SEMINAR SENSORNETZE
SICHERHEIT IN SENSORNETZEN
(JUNI 2003)

Danat Pomeranets
danat@student.ethz.ch

Distributed Systems
Computer Science
ETH Zürich

ÜBERSICHT

Das Thema Sicherheit ist heutzutage in aller Munde. Auch im Bereich der Sensornetze spielt es eine wichtige Rolle, sowohl für militärische Anwendungen, als auch für viele kritische zivile Beispiele. In Anbetracht der sehr limitierten Ressourcen der Knoten eines Sensornetzes scheint die Aufgabe, entsprechende Sicherheitsanforderungen zu implementieren, schwierig bis unlösbar. Dass das Thema dennoch nicht hoffnungslos ist, möchte ich im Folgenden zeigen.
Ich werde zunächst den Begriff Sicherheit in Bezug zu Sensornetzen setzen und kurz auf einige grundlegende Aspekte eingehen. Daraufhin stelle ich 2 unabhängige Arbeiten vor.
Die erste beschäftigt sich mit Denial-Of-Service-Attacken auf den unteren Schichten des OSI-Modells, sowie deren Lösungen [1]. Ich werde einige Beispiele daraus herausgreifen und näher erläutern.
Die zweite Arbeit beschreibt ein Protokoll, welches, trotz vieler Einschränkungen, effiziente Verschlüsselung und Authentifizierung in Sensornetzen erlaubt, und unter bestimmten Annahmen auch authentischen Broadcast realisiert [2].
Zum Schluss stelle ich die erreichten Ergebnisse dar, und gehe auf weitere Forschungsthemen ein.

1. MOTIVATION

Zur Motivation des Thema Sicherheit in Sensornetzen möchte ich zunächst einige Anwendungsbeispiele für Sensornetze aufzeigen, wo bestimmte Sicherheitsaspekte Sinn machen und zum Teil absolut notwendig sind.
Als erstes muss wohl das militärische Einsatzgebiet erwähnt werden, wo Sensornetze z.B. zur Schlachtfeldüberwachung benutzt werden können. Dass es bei diesem Beispiel dem Gegner nicht möglich sein soll, Manipulationen jeglicher Art an dem Sensornetz vorzunehmen, stellt eine sinnvolle Anforderung dar. Es müssen also Sicherheitsvorkehrungen getroffen werden, um einen Schutz gegen Manipulationen zu gewährleisten. Die verschiedenen Angriffsmöglichkeiten und Schutzmechanismen sind das Thema dieser Ausarbeitung.
Sicherheit spielt aber nicht nur bei militärischen Anwendungen eine wichtige Rolle. Sensornetze werden vermehrt zur Überwachung wichtiger Parameter in der Umgebung eingesetzt, z.B. als Feuermelder in Gebäuden oder zur Überwachung von Nuklearreaktoren. Eine mögliche Art der Manipulation eines solchen Sensornetzes wäre z.B. die Auslösung eines Fehlalarms, was für die Betreiber einen hohen Aufwand und zusätzliche Kosten bedeuten würde.
Ein weiterer Bereich, wo bestimmte Sicherheitsmechanismen gebraucht werden, ist der Schutz der Privatsphäre. Da Sensornetze meist drahtlose Kommunikation benutzen, hat jeder Neugieriger bei genügend kleiner Distanz eine Möglichkeit, die Kommunikation abzuhören. Dies stellt z.B. im Bereich der Home Healthcare ein Problem dar. Sensornetze werden hier für die Überwachung der wichtigen Gesundheitsparameter von Patienten eingesetzt. Gesammelte Informationen werden dabei periodisch an den zuständigen Arzt übermittelt. Es ist aber nicht im Interesse jedes Patienten, dass seine gesundheitlichen Daten wie z.B. Puls oder Blutdruck für jedermann in seiner Umgebung zugänglich sind.
Jedes dieser Einsatzbeispiele für Sensornetze benötigt eine andere Art der Sicherheit, da die Voraussetzung jedes mal andere sind.

2. SICHERHEITSANFORDERUNGEN

Dieses Kapitel gibt eine Übersicht über die für Sensornetze relevanten Sicherheitsanforderungen. Es werden ausserdem mögliche Angriffe diskutiert, die durch die Implementierung der jeweiligen Anforderungen verhindert werden können.

2.1. Vertraulichkeit (*confidentiality*)

Vertraulichkeit bedeutet, dass der Inhalt der Nachrichten für Gegner nicht lesbar ist. Vertraulichkeit wird oft durch Verschlüsselung der Nachrichten realisiert, und macht das Abhören der Nachrichten (*eavesdropping*) unmöglich.

2.2. Integrität (*integrity*)

Wenn die Integrität der Nachrichten sichergestellt wird, können beide Kommunikationspartner sicher sein, dass die Nachrichten nicht nachträglich verändert werden können, ohne dass sie es bemerken. So werden Manipulationen von Nachrichten durch den Gegner verhindert.

2.3. Authentizität (*authencity*)

Ist die Authentizität gewährleistet, dann weiss jeder der Kommunikationspartner sicher, dass er mit dem „echten" Partner kommuniziert. Authentizität schützt gegen Angriffe durch Einschleuschen von falschen Nachrichten.

2.4. Aktualität (*freshness*)

Aktualität ist besonders bei den Sensornetzen wichtig, die zeitrelevante Informationen sammeln. Aktualität der Nachrichten bedeutet, dass alte Nachrichten als solche immer erkannt werden. Implementierung der Aktualitätsanforderung ist also der Schutz gegen *Replay*-Attacken.

2.5. Verfügbarkeit (*availability*)

Ein möglicher Angriff auf ein Sensornetz könnte darin bestehen, dessen Verfügbarkeit zu stören (*Denial of Service*). Mit verschiedenen Angriffsmöglichkeiten und den möglichen Schutzmechanismen beschäftigt sich das Kapitel 3.

3. DENIAL OF SERVICE

Was bedeutet Denial of Service (im Folgenden mit DoS abgekürzt)? In [1] wird DoS als „Jedes Ereignis, welches die Funktionalität des Netzwerkes erheblich beeinträchtigt" definiert. Dieses Ereignis kann dabei entweder gezielt hervorgerufen worden sein – dann spricht man von einem Angriff. Es kann aber auch durch Hardware-Ausfälle, Software-Bugs oder Einflüsse aus der Umgebung bedingt sein.
In diesem Kapitel betrachten wir die absichtlichen Angriffe auf Sensornetze, mit dem Ziel, die Verfügbarkeit des Netzes zu stören. Solche Angriffe sind auf mehreren Ebenen des OSI-Modells möglich.

3.1. Physische Schicht (*physical layer*)

Sensornetze bestehen meistens aus kleinen Knoten, die in der Zielumgebung untergebracht werden. Daraus ergibt sich die physische Angreifbarkeit der Knoten, die leicht gefunden und manipuliert werden können (*tampering*). Will man das verhindern, müssen Sensorknoten entsprechend hergestellt werden. Ein robustes Gehäuse könnte den physischen Eingriff erschweren, versteckte Knoten – das Auffinden. Beide Lösungen bieten jedoch keinen 100%-igen Schutz, was weitere Massnahmen notwendig macht. Eine weitere Massnahme könnte darin bestehen, innerhalb eines Sensorknotens Sensoren anzubringen, die Manipulationen an der eigenen Hardware feststellen können. Daraufhin würde der Knoten seinen sämtlichen Speicher löschen, so dass der Gegner keine sicherheitsrelevanten Informationen wie ggf. kryptographische Schlüssel erhält.
In Sensornetzen wird meistens drahtlose Radiokommunikation eingesetzt, was einen weiteren, naheliegenden Angriff auf der physischen Ebene ermöglicht – die Kommunikation durch Interferenzen mit einem Störsignal zu blockieren oder ganz unmöglich zu machen (*jamming*). Dieser Angriff ist einfach und effektiv realisierbar, das Störsignal kann aber auch ohne „böse Absichten" auftreten. Ein Schutz gegen Jamming ist für Sensornetze also von entscheidender Bedeutung.
Wird nur eine Radiofrequenz gestört und stehen dem Sensornetz mehrere Radiofrequenzen zur Verfügung, so kann man sich für eine der folgenden Varianten entscheiden:

- Breitspektrum-Kommunikation – dabei wird die gestörte Frequenz ignoriert und die Kommunikation auf den noch verfügbaren Frequenzen weitergeführt
- Frequenz-Hopping – wenn die Frequenzen mehrmals pro Sekunde gewechselt werden, dann ist der Verlust durch eine gestörte Frequenz verkraftbar
- Code-Spreading – durch den Einsatz der orthogonalen Kodierung wird das Störsignal bei Dekodierung herausgefiltert

Diese Lösungen setzen, wie bereits erwähnt, eine Mehrfrequenzkommunikation voraus, was nicht für alle Sensornetze zutrifft (u.a. werden dadurch Sensorknoten komplexer und teuerer). Steht nur eine Radiofrequenz für Kommunikation zur Verfügung, so sind die Verteidigungs-möglichkeiten beschränkt. Wenn das gesamte Sensornetz gestört wird, können die Sensorknoten sich in den Schlaf-Modus begeben, um die eigene Energie zu sparen, in der Hoffnung, den Angriff überleben zu können.
Wenn das Sensornetz aber weiträumig ausgelegt ist, ist das Stören des gesamten Netzes eine aufwendige Aufgabe. In diesem Fall kann es vorkommen, dass nur ein Teilgebiet gestört wird. Damit die Störung eines Teilgebiets nicht trotzdem zum Ausfall des gesamten Netzes führt, müssen Sensorknoten gemeinsam das gestörte Teilgebiet erkennen und von der Kommunikation ausschliessen. Dies geschieht in 2 Phasen:

1. Phase: Erkennung
   Die Sensorknoten innerhalb des Jamming-Gebietes merken dass sie angegriffen werden. Sie können zwar keine Nachrichten mehr empfangen, können aber eigene Nachrichten verschicken. Diejenigen von ihnen, die am Rand liegen, können evtl. ihre nächsten Nachbarn, die sich ausserhalb des gestörten Bereiches befinden und daher ihre Nachrichten auch empfangen können, erreichen und alarmieren. Abbildung 1 stellt Phase 1 graphisch dar.

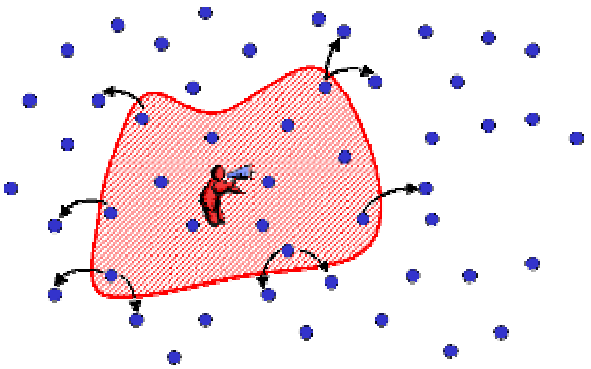

Abbildung 1: Sensorknoten melden Störung an die Nachbarn

2. Phase: Rerouting
   Die alarmierten Knoten müssen nun zusammenarbeiten, um eine Art Schutzmauer um das gestörte Teilgebiet zu

errichten. Schutzmauer bedeutet, dass keine Nachrichten mehr durch dieses Gebiet geroutet werden. Die gesamte Kommunikation wird um den betroffenen Bereich herum geleitet. Dadurch wird sichergestellt, dass keine Nachrichten durch das Jamming verlorengehen. Die Sensorknoten innerhalb des Störgebietes können jedoch weiterhin keine Nachrichten empfangen.

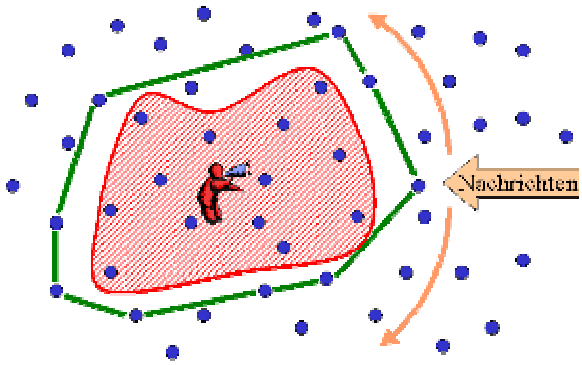

Abbildung 2: Schutzmauer, Nachrichten werden umgeleitet

3.2. Sicherungsschicht (link layer)

Die Aufgabe der Sicherungsschicht ist bekanntlich, den Zugang zu einem gemeinsamen Medium zu gewährleisten, und die dabei entstehenden Kollisionen zu detektieren und zu behandeln. Ein Gegner könnte dementsprechend versuchen, die Verfügbarkeit dieser Schicht durch absichtliche Kollisionen zu stören.

Das Problem hierbei ist, dass der Gegner viel energie-effizienter als die einzelnen Sensorknoten ist. Das bedeutet, dass der Gegner keine eigene Hardware benötigt, um ein Sensornetz zu stören. Es würde ihm genügen, einige wenige vorhandene Sensorknoten zu manipulieren, um einen DoS-Angriff auf das gesamte Sensornetz zu fahren. Anders als im Kapitel 3.1 besprochen, könnte der Gegner diesmal nicht die ganze Zeit ein Störsignal senden, sondern nur kurze Spitzen erzeugen, die bestimmte Bits der Datenpakete durch Überlagerung zerstören würden. Die ehrlichen Knoten müssten daraufhin das ganze Paket nochmals versenden. Würde der Gegner ausserdem bestimmte Pakete angreifen, wie z.B. das ACK-Paket bei einigen Sliding-Window - Protokollen, so würde es zum Zusammenbruch des Sliding-Window führen und somit zu einer ineffizienten Flusssteuerung.

Gegen solche Art von Angriffen existiert kein vollkommener Schutz. Man könnte es dem Gegner allerdings erschweren, einen solchen Angriff zu fahren, indem man z.B. fehlerkorrigierende Codes für die Kommunikation einsetzt, was den Knoten erlauben würde, einige wenige Bitfehler in den Paketen eigenständig zu reparieren, ohne das Paket nochmals versenden zu müssen. Eine weitere Möglichkeit ist der Einsatz eines probabilistischen Verfahrens zur Auflösung von Kollisionen, so wie es z.B. bei herkömmlichen Ethernet-Netzwerken der Fall ist (*random backoff*). Dadurch wird die Wahrscheinlichkeit reduziert, dass der Gegner ein bestimmtes Paket zerstören kann, da er nicht vorhersagen kann, wann genau dieses Paket übermittelt wird.

3.3. Routing

Auf der Routing-Ebene werden Pakete vom Sender zum Empfänger über mehrere Zwischenknoten übertragen. Gegenüber herkömmlichen Systemen (z.B. Internet) sind Sensornetze viel anfälliger für DoS-Angriffe auf dieser Ebene. Der Grund dafür liegt darin, dass Sensornetze über keine vordefinierte Infrastruktur verfügen, sondern sich spontan vernetzen können müssen, und daher jeder Sensorknoten gleichzeitig als Router fungieren muss. Jeder manipulierter Sensorknoten ist somit ein manipulierter Router.

Wenn es dem Gegner also gelingt, einen oder mehrere Knoten unter seine Kontrolle zu bringen, oder eigene Knoten im Sensornetz zu integrieren, dann stehen ihm mehrere Angriffsmöglichkeiten zu Verfügung. Einige davon stelle ich im Folgenden vor:

- Ein manipulierter Router (d.h. Sensorknoten) könnte einige der Nachrichten, die über ihn geroutet werden, verwerfen, oder zumindest eigene Nachrichten mit einer viel höheren Priorität verarbeiten. Ein möglicher Schutz gegen solche Art von Fehlverhalten ist die Einführung der Redundanz. Die Sensorknoten müssen ihre Nachrichten entweder gleichzeitig auf mehreren möglichen Wegen zum Ziel schicken, oder zumindest die selbe Nachricht mehrmals hintereinander verschicken.
- Eine andere Art von Fehlverhalten ist das sogenannte Wurmloch. Dabei verschicken ein oder mehrere fehlerhafte Router die Pakete nicht direkt in die Richtung des korrekten Empfängers weiter, sondern lassen sie zunächst im Netz zirkulieren – das geht um so besser, je mehr fehlerhafte Router vorhanden sind. Dadurch wird in Sensornetzen viel Energie unnötigerweise verbraucht und die Latenzzeit der Nachrichten steigt. Dieser Angriff auf ein Sensornetz ist schwierig zu detektieren, da die Nachrichten schlussendlich trotzdem ihr Ziel erreichen, und somit niemand alarmiert wird. Bei drahtloser Kommunikation können Sensorknoten allerdings ihre Nachbarn überwachen, da sie ja alles empfangen können, was ihre Nachbarn verschicken. So können sie feststellen, dass eine Nachricht in die falsche Richtung weitergeleitet wurde.
- Bei einigen Routing-Protokollen besteht die Gefahr eines Schwarzes-Loch - Angriffs. Bei diesem Angriff versucht ein Router alle Routing-Pfade zu sich zu holen, in dem er falsche Gewichte der Pfade an andere Router im Netz verschickt, woraufhin sie ihre Pakete immer an ihn weiterleiten. Dagegen kann man sich nur dann schützen, wenn Router (d.h. Sensorknoten) sich untereinander authentisieren müssen. Eine mögliche Lösung für Authentisierung in Sensornetzen ist das SNEP - Protokoll [2], das ich im Kapitel 4.2 vorstelle.

3.4. Zusammenfassung

DoS-Angriffe sind in beliebigen Netzen oft schwer abzuwehren und manchmal sogar schwer festzustellen. In Sensornetzen wird die Problematik dadurch verschärft, dass jeder Sensorknoten ebenfalls ein Router ist. Die Sicherheit muss also bereits zur Designzeit und auf allen Ebenen berücksichtigt werden, da die benötigten Sicherheitsanforderungen z.B. einen erheblichen Einfluss auf die Wahl der Protokolle haben. Man

wird zwar nicht alle Angriffsmöglichkeiten ausschliessen können, aber möglicherweise es dem Gegner bedeutend erschweren, einen DoS-Angriff erfolgreich durchführen zu können. Der Schutz besteht also darin, durch eigenen Aufwand den Aufwand für den Gegner überproportional zu steigern – ein Prinzip der Aufwandsverschiebung.

4. SPINS: SECURITY PROTOCOLS FOR SENSOR NETWORKS

In diesem Kapitel stelle ich 2 Protokolle für die Anwendungsebene des OSI-Modells vor: SNEP und µTESLA [2]. SNEP gewährleistet Vertraulichkeit, Authentizität und Aktualität zwischen den Kommunikationspartner, µTESLA realisiert authentischen Broadcast.
Beide Protokolle wurden für die SmartDust-Sensorknoten entwickelt [3], welche über folgende Hardware verfügen:

Characteristics of prototype SmartDust nodes.

CPU	8-bit, 4 MHz
Storage	8 Kbytes instruction flash
	512 bytes RAM
	512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 Kbps
Operating system	TinyOS
OS code space	3500 bytes
Available code space	4500 bytes

Abbildung 3: Hardware von SmartDust-Sensorknoten

Wie man in Abbildung 3 sieht verfügen SmartDust-Sensorknoten über sehr eingeschränkte Ressourcen, verglichen z.B. mit einem herkömmlichen PC. In Anbetracht dieser Einschränkungen können die Verfahren, die im Internet für Verschlüsseln oder Signieren eingesetzt werden, nicht auf Sensornetze portiert werden. So braucht z.B. das RSA-1024-Protokoll entsprechend 128 Byte pro Schlüssel oder pro Signatur. Wenn allerdings insgesamt nur 512 Byte an Datenspeicher zur Verfügung stehen, so wie es bei SmartDust der Fall ist, ist RSA nicht einsetzbar. Da ausserdem bei Sensornetzen eher kurze Nachrichten verschickt werden (der Durchschnitt liegt bei SmartDust bei ca. 30 Byte), sind Signaturen von 128 Byte ein enormer Overhead.
Allgemein benötigen Public-Key-Kryptographie-Verfahren um Grössenordnungen grössere Schlüssel und um Grössenordnungen aufwendigere Berechnungen, verglichen mit den symmetrischen kryptographischen Verfahren. Vor allem wegen der limitierten Energieressourcen der Sensornetze ist der Einsatz von Public-Key-Kryptographie zur Sicherung von Sensornetzen nicht praktikabel.

4.1. Ziele, Netzwerkmodell, Vertrauensmodell

Das Ziel von SPINS ist also, Vertraulichkeit, Authentizität und Aktualität der Nachrichten sowie authentischen Broadcast mit Hilfe von nur symmetrischer Kryptographie zu realisieren. Ausserdem muss dabei der Kommunikations-Overhead minimiert werden, da drahtlose Kommunikation den grössten Energieverbrauch der Sensorknoten verursacht.
Wenn von Sicherheitsanforderungen die Rede ist, müssen auch die Annahmen definiert werden, unter denen diese Anforderungen erreicht werden können.
Bei SPINS wird ein selbstorganisiertes, Mutli-Hop-fähiges Sensornetzwerk vorausgesetzt, welches zudem über synchrone Phasen verfügt. Weiterhin ist in dem Sensornetz eine Basisstation integriert, die sowohl über leistungsstärkere Hardware als auch über einen unbegrenzten Energievorrat verfügt. Sichere Kommunikation wird nur für die Kommunikation zwischen Knoten und der Basisstation gebraucht, wobei es 3 Arten von Kommunikationsmöglichkeiten gibt:

- Basisstation fragt einen Knoten nach seinen Daten
- Knoten schickt seine Daten der Basisstation
- Basisstation verschickt dieselbe Nachricht an alle Knoten per Broadcast (ebenfalls eine Anfrage, ein Software-Update, ...)

Als Kommunikationsprimitive dient den Sensorknoten dabei lokaler Broadcast, was durch drahtlose Kommunikation verursacht wird.
Eine weitere wichtige Voraussetzung ist das Vertrauen der Knoten in die Basisstation. Die Knoten halten nur andere Knoten für nicht vertrauenswürdig, was Sinn macht, wenn man bedenkt, dass einzelne Knoten in der Umgebung verteilt werden und daher leichte manipulierbar sind. Wegen drahtloser Kommunikation kann sich ausserdem jeder für einen Sensorknoten ausgeben. Das Vertrauen in die Basisstation ist hingegen eine grosse Einschränkung, es ist aber sowohl für SNEP als auch für µTESLA ein wichtiger Bestandteil. Ausserdem vertrauen Knoten auch sich selbst, was zur Folge hat, dass die Knoten keine besonderen Mechanismen haben, um eigene Sensorwerte zu überprüfen oder sich vor internen Manipulationen zu schützen.
Die letzte Voraussetzung, die erfüllt werden muss, ist die Verfügbarkeit des Netzwerks – das bedeutet dass Nachrichten mit einer Wahrscheinlichkeit grösser 0 ausgeliefert werden.

4.2. SNEP – Secure Network Encryption Protocol

Dieses Protokoll garantiert, wie bereits erwähnt, die Vertraulichkeit, Authentizität und Aktualität der Nachrichten. Dies wird mit Hilfe von symmetrischen kryptographischen Verschlüsselungs- und Signier-Operationen erreicht. Da nur ca. 4,5 KB an Programmspeicher für alle Anwendungen zur Verfügung stehen, ist die Minimierung des Programmcodes eine wichtige Aufgabe. Daher wird bei SNEP eine kryptographische Funktion für Schlüsselgenerierung, Verschlüsselung, Signaturberechnung und als Pseudozufallsgenerator eingesetzt. Die Autoren haben sich für die RC5-Funktion von Ronald Rivest entschieden [4], da sie über einige wichtige Vorteile verfügt.
Zum einen gilt RC5 ab einer Schlüssellänge von 64bit bisher als ungebrochen. Zum anderen läßt sich der RC5-Algorithmus sehr kompakt beschreiben, was für die Programmgröße eine wichtige Rolle spielt. Zwei weitere Vorteile sind die Flexibilität bei der Wahl der Parameter wie Wortgrösse, Rundenanzahl und Schlüssellänge, und die wichtige Eigenschaft, dass der verschlüsselte Text (*Chiffrat*) gleich lang

wie der ursprüngliche Klartext ist. Dadurch kann Overhead beim Verschicken der Nachrichten reduziert werden.
Im Weiteren sei die RC5-Funktion durch eine Blackbox repräsentiert, die 2 Inputs für den Schlüssel und den Klartext und einen Output für das Chiffrat hat, wie es in Abbildung 4 dargestellt ist.

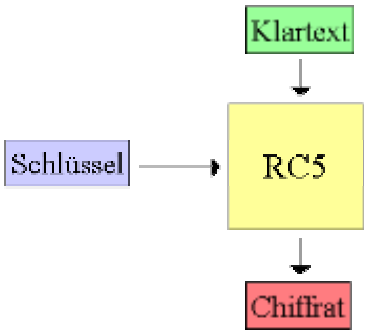

Abbildung 4: Die RC5-Funktion als Blackbox

4.2.1. Schlüsselgenerierung

Wie bereits gesehen wird RC5 für mehrere Operationen eingesetzt. Dabei dürfen für mehrere verschiedene Operationen nicht dieselben Schlüssel benutzt werden, da es andernfalls möglich sein könnte, durch bestimmte Korrelationen zwischen den beiden Operationen Informationen über den Schlüssel zu gewinnen. Daher benötigen beide Kommunikationspartner mehrere Schlüssel, die aufgrund von symmetrischer Kryptographie bei beiden Partner gleich sein müssen.
Bei SNEP verfügen alle Knoten über einen gemeinsamen Schlüssel mit der Basisstation, den sogenannten Master-Key. Dieser Master-Key ist pro Knoten unterschiedlich, die Basisstation muss also für jeden Knoten den entsprechenden Schlüssel speichern. Alle weiteren Schlüssel werden aus dem Master-Key mit Hilfe von RC5 folgendermassen generiert:

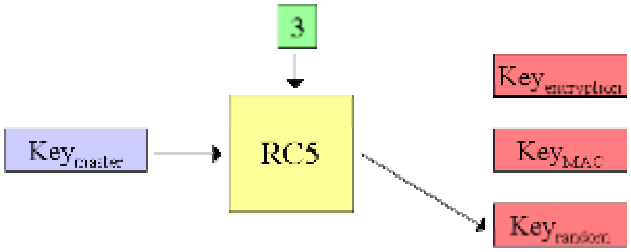

Abbildung 5: Schlüsselgenerierung mit RC5

Der Master-Key (Key_{master}) wird als Schlüsselinput eingesetzt, die Rolle von Klartext übernimmt ein Zähler, der bei 1 anfängt und bei jedem weiteren Schlüssel um 1 inkrementiert wird. Da sowohl die Basisstation als auch der jeweilige Sensorknoten über den selben Master-Key verfügen, und den selben Zählerwert für den jeweiligen Schlüssel benutzen, sind die generierten Schlüssel für beide ebenfalls identisch. Insgesamt benötigt SNEP 3 unabhängige Schlüssel – Verschlüsselungsschlüssel ($Key_{encryption}$), Signierschlüssel (Key_{MAC}), und ein Initialwert für den Pseudozufallsgenerator (Key_{random}).

4.2.2. Verschlüsselung

Verschlüsselung wird bei SNEP gebraucht, um die Vertraulichkeit der Nachrichten zu gewährleisten. Zur Verschlüsselung wird RC5 im Stream-Cipher-Modus eingesetzt. Der Output der RC5-Funktion wird also mit dem Klartext bitweise addiert, was dem One-Time-Pad-Prinzip entspricht. Im Gegensatz zu One-Time-Pad ist dieses Verfahren allerdings nur kryptographisch sicher, da der Output von RC5 nur pseudozufällig ist. Abbildung 6 zeigt den detaillierten Verschlüsselungsvorgang. **P** ist der gesamte Klartext, $\mathbf{P_2}$ und $\mathbf{P_3}$ sind Klartextblöcke. $\mathbf{C_1}$ und $\mathbf{C_2}$ sind entsprechende Chiffratblöcke, das gesamte Chiffrat wird folgendermassen notiert: **{P}<Kencr, Zähler>**

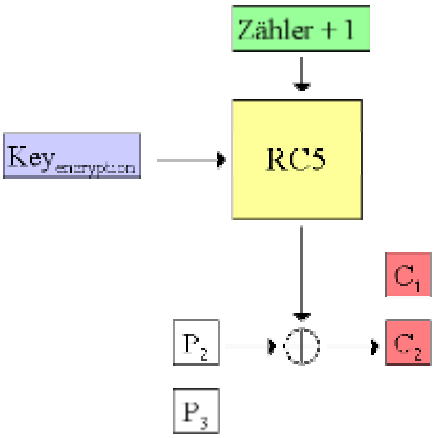

Abbildung 6: Verschlüsselung mit RC5

Wie man es der Abbildung entnehmen kann, wird der Klartext nicht an RC5 übergeben. Statt dessen benutzen beide Kommunikationspartner einen Zähler, der bei jeder Nachricht inkrementiert wird. Dadurch wird erreicht, dass einerseits jede Nachricht ein anderes Chiffrat zur Folge hat, selbst wenn der Inhalt gleich ist, und andererseits die Entschlüsselungs-operation mit der Verschlüsselungsoperation identisch ist. Dafür müssen allerdings beide Kommunikationspartner den selben Zählerwert haben, d.h. die Zähler müssen synchron sein. Die Synchronisation der Zähler kann nur dann gefährdet werden, wenn eine oder mehrere Nachrichten zwischen Sender und Empfänger verlorengehen. Gehen nur einige wenige Nachrichten verloren, so können die beiden Kommunikationspartner die Zähler wieder synchronisieren, indem sie ein paar nächst grössere Zählerwerte ausprobieren. Wenn aber mehrere Nachrichten verloren gegangen sind, oder auch in der Setup-Phase des Protokolls, müssen die Zähler explizit re-synchronisiert werden. Dafür wird ein anderes Protokoll gebraucht, welches nur Signaturen und keine Verschlüsselung benötigt.

4.2.3. Signatur

Signaturen dienen dazu, die Integrität und gleichzeitig auch die Authentizität der Nachrichten zu garantieren. Um Signaturen der Nachrichten zu berechnen, wird RC5 im Cipher-Block-Chaining-Modus eingesetzt. Dies ist ein gängiges Signierverfahren, welches bei vielen anderen Protokollen in dieser Form gebraucht wird. Abbildung 7 stellt den CBC-

Modus graphisch dar. $\mathbf{K_{MAC}}$ ist dabei der generierte Signierschlüssel, $\mathbf{C_1}$, $\mathbf{C_2}$ und $\mathbf{C_3}$ sind die Blöcke der verschlüsselten Nachricht **C**, die signiert werden muss, **MAC** – ist die Signatur (*message authentication code*), auch $\mathbf{MAC(K_{MAC},C)}$ geschrieben.

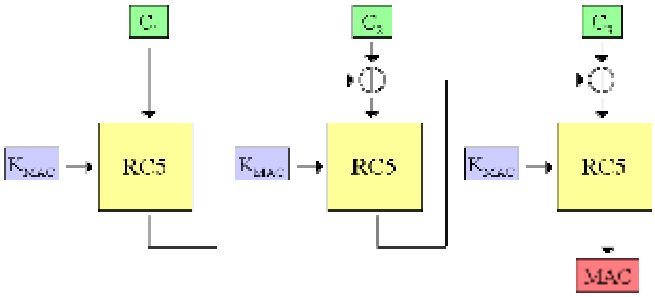

Abbildung 7: RC5 im CBC-Modus

Durch die Verkettung von RC5-Blocks wird sichergestellt, dass jedes Input-Bit sich im Output wiederspiegelt, was eine wichtige kryptographische Eigenschaft ist.

4.2.4. Beispiel

An dieser Stelle möchte ich ein Beispiel für die Kommunikation zwischen einem Sensorknoten und der Basisstation aufzeigen, und anhand dieses Beispiels einige weitere wichtige Eigenschaften von SNEP diskutieren.

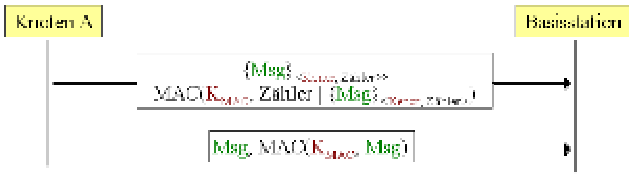

Abbildung 8: Beispiel für SNEP

Wie man in Abbildung 8 sieht, können Nachrichten bei SNEP entweder verschlüsselt und signiert, oder nur signiert verschickt werden, wenn die Vertraulichkeit nicht erforderlich ist. Wird die Nachricht verschlüsselt, so muss der Zählerwert, der für die Verschlüsselung benutzt wurde, in die Berechnung von der Signatur mit einfliessen. Dadurch wird die Aktualität der Nachrichten garantiert, da selbst Nachrichten mit dem selben Inhalt jedes mal eine andere Signatur bekommen. Damit kann jeder Sensorknoten feststellen, wenn der Gegner einen Replay-Angriff versucht, da die Signatur der wiederholten Nachrichten nicht mehr überprüft werden kann, weil der Zähler beim Empfänger nach jeder Nachricht inkrementiert wird.
Es ist wichtig zu bemerken, dass die Basisstation nicht nur gemeinsame Master-Schlüssel mit jedem Knoten hat, sondern auch für jeden Knoten einen eigenen Zähler speichern muss, der mit dem Zähler des Knotens synchron bleiben muss. Dadurch steigen die Anforderungen an die Basisstation, sowohl auf der Ebene der Ressourcen als auch auf der Ebene der physischen Sicherheit.

4.3. µTESLA – Micro Timed Efficient Stream Loss-tolerant Authentication

Da Sensornetze oft aus sehr vielen Knoten bestehen, ist das Versenden eine Nachricht an jeden einzelnen Knoten mit SNEP ziemlich aufwendig. Für den authentischen Broadcast von der Basisstation kann statt dessen µTESLA verwendet werden.
Authentischer Broadcast ist eine asymmetrische Operation, die man nicht mit nur symmetrischen Mitteln realisieren kann. Bei symmetrischer Kryptographie ist die Unterscheidung zwischen Sender und Empfänger per Definition nicht möglich, da beide über die selben Schlüssel verfügen. Genau dies wollen wir beim authentischen Broadcast aber erreichen. Die bekannten Lösungen mit Hilfe von asymmetrischer Kryptographie, d.h. mit Public-Key-Verfahren, sind für Sensornetze wegen der limitierten Ressourcen nicht realisierbar. µTESLA löst dieses Problem durch die Einführung der Asymmetrie durch verzögerte Schlüsselfreigabe.

4.3.1. Idee

Die Idee besteht darin, statt eines Schlüssels eine Kette von Schlüsseln zu benutzen. Diese Kette wird durch mehrfache Anwendung einer Einwegfunktion auf den Initialschlüssel generiert, wie man in Abbildung 9 sehen kann.

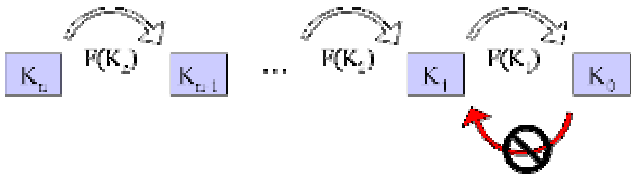

Abbildung 9: Schlüsselkette mit Einwegfunktion F

In Abbildung 9 ist $\mathbf{K_n}$ ist der Initialschlüssel, auf den die Funktion **F** mehrmals angewendet wird, $\mathbf{K_0}$ ist der Endwert. Einwegfunktion bedeutet in diesem Zusammenhang, dass jeder aus dem Schlüssel K_n den Schlüssel K_{n-1} berechnen kann, aber niemand zu einem gegebenen Schlüssel K_{n-1} einen Schlüssel K_n finden kann, so dass $F(K_n) = K_{n-1}$ ist. Diese Eigenschaft der Einwegfunktion erzeugt eine Asymmetrie, die wir für authentischen Broadcast brauchen.
Im µTESLA-Protokoll erzeugt die Basisstation eine solche Kette von Schlüsseln K_0 bis K_n (bzw. ist die Reihenfolge tatsächlich umgekehrt – siehe Abbildung 9) und verteilt den Schlüssel K_0 authentisch an alle Sensorknoten per Unicast – z.B. mit Hilfe von SNEP. Daraufhin wird die Zeit in Intervalle eingeteilt. Um dies zu ermöglichen müssen alle Sensorknoten und die Basisstation über synchrone Zeitphasen verfügen, was eine Anforderung an das Netzwerkmodell ist (s. Kapitel 4.1) Die Basisstation benutzt für jedes Zeitintervall *i* einen anderen Schlüssel K_i, um die Nachrichten zu signieren, wobei zum Signieren dasselbe Verfahren eingesetzt wird wie bei SNEP (s. Kapitel 4.2.3) Abbildung 10 dient der besseren Anschaulichkeit.

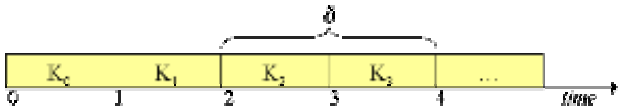

Abbildung 10: Verzögerte Schlüsselfreigabe bei μTESLA

Die signierten Nachrichten werden nach dem Signieren an die Sensorknoten per Broadcast verschickt, die aber zu dieser Zeit noch nicht die Authentizität der Nachrichten überprüfen können. Die Basisstation verschickt den Schlüssel K_i erst nach einer Verzögerung von δ Intervallen an alle Knoten, ebenfalls mittels Broadcast, der aber nicht authentisch sein muss. Die Knoten müssen die empfangenen Nachrichten also zunächst δ Intervalle lang zwischenspeichern, bevor sie deren Authentizität überprüfen können und sie weiter verarbeiten.

4.3.2. Authentizität

Bevor die Authentizität der Nachricht überprüft werden kann, muss die Authentizität des Schlüssels überprüft werden. Da jeder der Knoten über mindestens einen der früher benutzten Schlüssel verfügt, und zwar authentisch (dafür wird am Anfang der Schlüssel K_0 verschickt), kann er die Authentizität des letzten Schlüssels durch das Anwenden der Einwegfunktion auf diesen Schlüssel überprüfen (oder ggf. durch mehrfaches Anwenden, wenn Nachrichten verloren gehen). Die Einwegfunktion angewandt auf den letzten Schlüssel muss per Definition den vorherigen Schlüssel ergeben, der auf die selbe Weise bereits authentisiert worden ist. Erst wenn der jeweilige Knoten die Authentizität des Schlüssels K_i überprüft hat, wird er die Signatur der gespeicherten Nachrichten aus dem Intervall *i* überprüfen – dazu wird das bereits für SNEP gebrauchte RC5-MAC-Verfahren eingesetzt (s. Kapitel 4.2.3)
Durch die Verzögerung bei der Schlüsselfreigabe kommt die Asymmetrie der Einwegfunktion zum Einsatz, denn sowohl ehrliche Knoten als auch der Gegner verfügen zum Intervall *i* nur über die Schlüssel K_0 bis K_{i-1}. Wenn ein Knoten also eine Nachricht im Intervall i empfängt, und im Intervall i+δ den richtigen Schlüssel dazu bekommt, kann er sicher sein, dass die Nachricht nur von der Basisstation verschickt worden sein kann, denn nur die Basisstation besitzt die passende Schlüsselkette und niemand sonst hätte aus den bisherigen Schlüsseln den nächsten berechnen können, wenn er nicht davor die Schlüssel in umgekehrter Reihenfolge generiert hätte. Damit diese Eigenschaft immer gilt, müssen folgende Punkte allerdings eingehalten werden:

- Ein Knoten darf keine Nachrichten akzeptieren, die älter als δ Intervalle sind
- Die Verzögerung δ darf nicht zu kurz gewählt werden, z.B. kürzer als die Round-Trip-Zeit im Sensornetz. Andernfalls könnten die ersten Sensorknoten, die den Schlüssel K_i bereits von der Basisstation erhalten haben, damit eigene Nachrichten signieren und sie an diejenigen Knoten verschicken, die sich weiter weg von der Basisstation befinden

4.4. Diskussion

SPINS besteht, wie bereits gesehen, aus 2 Protokollen, welche gemeinsam die Anforderungen der Vertraulichkeit, Authentizität, Aktualität und authentischen Broadcast garantieren. Durch den Einsatz des Protokolls entstehen allerdings folgende wichtige Einschränkungen, die entweder den Einsatz des Sensornetzes oder die Sicherheit betreffen:

- Das Sensornetz benötigt eine zentrale Basisstation, die sowohl einen Flaschenhals für die Kommunikation als auch den zentralen Angriffspunkt für den Gegner darstellt.
- Diese Basisstation muss über genügend Ressourcen verfügen, um Master-Keys und Zählerwerte für jeden Sensorknoten speichern zu können, und ebenfalls die Schlüsselkette für den authentischen Broadcast.
- Jeder Sensorknoten, der am Netz teilnimmt, muss über einen vordefinierten Master-Key verfügen, der auch der Basisstation bekannt sein muss. Das macht das Sensornetz unflexibel.
- Die Sensorknoten müssen über eindeutige IDs verfügen, die z.B. aus ihren Positionsdaten abgeleitet werden können. Aufgrund von geringer Grösse und grosser Stückzahl könnte eindeutige Identifizierung der Sensorknoten ein unlösbares Problem sein.
- Das Sensornetz muss über synchrone Zeitphasen verfügen. Da jeder Sensorknoten über eine eigene Uhr verfügt, und jede Uhr einen gewissen Uhrdrift aufweist, müssen alle Uhren von Zeit zur Zeit resynchronisiert werden.
- Sensorknoten müssen über genügend Speicher verfügen, um alle Nachrichten δ Intervalle lang zwischenspeichern zu können. Bei den sehr limitierten Ressourcen der einzelnen Knoten muss die Wahl der Parameter Intervallslänge und Zeitverzögerung gut überlegt sein.

4.5. Auswertung

Die am meisten eingeschränkte Ressource bei Sensornetzen ist die Energie. Daher ist der Energieverbrauch durch zusätzlich eingeführte Sicherheitsmechanismen von grosser Bedeutung für die Entscheidung, ob sie in der Praxis tatsächlich eingesetzt werden. Wie man es dem Diagramm in Abbildung 11 entnehmen kann, wird die meiste Energie bei Sensornetzen durch die Kommunikation verbraucht, der grösste Overhead entsteht dementsprechend durch das verschicken der zusätzlichen 8 Byte pro Signatur, was bei mittlerer Nachrichtengrösse von ca. 30 Byte einleuchtend ist. Im Vergleich dazu sind alle zusätzlichen Berechnungen für den Energieverbrauch nicht relevant.

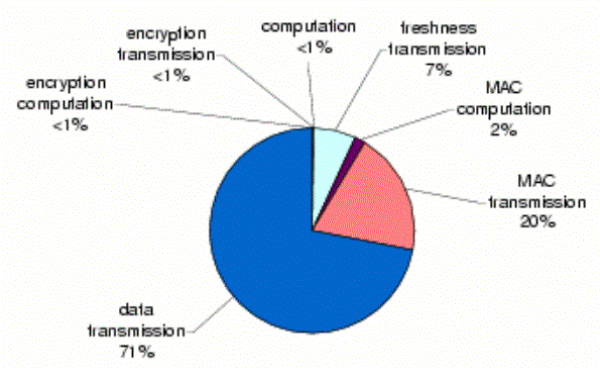

Abbildung 11: Energieverbrauch bei SmartDust mit SPINS

5. RESÜMEE

Diese Ausarbeitung zeigt einige Möglichkeiten, Sicherheit in Sensornetzen auf verschieden Ebenen des OSI-Modells zu implementieren. Wichtig ist dabei zu beachten, dass in verschiedenen Beispielen oft verschiedene Voraussetzungen gegeben sind, sowie auch Sicherheitsziele anders definiert werden, was sie komplementär zueinander macht. In meinem Seminarvortrag habe ich 2 verschiedene Themen herausgegriffen: Denial-Of-Service-Angriffe und SPINS-Protokolle. Das erstere beschäftigt sich mit Verfügbarkeit des Sensornetzes auf den unteren Ebenen des OSI-Modells, das letztere führt ein Protokoll ein, welches mit Hilfe von symmetrischer Kryptographie Vertraulichkeit, Authentizität und Aktualität sowohl für Unicast als auch für Broadcast garantiert.

Allgemein wird auf dem Gebiet Sicherheit für mobile Ad-Hoc-Systeme viel Forschung betrieben. Vor allem mit dem Thema der Sicherheit der Routing-Protokolle bei verteilten Netzen ohne vordefinierte Infrastruktur und mit vielen autonomen Knoten beschäftigen sich viele Forscher [7] [12] [13]. Das andere Problemgebiet ist das Schlüsselmanagement. Die Lösung mit den vordefinierten Schlüsseln bedeutet grosse Einschränkungen in der Praxis und widerspricht einer der Hauptidee der Sensornetze: sie ist nicht flexibel. Daher suchen ebenfalls viele Forscher nach einer geeigneteren Lösung [6] [8] [9] [10] [11]. Diese Arbeiten sind allerdings nicht direkt auf Sensornetze übertragbar, denn sie setzen allesamt die Publik-Key-Kryptographie voraus, was wiederum viel grössere Hardwareanforderungen nach sich zieht.

Auf dem speziellen Gebiet der Sensornetze ist bisher noch vieles offen. Es existieren einige Arbeiten, die sich mit dem Thema Sicherheit in Sensornetzen allgemein beschäftigen, ohne konkrete Implementierungen vorzuschlagen [5] [14]. Das SPINS-Protokoll scheint der aktuelle Stand der Entwicklung zu sein, viele Arbeiten beschäftigen sich mit dessen Eigenschaften [15].

Dennoch ist Sicherheit in Sensornetzen im begrenzten Raum möglich. Das Ressourcenverhältnis spielt bei Sensornetzen eine viel wichtigere Rolle als bei herkömmlichen Netzen, so dass die Entscheidung, welche Sicherheitsanforderungen genau notwendig sind und in welchem Mass, viel genauer überlegt sein soll. Bereits zur Designzeit eines Sensornetzes muss Sicherheit in die Entscheidungen miteinbezogen werden, da sie grundlegende Bestandteile wie Netztopologie oder Routing unmittelbar betrifft.

Sicherheit bedeutet des weiteren immer einen Overhead, den zu minimieren das Thema der Forschung sein soll. Das Thema Sicherheit für Sensornetze lässt somit noch viele Fragen offen und bleibt nach wie vor ein spannendes Forschungsgebiet für weitere Arbeiten.

6. REFERENZEN

[1] Anthony D. Wood, John A. Stankovic
”Denial of Service in Sensor Networks”
IEEE Computer, 2002

[2] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar
”SPINS: Security Protocols for Sensor Networks”
University of California, Berkeley
Mobile Computing and Networking, 2001

[3] B. Warneke, M. Last, B. Leibowitz, K. S. J. Pister
”Smart Dust: Communicationg with a Cubic-Millimeter Computer”
IEEE Computer Magazine, Jan. 2001

[4] Ronald L. Rivest
”The RC5 Encryption Algorithm"
http://theory.lcs.mit.edu/~cis/pubs/rivest/rc5rev.ps, 1995

[5] Sasha Slijepcevic, Vlasios Tsiatsis, Scott Zimbeck, Mani Srivastava
”On Communication Security in Wireless Ad-Hoc Sensor Networks”
IEEE WETICE’02

[6] Stefano Basagni, Kris Herrin, Danilo Bruschi, Emilia Rosti
”Secure Pebblenets”
ACM MobiHOC 2001

[7] Vesa Kärpijoki
“Security in Ad Hoc Networks“
Helsiniki University of Technology

[8] Jean-Pierre Hubaux et al.
“The Quest for Security in Mobile Ad Hoc Networks”
ACM MobiHOC 2001

[9] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu,Lixia Zhang
”Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks”
University of California, Los Angeles

[10] Bocheng Lai, Sungha Kim, Ingrid Verbauwhede
“Scalable Session Key Construction Protocol for Wireless Sensor Networks“
University of California, Los Angeles

[11] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, Sushil Jajodia
”Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach”
University of California, Irvine

[12] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, Herbert Rubens
“An On-Demand Secure Routing Protocol Resilient to Byzantine Failures“
ACM WiSe’02, 28. September 2002

[13] Chris Karlof, David Wagner
”Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures”
University of California, Berkeley

[14] Y. W. Law, S. Dulman, S. Etalle, P. Havinga
”Assessing Security-Critical Energy-Efficient Sensor Networks”
University of Twente

[15] Y. W. Law, P. Hartel, S. Etalle
”Evaluation of the Applicability of SPINS”
University of Twente, 27. March, 2002