ein mit einem privaten Schlüssel verschlüsseltes - man sagt auch signiertes - Datum die Information dar, dass der Besitzer dieses Schlüssels das Datum kannte, denn nur er konnte es verschlüsseln. Handelt es sich bei diesem Datum beispielsweise um eine Nachricht, so kann also von

jeder beliebigen Person geprüft werden, ob die Nachricht dem Signierer vorgelegen hat. Durch Signaturen lässt sich also verbindlich nachweisen, aus welcher Quelle eine Nachricht stammt [10].

Um die Vorteile sowohl der symmetrischen als auch der asymmetrischen Kryptographie ausnutzen zu können wurden sogenannte hybride Verfahren entwickelt. Diese Verfahren verwenden zunächst ein asymmetrisches Verfahren, um symmetrische Schlüssel auszutauschen. Die darauf folgende Kommunikation wird dann mit den eben ausgetauschten Schlüsseln symmetrisch verschlüsselt [10].

3.3 Hashfunktionen

Unter einer Hashfunktion versteht man im mathematischen Sinne eine Einwegfunktion, welche ein Eingabedatum beliebiger Größe auf einen Bitstring fester Länge abbildet. Für kryptographisch starke Hashfunktionen wie dem SHA-1 [8] ist es u.a. nicht möglich, in realistischer Zeit

- zu einem gegebenen Hash H ein Datum zu bestimmen, welches gehasht H ergibt [10].
- zwei verschiedene Eingabewerte zu finden, welche gehasht denselben Bitstring erzeugen [10].

Ein beliebtes Einsatzgebiet für Hashfunktionen ist die Sicherstellung der Integrität von über ein Netzwerk übertragenen Nachrichten. Ein mögliches Vorgehen hierfür ist, dass der Sender zunächst zu einer Nachricht M ihren korrespondierenden Hashwert H(M) bestimmt. Anschließend wird dieser Wert unter Benutzung eines asymmetrischen Verfahrens signiert und an die Nachricht angehängt. Wird die Nachricht während der Übertragung verändert, so kann dies vom Empfänger erkannt werden [10].

3.4 TPM und Trusted Computing

Da die in Abschnitt 3 genannten Sicherheitsschutzziele auch in großen, potentiell nicht gänzlich vertrauenswürdigen Systemen gewährleistet werden sollen (Trusted Computing), werden viele Systeme heutzutage mit Trusted Plattform Modulen (TPM) ausgestattet. Im Wesentlichen handelt es sich hierbei um ein Subsystem mit geschütztem Speicher und der Möglichkeit sicherheitskritische Operationen in Hardware durchzuführen. Die für diese Arbeit zentralen Funktionen sind [24]:

- **Tcsip_GetRandom:** Erzeugen einer starken Zufallszahl.
- **Tspi_Data_Bind:** Asymmetrisches Verschlüsseln von Daten z.B. mit RSA (vgl. Abschnitt 3.2).
- **Tspi_Data_Unbind:** Entschlüsseln Asymetrisch verschlüsselter Daten (vgl. Abschnitt 3.2).
- **Tspi_Hash_Sign:** Hasht ein Objekt und signiert das Ergebnis (vgl. Abschnitte 3.2, 3.3).
- **Tspi_Hash_VerifySignature:** Verifiziert, ob der Hash eines Objekts identisch ist mit dem, der in seiner Signatur steht (vgl. Abschnitte 3.2, 3.3).
- **Tcsip_PcrRead:** Auslesen der PCR Register (vgl. Abschnitt 3.4.2).
- **Tcsip_Seal:** Verschlüsseln mit binden an den TPM-Chip. Selbst dieser kann die Daten nur dann wieder entschlüsseln, wenn sich das System im passenden Zustand befindet (korrekter Wert des PCR-Registers).
- **Tcsip_Unseal:** Entschlüsseln versiegelter Daten.

Lediglich die asymmetrische Verschlüsselungsfunktionalität ist Teil derjenigen Spezifikation, die ein TPM-Chip erfüllen muss. Symmetrische Verfahren müssen nicht notwendigerweise unterstützt werden, weshalb sie auch oft nicht angeboten werden [24].

Der TPM-Chip stellt seiner Plattform neben den kryptographischen Funktionen eine Reihe von Features zur Verfügung, welche im Folgenden kurz erläutert werden.

3.4.1 Überprüfbare anonyme Identitäten

Jeder TPM-Chip ist in der Lage beliebig viele asymmetrische Schlüsselpaare zu erzeugen, von denen zwar überprüft werden kann, ob sie von einem dem Standard entsprechenden TPM-Chip generiert wurden, aber nicht von welchem. Indirekt ist somit überprüfbar, ob ein Schlüsselpaar vorliegt, dessen privater Schlüssel nur *einem einzigen* Chip bekannt ist. Diese Schlüsselpaare werden „Attestation Identity Key Pairs“ (AIKs) genannt [19].

Das Prinzip nach dem diese Schlüssel erzeugt werden lässt sich in wenigen Sätzen erklären: Bei der Fertigung wird jedem TPM-Chip sowohl ein sog. „Endorsement Key Pair“ (EK), als auch ein vom Hersteller unterschriebenes Zertifikat eingespeichert, das versichert, dass besagter EK einem standardkonformen TPM-Chip gehört. Da der EK nicht selbst zum Verschlüsseln von Daten verwendet werden darf generiert sich der TPM-Chip ein AIK Paar. Den öffentlichen Schlüssel dieses AIKs signiert der Chip mit seinem EK und schickt das Ergebnis zusammen mit seinem EK-Zertifikat an eine beliebige „Privacy-Certification Authority“ (P-CA). Diese kann dank dem vom Hersteller unterschriebenen EK-Zertifikat ein weiteres Zertifikat ausstellen, das besagt, dass der AIK wirklich einem standardkonformen TPM-Chip gehört. Da die P-CA nur dann signiert, wenn sie dem Hersteller vertraut, kann daraufhin jeder, der der P-CA vertraut, dem AIK vertrauen [19].

Diese Funktionalität kann in Sensornetzwerken von großem Vorteil sein. So ist beispielsweise ein Szenario vorstellbar, in dem Personen mit Positionssensoren ausgestattet werden. Diese Daten werden durch das Netzwerk zu einer Basisstation geschickt, wo sie ausgewertet werden. Theoretisch wäre es einem manipulierten Sensorknoten also möglich Bewegungsprofile von jedem anderen Sensor anzufertigen und einer Person zuzuordnen. Werden jedoch statt einem Knotenbezogenem Schlüssel AIKs verwendet, die von der Basisstation unterschrieben wurden, so ist nur noch diese zu dieser Zuordnung in der Lage.

3.4.2 Beglaubigter Bootvorgang

Jeder TPM-Chip verfügt über eine Anzahl von „Platform Configuration Registers“ (PCRs), in denen eine Verknüpfung von Messergebnissen über den Zustand des Systems gespeichert wird. Diese Register befinden sich in einem geschützten Speicherbereich und können nur durch den Neustart des Systems zurückgesetzt werden. Typischerweise wird in diesen Registern der Bootvorgang mitprotokolliert. Konkret bedeutet dies, dass jede am Bootvorgang beteiligte Komponente zuerst von der Vorherigen gehasht wird, das Ergebnis den PCR-Registern hinzugefügt wird und danach erst die Kontrolle an die Komponente übergeben wird (Abb. 2) [19].

Die Methodik, nach der die PCRs modifiziert werden basiert auf der kryptographisch starken Hashfunktion SHA-1 (vgl. Abschnitt 3.3). Aus diesem Grund ist es einer modifizierten - also pot. schädlichen - Komponente nicht möglich ihre Andersartigkeit gegenüber dem Standard im Nachhinein zu verschleiern. Stimmt hingegen der Wert der PCRs am Ende des Bootvorgangs, so kann von einem integeren System ausgegangen werden [19].

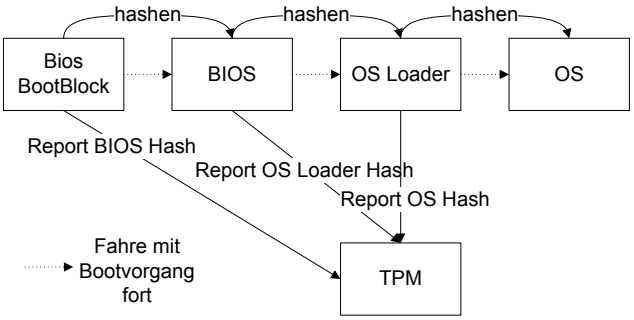

Abbildung 2: Beglaubigter Bootvorgang nach [19]

Das TPM bietet mehrere Möglichkeiten für die Verwendung des Wertes des PCRs. Einerseits kann das TPM dazu verwendet werden, Daten wie z.B. Passwörter nur dann zur Verfügung zu stellen, wenn der Wert des PCR korrekt ist (Seal bzw. Unseal). Andererseits kann es - und das ist gerade in Sensornetzwerken wichtig - auch dazu verwendet werden, die Integrität eines Systems anderen gegenüber zu bescheinigen. Dazu wird der PCR-Wert mit einem AIK unterschrieben und das Ergebnis z.B. der Basisstation zugesendet. Da der AIK nie das TPM verlassen hat kann dieser „Report on integrity“ von niemandem gefälscht worden sein [19].

3.4.3 Sicherer Speicher

TPMs können dazu verwendet werden, Daten sicher zu speichern. Hierzu werden die zu sichernden Daten mit einem Schlüssel verschlüsselt, der nur dem TPM selbst bekannt ist. Zugriff auf diese Daten erhält dann nur, wer sich dem TPM gegenüber als berechtigt ausweisen kann. Zusätzlich können Daten auch so geschützt werden, dass das TPM die Daten nur dann entschlüsselt, wenn nicht nur die Berechtigung korrekt ist, sondern auch der Systemzustand (PCR) [19].

Da sich Sensorknoten potentiell in nicht vertrauenswürdigen Umgebungen befinden, besteht gerade hier ein hohes Bedürfnis nach solchen sicheren Speichermöglichkeiten.

4. SECFLECK

Jeder Fleck^{TM}-Sensorknoten verfügt standardmäßig über einen Erweiterungssteckplatz. Dieser ist dafür gedacht Umgebungssensoren an das System anzuschließen - kann jedoch prinzipiell auch für andere Erweiterungen genutzt werden. Diese Tatsache haben sich Wen Hu et al. in [12] zunutze gemacht und den TPM v1.2 standardkonformen TPM-Chip Atmel AT97SC3203S angeschlossen. Ihr Ziel war es, ein auf der Fleck^{TM}-Platine bzw. dem FOS aufsetzendes sicheres WSN zu schaffen, dessen

- Sicherheitslevel dem des E-Commerce im Internet entspricht [12].
- Geschwindigkeit dem des normalen Fleck^{TM}nicht wesentlich nachsteht [12].
- Energieverbrauch die Möglichkeiten eines drahtlosen Sensorknotens nicht übersteigt [12].
- Kosten im überschaubaren Bereich liegen [12].

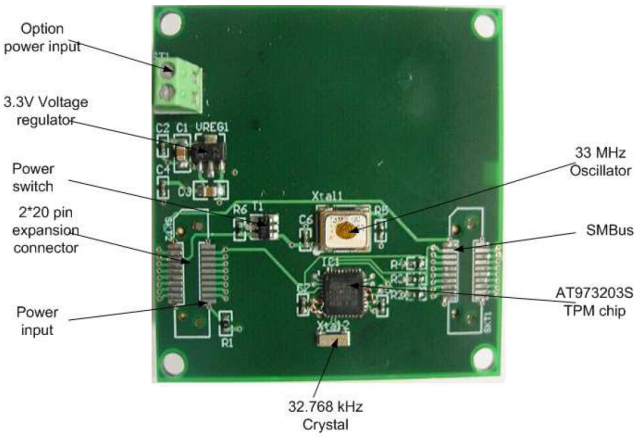

Abbildung 3: secFleck Erweiterungsplatine [12]

Das daraufhin entwickelte System haben die Autoren auf den Namen secFleck getauft. Dieses besteht sowohl aus der Hardware der Erweiterungsplatine (Abb. 3), als auch aus der Erweiterung des FOS um eine Schnittstelle zu Funktionen des TPMs [12].

4.1 Evaluation der Designziele

Im Folgenden wird dargelegt in wie weit und auf welche Weise die in Abschnitt 4 geforderten Designziele erreicht werden konnten. Die Evaluation beginnt mit dem Sicherheitslevel, geht über die Geschwindigkeit des Systems und seinen Energieverbrauch bis hin zu seinen Kosten.

4.1.1 Sicherheitslevel

secFleck stellt an sich selbst den Anspruch, dem Anwendungsprogrammierer sowohl symmetrische als auch asymmetrische Kryptographie zur Verfügung zu stellen. Da TPM-Chips wie bereits in Abschnitt 3.4 erwähnt keine symmetrische Kryptographie anbieten, musste diese in Software implementiert werden.

Die Wahl der Entwickler fiel auf den „eXtended Tiny Encryption Algorithm“ (XTEA) mit einer Schlüssellänge von 128 Bits, da dieser mit nur wenig Hauptspeicher auskommt und somit besonders für Sensorknoten geeignet ist. Überdies gilt der originale XTEA-Algorithmus derzeit als ungebrochen und somit als hinreichend sicher [12].

Die Erzeugung symmetrischer Schlüssel ist eine kritische Aufgabe. Wird hierzu lediglich ein Pseudozufallszahlengenerator eingesetzt, so ist es einem Angreifer, der den Initialwert des Generators bestimmen konnte möglich sämtliche mit diesem Generator erzeugte Schlüssel zu brechen. Um dieser Problematik zu entgehen setzt secFleck auf den TPM-Chip, welcher in der Lage ist starke Zufallszahlen zu erzeugen [12].

Als asymmetrisches Kryptographieverfahren wählten die Entwickler das RSA Verfahren aus. Hierzu ist anzumerken, dass die Sicherheit des RSA-Verfahrens von zwei Faktoren abhängt: Einerseits von der Schlüssellänge k und andererseits vom Exponenenten des öffentlichen Schlüssels, dem sog. „e-Wert“ [21]. Beide beeinflussen die Sicherheit des Verfahrens negativ, wenn sie zu klein gewählt werden. Die derzeitige Empfehlung für die minimale Schlüssellänge liegt bei 1369 Bit. Um darüber hinaus eine gewisse Zukunftssicherheit zu erreichen, wird diese deshalb gerne zur nächst größeren Zweierpotenz von 2048 Bit aufgestockt [17]. Der „e-Wert“ sollte nach Empfehlung der Entwickler nicht kleiner sein als $\log_2$(Schlüssellänge) [21]. SecFleck erfüllt beide Kriterien, indem es eine Schlüssellänge von 2048 Bits verwendet und einen „e-Wert“ des öffentlichen Schlüssels von 65.537 und kann somit ebenfalls als hinreichend sicher angenommen werden [12].

All diese Funktionen stellt secFleck dem Nutzer in einer API zur Verfügung, welche in Abbildung 4 dargestellt wird.

4.1.2 Geschwindigkeit

Den RSA Algorithmus mit den in Abschnitt 4.1.1 geforderten Werten für die Schlüssellänge bzw. für den „e-Wert“ zu betreiben ist für Sensorknoten mit ihrer beschränkten Rechenleistung eine Herausforderung. So würde das einmalige Durchführen einer Verschlüsselung mit 2048 Bit und einem „e-Wert“ von 65.537 auf einem Fleck^{TM}-System mehr als 7 Minuten benötigen und somit unpraktikabel werden (Tabelle 1) [12].

Für dieses Problem gibt es nun theoretisch zwei mögliche Lösungen: Reduzierung der Größe der Verschlüsselungsparameter mit Inkaufnahme von Sicherheitsrisiken oder - wie es von den secFleck-Entwicklern in [12] gemacht wurde - Anbindung eines kryptographischen Koprozessors. Dieser ist in der Lage, die Berechnung in Hardware durchzuführen, und ermöglicht so asymmetrische Kryptographie nahezu „in Echtzeit“ anzuwenden (vgl. Tabellen 1 und 2). Bei dem

```
1 /* Duty cycle TPM chip functions. */
2 uint8_t fos_tpm_startup(void);
3 uint8_t fos_tpm_turnoff(void);
4
5 /* True random number generator. */
6 uint8_t fos_tpm_rand(uint8_t *randNumber,
      uint8_t len);
7
8 /* secFleck public key collector. */
9 uint8_t fos_tpm_getPubKey(uint8_t *pubKey);
10
11 /* Asymmetric key encryption/decryption. */
12 uint8_t fos_tpm_encryption(uint8_t *msg,
      uint16_t len, uint8_t *pubKey, uint8_t *
      cipher);
13 uint8_t fos_tpm_decryption(uint8_t *cipher,
      uint8_t *msg, uint16_t *len);
14
15 /* Digital signature and verification. */
16 uint8_t fos_tpm_sign(uint8_t *digest, uint8_t
      *signature);
17 uint8_t fos_tpm_verifySign(uint8_t *signature
      , uint8_t *pubKey, uint16_t *digest);
18
19 /* Symmetric session key encryption/
      decryption. */
20 uint8_t fos_xtea_encipher(uint8_t *msg,
      uint8_t *key, uint8_t *cipher, uint8_t
      nRounds);
21 uint8_t fos_xtea_decipher(uint8_t *cipher,
      uint8_t *key, uint8_t *msg, uint8_t
      nRounds);
```

Abbildung 4: secFleckAPI [12]

Public Exponent (e)	Software 1024 bit	Software 2048 bit	Hardware 2048 bit
3	0,45s	65s	N/A
65.537	4,185s	450s	0,055s

Tabelle 1: Vergleich von Verschlüsselungszeiten bei RSA [12].

kryptographischen Koprozessor handelt es sich wie oben bereits erwähnt um ein TPM, dessen interne Funktionen durch die secFleck API (vgl. Abb. 4) dem Anwendungsentwickler zugängig gemacht werden.

Um einen vollständig ungestörten Betrieb trotz des Einsatzes von Kryptographie zu gewährleisten ist es notwendig, dass Daten schneller ver-/entschlüsselt werden können als sie über das Netzwerk gesendet werden. Den Autoren in [12] zufolge benötigt der NRF905 Transceiver 23,75μs um ein Bit zu versenden, während die asymmetrische Verschlüsselung eines Bits selbst mit Hardwareunterstützung noch 27μs benötigt. Da jedoch keine Notwendigkeit dafür besteht, Daten asymmetrisch zu verschlüsseln, sobald ein initialer symmetrischer Schlüssel übertragen wurde („Hybride Verschlüsselung“ Abschnitt 3.2), kann alternativ auch der XTEA eingesetzt werden. Dieser benötigt für die Verschlüsselung eines Bits nur noch 18μs, und ist somit signifikant schneller als der Transceiver (vgl. Tabelle 2).

Plattform	Stromstärke (mA)	Zeit (μs)	Energie (μJ)
RSA SW, 2048 bit	8,0	219,730	7.030,0
RSA HW, 2048 bit	50,4	27	5,4
XTEA SW, 128 bit	8,0	18	0,6

Tabelle 2: secFleck Stromaufnahme und Zeitverbrauch für die Verschlüsselung *eines* Bits [12].

Modul	Stromstärke (mA)
Fleck3 (Leerlauf)	8,0
Fleck3 + Empfangen	18,4
Fleck3 + Senden	36,8
Fleck3 + TPM verschlüsseln	50,4
Fleck3 + TPM entschlüsseln	60,8
Fleck3 + TPM signieren	60,8
Fleck3 + TPM Signatur prüfen	50,4

Tabelle 3: secFleck Stromaufnahme [12]

4.1.3 Energieverbrauch

Wie Tabelle 3 entnommen werden kann kostet der Betrieb des secFlecks mit aktiviertem TPM zehnmal mehr Energie als der ohne TPM. Dennoch handelt es sich um eine rationale Entscheidung den TPM-Chip anzubinden.

Der erste Grund dafür ist, dass die Verwendung des TPM-Chips gegenüber der reinen Softwarelösung sogar noch Energie einspart. Tabelle 2 stellt diese Ersparnis sehr eindrücklich dar: 7.030,0μJ wird bei der reinen Softwarelösung für die Verschlüsselung eines Bits benötigt, während bei der Verwendung des TPM-Chips dazu nur 5,4μJ nötig sind. Sofern also die Verwendung von asymmetrischer Kryptographie gewünscht ist, ist die von secFleck verwendete Lösung der Softwarelösung vorzuziehen.

Zudem muss bedacht werden, dass die secFleck API des TPM-Chips die Möglichkeit bietet den TPM-Chip explizit ein- bzw. auszuschalten (vgl. Abb. 4). Dies ist deshalb besonders interessant, da der TPM-Chip bei der Verwendung Hybrider Verschlüsselungsmethoden nur während der initialen Schlüsselaustauschphase benötigt wird. Diese ist im Verhältnis zur gesamten Betriebszeit des Moduls nur relativ kurz, weshalb davon ausgegangen werden kann, dass der durch den TPM-Chip hervorgerufene Mehrverbrauch an Energie die Möglichkeiten des Sensorknotens nicht übersteigt.

4.1.4 Kosten

Die Kosten des von den Autoren in [12] verwendeten TPM-Chips „Atmel AT97SC3203S“ lagen den Angaben der Autoren nach bei $4.5, und machten somit weniger als 5% der Gesamtkosten eines gängigen Sensorknotens aus [12].

Diese Angabe ist nur schwer zu überprüfen. Zwar liegen die Preise gängiger Sensorknoten nach wie vor um die $100, es war zum Zeitpunkt der Anfertigung dieser Arbeit jedoch nicht möglich aktuelle Preisinformationen für den in [12] verwendeten TPM-Chip zu erhalten. Es konnte jedoch der Preis des in seinen Leistungsdaten vergleichbaren „Infinion SLB9635TT 1.2“ TPM-Chips zu 2€ - also in etwa $2,6 - bestimmt werden [9]. Der Einbau von TPM-Chips in Sensorknoten bewegt sich also damals wie heute im vertretbaren Kostenrahmen.

4.2 secFleck Features

SecFleck bietet dem Anwender als Gesamtsystem eine Reihe von Features, die im Folgenden kurz beschrieben werden.

Die secFleck-API erlaubt es, kryptographische Schlüssel nicht nur im EEPROM oder dem RAM des Knotens zu speichern, sondern auch im EEPROM des TPMs. Möglich wird dies dadurch, dass der von den secFleck-Autoren verwendete TPM-Chip Atmel AT 97SC3203S über den TPM v1.2 Standard hinaus über einen internen, sicheren Speicher verfügt. Das Feature der Schlüsselspeicherplatzwahl ist ein außerordentlich wichtiges: Einerseits wurde in [11] gezeigt, wie einfach es ist, den RAM und den EEPROM eines Knotens auszulesen, wodurch die Möglichkeit des Speicherns im TPM EEPROM besonders wichtig wird. Andererseits erfordert das Speichern der Schlüssel im TPM-Chip einen höheren Energieverbrauch als das im Knoten selbst. Weniger kritische Schlüssel sollten deshalb auch im Knoten selbst gespeichert werden können [12].

Als zweites wichtiges Feature preisen die Autoren in [12] secFlecks sichere Sitzungsschlüssel-Verwaltung an. Nicht nur, dass es die secFleck-API ermöglicht komfortabel mit Hilfe asymmetrischer Verfahren symmetrische Schlüssel für die direkte Kommunikation zwischen Basisstation und den Knoten zu verteilen, auch die Etablierung von Gruppenkommunikationsschlüssel wird unterstützt. Auf diese Weise ist es z.B. möglich, allen Knoten des WSNs denselben Schlüssel mitzuteilen, und so das gesamte Netzwerk vertraulich kommunizieren zu lassen [12].

Ein weiteres Feature das secFleck seinen Anwendern zur Verfügung stellt, ist die sichere Reprogrammierung von Sensorknoten. Dieses Feature ist deshalb so wichtig, weil es zwar bereits eine Reihe von „Multihop Over the Air Programming“ (MOAP) Protokollen gibt, der Sicherheitsaspekt bei diesen bisher jedoch vernachlässigt wurde. Ein Beispiel für ein solches System wäre Deluge [14]. Soll ein WSN mithilfe von Deluge umprogrammiert werden, so wird zunächst ein neues Programm-Image erstellt und dieses in kleine Chuncks aufgeteilt. Diese werden dann solange von Sensorknoten zu Sensorknoten verteilt, bis alle Knoten die neueste Programmversion besitzen. SecFleck erweitert dieses System um zwei Sicherheitsmechanismen. Einerseits werden die Chunks nach ihrer Erstellung durch den Programmierer unterschrieben, wodurch es den Knoten möglich wird die Integrität und die Quelle des neuen Programms zu prüfen. Andererseits bietet secFleck wie oben beschrieben die Möglichkeit einen globalen Gruppenschlüssel einzurichten und so die Vertraulichkeit der übertragenen Daten zu gewährleisten [12].

Das letzte von den Autoren in [12] angepriesene secFleck-Feature ist das des sicheren „Remote Procedure Calls“ (RPC). Bereits das normale FOS bietet Anwendungsprogrammen die Möglichkeit per RPC z.B. den Batteriezustand von Sensorknoten abfragen oder ihren RAM bzw. ihren EEPROM auslesen. Auch hier setzt secFleck wieder auf Sicherheit durch einen globalen symmetrischen Gruppenschlüssel. Dies stellt

sicherlich nicht die sicherste Variante dar, da lediglich die Vertraulichkeit - nicht jedoch die Verbindlichkeit oder die Integrität sichergestellt werden kann. Zudem führt die Kompromittierung des Schlüssels zu einem globalen Sicherheitsproblem. Der Vorteil eines einzigen globalen Schlüssels liegt jedoch auf der Hand: Eine einzelne RPC-Anfrage kann an viele Sensorknoten geschickt werden, ohne dass vorher bekannt sein muss, welcher Knoten wirklich die Anfrage bearbeiten wird.

5. VERWANDTE ARBEITEN

Neben dem in dieser Arbeit vorgestellten Ansatz die Kommunikation in drahtlosen Sensornetzwerken sicherer zu gestalten gibt es noch zahlreiche Weitere. Einige davon seien hier kurz vorgestellt:

In [15] wird mit TinySec eine Link-Layer-Sicherheitsarchitektur vorgestellt. Diese ist in das offizielle TinyOS-Release eingearbeitet worden und steht somit einer breiten Community zur Verfügung. Gegenüber dem von secFleck gewählten Weg der Ende-zu-Ende Verschlüsselung bietet der von TinySec gewählte Weg der Link-Layer-Verschlüsselung sowohl deutliche Vor- als auch Nachteile. Beispielsweise gehen die Autoren in [15] davon aus, dass Ereignisse meist von mehr als nur einem Sensorknoten aufgenommen werden. Im Falle einer Ende-zu-Ende Verschlüsselung müssen alle Nachrichten - auch wenn sie einen identischen Inhalt enthalten - dem Empfänger zugestellt werden. Unter Verwendung einer Link-Layer-Verschlüsselung können Duplikate frühzeitig erkannt und entfernt werden. Die aktuelle TinySec Implementierung lässt jedoch einige Fragestellungen wie die der effektiven Einbindung asymmetrische Verfahren unbeantwortet. Weitere Architekturen mit Link-Layer-Sicherheitsmechanismen wären beispielsweise GSM, Bluetooth oder 802.15.4 [15].

Auch Ansätze ein WSN auf der Network-Layer abzusichern existieren. So wird beispielsweise in [13] mit Ariadne eine Sicherheitserweiterung für das „Dynamic Source Routing“ (DSR) Routing-Protokoll vorgestellt, welches unter anderem eine Reihe von Denial-of-Service-Angriffen auf das WSN ausschließt.

Handelt es sich bei den Sensorknoten des Netzwerkes statt um eingebettete um vollwertige Rechensysteme, so können auch die aus dem Internet bekannten Sicherheitsmechanismen wie IPSec [7] oder TLS [6] eingesetzt werden. So wird in [16] beispielsweise IPSec eingesetzt, um eine Sicherheitsarchitektur für globale Raketenabwehrsysteme zu entwickeln.

Eine Liste aktueller Publikationen zum Thema Sicherheit in drahtlosen Sensornetzwerken kann in [3] gefunden werden.

6. ZUSAMMENFASSUNG UND AUSBLICK

Zusammenfassend lässt sich sagen, dass der von secFleck gegangene Weg der Hardwareanbindung eines TPM-Chips zur Verfügbarmachung von asymmetrischen Verschlüsselungsverfahren in drahtlosen Sensornetzwerken Erfolg verspricht. Nicht nur ermöglicht der TPM-Chip eine vollwertige, unbeschnittene RSA-Verschlüsselung, er führt diese zudem auch nahezu „in Echtzeit“ durch. Der Preis, der für diese Funktionalität gezahlt werden muss ist der deutlich höhere Stromverbrauch gegenüber einem Fleck^{TM}-System ohne TPM-Chip. Da jedoch der TPM-Chip abschaltbar ist und bei Verwendung eines hybriden Verschlüsselungsverfahrens zudem nur sporadisch zum Einsatz kommt, stellt dies einen akzeptablen Preis dar.

Kritisch sei an dieser Stelle angemerkt, dass der TPM-Chip über wesentlich mehr Fähigkeiten verfügen würde, als tatsächlich genutzt werden. So ist gerade das Feature des beglaubigten Bootvorgangs in einer typischen WSN-Umgebung, in der keineswegs davon ausgegangen werden kann, dass jeder Knoten unmanipuliert ist, von extrem hohem Wert. Auch für die Features der überprüfbaren anonymen Identitäten und des sicheren, an den Zustand des Systems gebundenen Speicherplatzes sind Szenarien vorstellbar, in denen sie einen erheblichen Nutzen darstellen würden.

Für die Zukunft wäre also durchaus ein secFleck-v2 mit einer besseren Ausnutzung des TPM-Chips vorstellbar. Auf diese Weise würde secFlecks vielversprechendes Potential noch besser ausgenutzt werden.

7. LITERATUR

[1] Australian commonwealth scientific and research organization (csiro). `http://www.csiro.au`.

[2] Datacall telemetry. `http://www.datacall.net.au`.

[3] H. Alzaid. Wireless sensor networks security, 2010 5. `http://www.wsn-security.info/Security_Lounge.htm`.

[4] B. Carter, A. Kassin, and T. Magoc. Advanced Encryption Standard. 2007.

[5] P. Corke and P. Sikka. Fos-a new operating system for sensor networks. In *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN08)*, Brisbane, Australia, 2008. CSIRO ICT Centre.

[6] T. Dierks and C. Allen. The TLS protocol version 1.0, 1999. RFC 2246, January 1999.

[7] N. Doraswamy and D. Harkins. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall, 2003.

[8] D. Eastlake and P. Jones. US secure hash algorithm 1 (SHA1), 2002. RFC 3174, September 2001.

[9] EBV Elektronik GmbH & Co. KG. `http://www.ebv.com`.

[10] C. Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Oldenbourg, 5 edition, 2008.

[11] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. *Department of Computer Science University of Colorado at Boulder*, 2005.

[12] W. Hu, P. Corke, W. C. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. In *Wireless Sensor Networks*, volume 5432 of *Lecture Notes in Computer Science*, pages 296–311. Springer Berlin / Heidelberg, 2009.

[13] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1):21–38, 2005.

[14] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA,

2004. ACM.

[15] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM.

[16] G. Katsis and NAVAL POSTGRADUATE SCHOOL MONTEREY CA. Multistage Security Mechanism For Hybrid, Large-Scale Wireless Sensor Networks. 2007.

[17] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. In *PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pages 446–465, London, UK, 2000. Springer-Verlag.

[18] M. Li and Y. Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(2):1–29, 2009.

[19] C. Mitchell. Trusted computing, August 2006. `http://www.isg.rhul.ac.uk/cjm/060823.zip`.

[20] R. Needham and D. Wheeler. eXtended Tiny Encryption Algorithm, October 1997.

[21] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[22] P. Sikka, P. Corke, and L. Overs. Wireless sensor devices for animal tracking and control. *Local Computer Networks, Annual IEEE Conference on*, 0:446–454, 2004.

[23] D. Standard. Federal Information Processing Standards Publication 46. *National Bureau of Standards, US Department of Commerce*, 1977.

[24] N. Sumrall and M. Novoa. Trusted Computing Group (TCG) and the TPM 1.2 Specification. In *Intel Developer Forum*, volume 32, March 2007.

[25] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, 2004.

[26] The Powercom Group. Fleck long range wireless sensing and control, November 2008. `http://www.powercomgroup.com/Latest_News_Stories/Fleck_long_range_wireless_sensing_and_control.shtml`.

[27] TIK WSN Research Group. The sensor network museum™, 2010. `http://www.snm.ethz.ch/Projects/Fleck`.