Composition of the Graduation Committee:

prof.dr.	P.H.	Hartel	University of Twente (promoter)
dr.	S.	Etalle	University of Twente (assistant promoter)
prof.dr.ir.	B.R.H.M.	Haverkort	University of Twente
dr.	P.J.M.	Havinga	University of Twente
prof.dr.	W.	Jonker	University of Twente
prof.	L.V.	Mancini	Università di Roma “La Sapienza”
prof.dr.ir.	H.J.	Sips	Technical University of Delft

This research was conducted within the EU project EYES (IST-2001-34734) and the Smart Surroundings project.

Series title: CTIT Ph.D.-thesis Series
Series number: 1381-3617
CTIT Number: 05-75

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Keywords: Wireless sensor network, security, cryptography, key management, jamming.

Printed by Ipskamp PrintPartners, Enschede, The Netherlands.
ISBN 90-365-2282-X

KEY MANAGEMENT AND LINK-LAYER SECURITY OF WIRELESS SENSOR NETWORKS

ENERGY-EFFICIENT ATTACK AND DEFENSE

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof.dr. W.H.M. Zijm,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, December 1, 2005 at 15.00

by

Yee Wei Law

born on 19 May 1976,
in Kuala Lumpur, Malaysia

This dissertation is approved by:
prof.dr. Pieter H. Hartel (promoter) and
dr. Sandro Etalle (assistant promoter)

Abstract

The relentless miniaturization of computers and advances in radio-based communications have given rise to a new exciting technology: *wireless sensor networks* (WSNs). A WSN is a network consisting of numerous small, low-cost, independent sensor nodes that communicate through wireless means. These sensor nodes are self-contained units composed primarily of a microcontroller, a radio, a battery, one or more sensors and some interconnecting circuits. As such, the sensor nodes have limited computing power and energy supply, and instead of performance, they are frequently operated with energy efficiency in mind. Sensor nodes are meant to be deployed without a pre-configured topology and operated unattended, but radio communications enable them to organize themselves. When data are available through their sensors, radio communications allow the data to be distributed across the network. For monitoring-type applications, e.g. monitoring of natural habitats, this presents unprecedented sensing scale and resolution. For other types of applications, at the same time providing a transparent interface between human activities and the surroundings, WSNs integrate seamlessly with the environment.

As with other technologies, WSN is not without its perils. When data are circulated in an unattended network, they might be leaked against the interests of the concerned parties. Operation-wise, there should be some means to ensure the availability of the service provided by the network, as well as a way to discriminate between legal and malicious requests to avoid executing tasks that might be disruptive to the network itself. This is where *security* comes in. These problems are largely categorized as information security and security against denial-of-service (DoS) attacks, and they are the foci of this thesis. The common thread that permeates through these two areas is *energy efficiency*. While security can be gained by tamper-proofing the hardware, increasing the computational power of the sensor nodes and so on, *practical* WSN security is a balancing act that is constantly in search of the highest level of protection that can be squeezed out of the judicious use of limited resources.

The thesis is divided into two parts. The first part of the thesis concerns defense-oriented issues: design, implementation and key management. The second part of the thesis looks at attack-oriented issues: link-layer jamming attacks. In short, this thesis presents six contributions in the field of WSN security:

Contribution 1: Design framework We propose the concept of the *system profile* and *critical system parameters*. A system profile is a set of four boolean-valued critical system parameters that describe a system or application. Any application described by the same set of critical system parameters belongs to the same system profile. Any security architecture designed for a particular system profile can then be directly applied to the applications belonging to that same system profile. In the following, Contributions 2, 4, 5, 6 are applicable to all system profiles, whereas Contribution 3 is applicable to the system profile that best describes the class of networks in which all nodes are not tamper-resistant and cannot use public-key cryptography, but in which some nodes are more resource-rich than the others.

Contribution 2: Evaluation of block ciphers We present a *detailed* benchmark for a selection of block ciphers, one of the most important cryptographic primitives for WSNs. Taking into account the security properties, storage- and energy-efficiency of a set of candidates, we arrive at a systematically justifiable selection of block ciphers and operation modes. The

result should serve as a practical guideline on the suitable block cipher to use according to the constraint of available memory and the required level of security. This is the first systematic evaluation of block ciphers for WSNs in the literature and the first that focuses explicitly on energy.

Contribution 3: Key management We present ESA1, a lightweight, cluster-based, decentralized key management architecture for WSNs, covering the aspects of key deployment, key refreshment and key establishment. Our architecture uses only symmetric-key cryptography, and is based on a clear set of assumptions and guidelines. The balance between security and energy consumption is achieved by partitioning a system into two interoperable *security realms*: the *supervised realm* trades off simplicity and resources for higher security whereas in the *unsupervised realm* the opposite is true. Key deployment uses minimal key storage while key refreshment is based on the well-studied scheme of Abdalla et al. The novelty of this work lies at the concept of the security realms and the fact that the protocol suite is the first in WSNs to be formally verified.

Contribution 4: Secure multicast We present LKHW, a secure multicast scheme that is optimized for directed diffusion, and designed to facilitate secure data aggregation, an important operation primitive in WSNs. Like ESA1, the scheme is based on inexpensive symmetric-key cryptography. Due to its LKH heritage, LKHW is scalable, that is, in terms of efficiency, the re-keying overhead in terms of energy is approximately logarithmic to the group size. This is the first work that aims to harvest the synergy between directed diffusion and LKH.

Contribution 5: Link-layer jamming with detailed knowledge To show that the data link layer is an attractive entry point for energy-efficient attacks, we demonstrate both qualitatively and quantitatively using a set of novel metrics how listen interval jamming, CTRL interval jamming and data packet jamming affect S-MAC. Based on our results, we show that data packet jamming is better than the other attacks in censorship rate by a small margin, but CTRL interval jamming has the best lifetime advantage. As a countermeasure to data packet jamming, we propose a new technique based on data blurting and schedule switching.

Contribution 6: Link-layer jamming with minimal knowledge We derive several jamming attacks that allow the jammer to jam S-MAC, LMAC and B-MAC energy-efficiently with *minimal* knowledge and even when the packets are encrypted. Careful analysis of other protocols belonging to the respective categories of S-MAC, LMAC and B-MAC reveals that the protocols are also susceptible to our attacks, with minor adaptations. With typical WSN systems in use today no effective measures against link-layer jamming are possible. For WSNs that require high security against link-layer jamming we recommend (1) encrypting link-layer packets to ensure a high entry barrier for jammers, (2) the use of spread spectrum hardware, and (3) the use of a TDMA protocol.

Acknowledgements

First, I must thank the graduation committee members for their comments.

I owe these four amazing years to my supervisors Pieter Hartel and Sandro Etalle, without whose appreciation I would never have got this research position. Pieter is not only a mentor who teaches his students the correct ways of conducting research, but also a supervisor who genuinely cares about his students. For example, he lifted me out of misery when my laptop was stolen from my office, by approving a new laptop for me. For that alone, I would feel indebted to him for years to come, although my greatest gratitude to him should lie in how he fervently helped reviewing and polishing my papers. Anybody else would have saved himself the trouble of going through drafts upon drafts of the same old paper. It is sometimes hard to suppress the frustration associated with being unable to satisfy Pieter's rigorous requirement, but if I am a better me now than I was four years ago, it is undoubtedly because of Pieter.

Sandro englightened me about the proper ways of writing papers, and more generally about the various skills required for research. If not because of his encouragement, I would never have taken the possibly once-in-a-lifetime opportunity to visit the University of Rome "La Sapienza" for a 3-month research collaboration. I owe him dearly.

Paul Havinga was my project manager in EYES. I cannot thank him enough for his feedback on my research, the various opportunities he has introduced me to, and his approval of funding for the items that I need for my research.

I should have made it clear earlier that I also owe these four years to Cheun Ngen Chong (张俊年) or Jordan in short, my fellow Malaysian, ex-secondary schoolmate, and fellow University of Southampton alumnus. He is the one who recommended me to Pieter while I was taking my Master in Nanyang Technological University, Singapore. We have shared many experiences in and outside office. I cannot even begin to count the number of things that I am indebted to him.

Ricardo, even though showed me no mercy in Unreal Tournament, did not hesitate to help me with many things. Together with Vasughi Sundramoorthy, they are simply the icons of fun in the office. Ricardo, Vasughi, Jordan and I have had some wonderful time in numerous workshops.

Roberto Di Pietro was my excellent co-author. His ability to think outside the box can only be matched by his exotic Italian humor and whirlwind dance moves. His generous friendship and hospitality has allowed me to see Rome and Pisa in a different light. The memories of riding in his scooter on the frantic streets of Rome are something that I will recount to my grandchildren with profound nostalgia.

Jeroen Doumen and Jerry den Hartog were also my excellent co-authors. In particular, Jeroen was my tour guide on the treacherous land of cryptography. He answered whatever questions I threw at him without a single frown. I cannot express enough my gratitude.

Supriyo Chatterjea, Yuanqing Guo (郭园青), Nikolay Kavaldjiev and Jian Wu (吴剑) are simply fantastic beings, with whom I have never hesitated to share my thoughts with. I especially treasure the lunches I enjoyed with Nikolay, and the gatherings that Yuanqing, Jian and I have had together.

I have been bestowed with generous help by all the members of the EYES project: Stefan Dulman, Lodewijk van Hoesel, Tjerk Hofmeijer and Tim Nieberg. Lodewijk once ignited a gleam of hope in me that I can eventually master Dutch, but I have to disappoint him until

further notice. Nevertheless for this effort, I will remain forever grateful.

Among other colleagues, Lodewijk Smit and Michel Rosien helped me with Dutch. Lodewijk in particular gave me invaluable help with LaTeX, especially for writing this very thesis. He has also been a great companion in the PWC 2003 conference. I also want to thank Maria Lijding, for giving me her computer table. Pierre Jansen helped me order the computer equipments I needed. Ferdy Hanssen gave me pointers on Linux and Dutch. Paul Heysters and Omar Mansour helped me decipher the mind-racking Dutch terms that make up the tax forms. Marlous Weghorst, Annelies Klos and Nicole Baveld were the greatest secretaries. Omar Sinan Kaya gave me abundant advice on my trip to Istanbul. Leon Kleiboer gave me advice on Matlab. Richard Brinkman helped me manage my source code. Illeana Buhan and Ari Saptawijaya shared my work in reviewing papers without complaints. Ha, Marcin and Anna were wonderful companions in several outings. Lastly, it has been a great pleasure sharing the office with Leon, Kavitha, Hilbrandt, George, Roland, Mihai and Ozlem. To other colleagues from the DIES and CADTES group, whose names are too many to list, I thank them for making my stay here enjoyable.

Outside the university, Koen Langendoen and Gertjan Halkes, from TU Delft, have given me much help with their MAC simulator. I should especially thank Gertjan for his inexhaustible patience. Li-Hsing Yen (严力行) from Chung Hua University in Taiwan enlightened me about the essentials of graph theory. I thank them from the bottom of my heart.

On the other side of Europe, thanks are extended to Luigi Mancini, Chiara Petrioli and Antonio Durante who assisted me greatly during my stay in Rome. I have come to enjoy the whole package of Italian hospitality, humor and food that made my stay in Rome absolutely unforgettable.

Among the friends I have made outside office, I thank Anthony Lo for the pleasant conversations we have had as well as his guidance on communications-related subjects. I must also thank Dennis Vierkant who has never hesitated to answer my questions, no matter how trivial they were. Most importantly, I admire his dedication to his online Chinese-English dictionary, and his openness.

My friends whom I have known since high school back in Malaysia never hesitated to provide me with all kinds of help. Sometimes it was a matter of life and death, sometimes a little bit of advice and encouragment helped me through the day. I must thank them in earnest (蔡锦泰, 陈立辉, 陈淑晴, 邓为隆, 刁建鸿, 何俐斯, 洪文钞, 黄伟良, 李文政, 廖浩辉, 林德全, 林洧峰, 林秀慧, 卢立融, 罗待诗, 谭于琳, 温世明, 翁健瑞, 徐悦欣, 严晓慧, 张嘉田, 张银河, 郑立礼). I have also come to make some great friends in Singapore (朱超) and in the Netherlands (陈微, 王颖) who have never given up on me.

The fact is, I would not have made it without Jieyin Cheng (程洁音), my source of courage and determination. Her affection and patience helped me through the most difficult of days. Nothing else could have redefined the priorities of my life.

Lastly, I am forever grateful to my supportive family (王瑞琴, 罗盛南, 梁美兰, 罗裔敏, 陈镇欣, 罗裔闻). It is not an overstatement that my parents have given their everything to the best interest of their children. For the Saturdays they made me spend on piano lessons, for the Sundays they let me spend on Tae Kwon Do trainings, for the normal days they let me spend on my studies and my studies alone, and for the trust they place in me, I wish to say more than words allow me to. Just as there is no greater love than theirs for me, there is no greater debt than mine to them.

Yee Wei Law (罗裔纬)

Contents

CHAPTER I

Introduction

The evolution of computers has come to the stage when it is finally feasible to grant a grain of dust intelligence, or at least that is what the Smart Dust team from University of California at Berkeley (UC Berkeley) aimed to achieve in 1999 [132]. What they call "dust" is also what we now call *wireless sensor nodes* (henceforth simply called *sensor nodes* or *motes* [137]). The motivation for this technology can be traced back to 1991 when Marc Weiser suggested that the most useful computer would be an invisible one [295]. In other words, in the ideal world, the physical environments around us would be imbued with such wisdom that they would help us complete our daily tasks, or simply make our lives easier without us even knowing it. Thus the vision of *ubiquitous computing* or *pervasive computing*, where the computers are not only invisible but also ubiquitous [296], was born.

The vision that was proposed more than a decade ago is fast becoming a reality, thanks to Moore's law and the recent advances in micro-electromechanical (MEMS) technologies [40]. With sensor nodes as the basic building blocks, *wireless sensor networks* (WSNs) are helping us achieve this vision. A WSN is a large-scale network comprising a large number of physically small and low-cost sensor nodes, that despite being deployed without a fixed topology, are capable of self-organizing among themselves to accomplish their pre-defined tasks.

However with every new technology comes a certain risk. We are living in the age of *insecurity*: Windows viruses roam the Internet, RFID car keys and electronic payment cards no longer earn our trust [38], and even so-called high-security locks can be picked in a matter of seconds [100]. It is thus the aim of this thesis to investigate a number of pressing open problems in the security of WSNs.

This chapter is organized as follows. First, the characteristics of a WSN are given. Applications of WSNs, both existing and envisioned, are then briefly described. The ensuing discussion then sheds light on how security becomes an issue, bringing out the focus of this thesis. An overview of the thesis then follows.

1.1 Characteristics

The characteristics of WSNs are discussed from two perspectives: from the nodes that make up the network, and from the network itself.

1.1.1 Nodes

Sensor nodes are generally small, light and cheap. In fact, some researchers put an upper limit on the size, mass and cost at 1 cm^3, 100 g and \$1 respectively [234] (100 g is a far cry from a grain of dust, so the term Smart Dust should not be taken literally). Although currently a typical commercial off-the-shelf (COTS) mote costs around \$200, the price is projected to fall to around \$5 in the near future [96]. Compared with traditional sensors, sensor nodes (1) occupy less physical space for inventory and transportation; (2) allow for more efficient and larger-scale deployment; (3) allow for higher sensing resolution; and (4) blend better into the sensed environment.

A typical sensor node has a sensing unit, a microcontroller, a transceiver and a power supply (Figure 1.1). Although single-chip components with integrated microcontroller and transceiver are beginning to appear on the market [208], they are usually IEEE 802.15.4/ZigBee^{TM}-compliant, which is not necessarily suitable for every type of WSN application. The communication medium is usually radio, with the alternative being infrared.

There are many types of sensors, e.g. thermistors for measuring temperature; accelerometers for measuring acceleration; magnetometer and micro-power impulse radar (MIR) sensors for detecting metallic objects; barometric pressure sensors; dielectric-based humidity sensors; acoustic sensors; sensors for various types of chemicals etc. The nature of the sensors may affect the cost and physical size of the sensor nodes, but does not affect the general characteristics of WSNs (discussed in the next section).

The microcontroller usually has a low-power 8-bit or 16-bit RISC core and little on-chip storage that consists of RAM, Flash memory and/or EEPROM. The microcontroller usually supports several sleep modes and operates at a maximum frequency of a few MHz.

The transceiver usually has a single channel, a low data rate, and operates at unlicensed bands at 902-928 MHz and near 2.4 GHz. The frequencies are chosen as a trade-off between the higher energy cost at higher frequencies, and the lower antenna gain for compact antennas at lower frequencies [19].

The power supply usually comes in the form of alkaline, NiMH, NiCd or LiSO2 batteries, and sometimes with a solar array as a supplement [317], although battery-less, energy-harvesting (or energy-scavenging) nodes can already be found on the market [89], and research is undergoing in the generation of power from low-level vibrations, airflow, ambient light, thermal gradients etc [180, 247].

From an operational point of view, it is also worth mentioning that sensor nodes might or might not have addressable global identification (ID). This fact affects how protocols and security schemes are designed for WSNs.

Table 1.1 compares a few well-known types of sensor nodes both on the market as well as in academia. Among microcontrollers, 4 MHz Atmel's and the 8 MHz MSP430F149 are the more popular choices, whereas Chipcon and RFM are popular among radio transceivers. 4 KB of RAM and 128 KB of program memory should be considered the current upper limits. All in all, sensor nodes are notably characterised by their limitations – slow processor, little storage, low bandwidth and scarce energy reserve. Among these, energy is the most severe constraint. For example, when used with MICA2 motes, the effective lifetimes of alkaline, NiMH and LiSO2 batteries are 18 hours, 100 hours and 30 days [235] respectively, while these motes are meant to operate, ideally, for years. To make things worse, battery technologies advance much more slowly than semiconductor technologies do, e.g. the energy densities (both per unit volume and per unit mass) of Li-ion batteries have only increased 50% in 1999 compared to 1994 [33]. As a result, energy efficiency becomes the primary

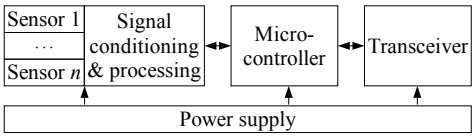


Figure 1.1: Components of a typical sensor node.

optimization parameter for most applications and the focus of this thesis is energy efficiency.

Table 1.1: Comparison of sensor nodes (listed roughly in chronological order).

Node	Microcontroller				Transceiver		
	Type	Clock freq (MHz)	RAM (KB)	Program memory (KB)	Type	Freq (MHz)	Max. data rate (kbps)
weC [175]	Atmel AT90S2313[a]	4	0.5	8	RFM TR1000	916.5	10
UCLA/Rockwell WINS (`http://wins.rsc.rockwell.com`)[b]	Intel StrongARM SA-1100	133	1024	4096	Conexant Systems RDSSS9M	900	100
Rene mote [175]	Atmel AT90LS8535	4	0.5	8	RFM TR1000	916.5	10
Rene2 mote [175]	Atmel ATmega163	4	1	16	RFM TR1000	916.5	10
MIT μAMPS [185, 262]	Intel StrongARM SA-1100	206	16384	512	National Semiconductor LMX3162	2400	1024
Crossbow MICA [69]	Atmel ATmega128L	4	4	128	RFM TR1000	433/ 915	40
Crossbow MICA2DOT [69]	Atmel ATmega128L	4	4	128	Chipcon CC1000	315/ 433/ 915	38.4
Crossbow MICA2 [69]	Atmel ATmega128L[c]	7.37	4	128	Chipcon CC1000	315/ 433/ 915	38.4
Crossbow MICAz [69]	Atmel ATmega128L	4	4	128	Chipcon CC2420	2400	250
EYES (Nedap) [286]	TI MP430F149	8	2	60	RFM TR1001	868.35	57.6
EYES (Infineon) [105]	TI MP430F149	8	2	60	Infineon TDA 5250	868- 870	64
BTnode rev3 (`http://www.btnode.ethz.ch`)	Atmel ATmega128L	7.37	244[d]	128	Chipcon CC1000	433- 915	38.4
Intel Research mote (3rd generation) [147]	Atmel ARM7TDMI	12	64	512	Zeevo Bluetooth	2400	600
Moteiv Telos (Rev A) [191]	TI MP430F149	8	2	60	Chipcon CC2420	2400	250
ZebraNet node (Version 3) [317]	TI MP430F149	0.03/8[e]	2	60	MaxStream 9XStream	900	19.2

[a] A smaller version of the AT90S8535.
[b] First generation prototype "AWAIRS 1" has 128 KB of RAM and 1 MB of Flash memory [10].
[c] The first 1000 used ATmega103L, the rest use ATmega128L in ATmega103L compatibility mode [225].
[d] 4 KB on-chip, 240 KB off-chip.
[e] 8 MHz while GPS, radio and Flash modules are running; 32 kHz otherwise.

1.1.2 Network

WSN is an application-specific technology – for example, temperature sensor nodes only measure temperature and nothing else. It is due to this very nature that WSNs can range from dense to sparse, from mobile to static. However, the following characteristics in general apply:

- **Scale**: WSNs must be scalable to a network size at least of the order of thousands, since sensor nodes are cheap, and adding new nodes to the network, either to expand the network or to replace the nodes that have failed, without incurring significant overheads, is a basic requirement.

- **Local computation**: Benefiting from the computational resources of individual sen-

sor nodes, local computation allows raw data obtained over a period of time to be buffered, processed, aggregated with data from neighboring motes, and transmitted at a later stage. This is one way of reducing the number of transmissions, one of which consumes 3 orders of magnitude more energy than the execution of 1 instruction [228]. This functionality is commonly known as *in-network processing* or *data aggregation* [119, 228].

- **Multihop**: Transmitted packets frequently have to 'hop' from one node to another to reach the endpoint, because (1) embedded radio transceivers have low transmission power ($\approx$ 0 dBm) and low range ($\approx$ 20 m); (2) the transmission energy increases exponentially with the transmission range. However Min et al. [184] remind us that multihop saves energy only when the path attenuation dominates the static energy consumption of the hardware.

- **Network traffic**: Embedded radio transceivers have little buffer space, and the requirement of a long network lifetime usually means that even if sensor data are transmitted periodically, the data rate is low. However phenomena that trigger the sensors also trigger bursty traffic, e.g. a fire monitoring system only becomes active in the event of a fire.

- **Data-centric**: Redundancy is built into the system – one node is not more nor less important than the others. Instead of addressing a specific node, the semantics of the system is frequently that only the data matters. This is the reason why sensor nodes might not have IDs, and a whole new data-oriented communication paradigm is called for [82, 99].

1.2 Applications

It is probably not an overstatement that WSN technologies are applicable wherever the imagination reaches. In fact, WSNs are being or will be used across a broad spectrum of military, civil and commercial applications [12]:

Military Much of today's technological achievements arose from military needs (e.g. the Internet). This remains true for WSNs especially because WSNs have a wide range of military applications. WSN for example is the key technology towards detection of nuclear, biological and chemical (NBC) attacks [126]. The sensors for plastic explosives [222] and TNT [127] already exist. Coupled with the discovery of an effective way to propel MEMS robots [127], the detection process can further be streamlined. In August 2002, the Department of Defense (DoD) in the US announced that bioweapons sensors [76] would be deployed in certain cities [128]. To detect potentially dangerous radioactive sources in the cities, low-cost radiation sensors that are no bigger than a wristwatch can be distributed to police officers, firefighters and postal workers, to form large-scale mobile detection networks [128]. The Defense Advanced Research Projects Agency (DARPA) has been funding a number of high-profile WSN projects to complement its Command, Control, Communications, Computing, Intelligence, Reconnaisance and Targeting (C4ISRT) systems, most notably through its SensIT and NEST programs. In one well-known experiment funded by SensIT and carried out by researchers from UC Berkeley at the Marine Corps Air/Ground Combat Center in Twentynine Palms of California, an unmanned aerial vehicle (UAV) successfully (1) dropped sensor nodes onto a road to track ground vehicles, (2) collected the

track information from the network and (3) transfered the information back to the base camp [223]. Obviously, vehicle tracking emerges as an important application for the military [113]. Another application that has gained DARPA's attention is surveillance and control. In a project entitled "Quality of Service in Surveillance and Control", researchers from 3 universities are looking at the resource management aspects of DoD's surveillance system consisting of phase array radars, sensor networks and advanced avionics components (`http://www-rtsl.cs.uiuc.edu/muri`). UCLA has a number of contracts with DARPA for developing next generation low-power Wireless Integrated Network Sensors (WINS) (`http://www.janet.ucla.edu`). One extreme example of the use of WSNs in modern warfare is a DARPA-funded project that develops mobile anti-tank land mines that shift position automatically to fill gaps after other mines have detonated [83].

Diaster detection and relief Many lives can be saved if disasters like floods, wildland fires, earthquakes, tsunamis, volcanic eruptions are duly detected at the early stage of their development. WSNs can be used to monitor rivers and functional floodplain environments (`http://envisense.org/glacsweb.htm`). For fire detection, WSNs can be used to collect real-time data from wildfires as a basis for predictive analysis of evolving fire behavior [39]. WSNs embedded in buildings can pick up the early signs of an earthquake that come in the form of seismic waves [148]. Tsunamis can be detected with seismic, hydroacoustic, and infrasound sensors [248]. For monitoring volcanic activities, WSNs can be used to collect seismic and infrasonic signals [297].

Industry Asset monitoring, asset tracking, inventory control are among the industrial applications of WSNs. Examples of asset monitoring include (1) BP's use of WSNs in detecting harzadous storage condition for its petrochemical products [149] and in continuous vibration monitoring of the engines of its oil tankers; (2) Intel's pilot deployment of WSNs in monitoring the health of its semiconductor fabrication equipments [122]; (3) Boeing's use of its "pressure belt" sensor networks for measuring pressure distribution on the wing surfaces of its airplanes [52]. An example where the use of WSNs becomes critical is the monitoring of the temperature of oil pipelines near the Arctic Circle, which if not heated, would burst due to clogging [97]. Examples of asset tracking include BP's use of WSNs in tracking its railcar fleets and the contents of its railcars in North America [149]. Some companies are already selling WSN solutions for inventory control.

Agriculture WSNs can be used to monitor temperature, moisture level and soil conditions for example in vineyards, or to monitor the storage conditions of crops like sugar beets [83]. Cattles in a herd resemble nodes in a WSN. If a node is attached to each animal, then the herd as a whole can easily be tracked, and the movement of which can even be controlled [26]. WSNs can also be used to monitor cattles' health [177].

Environmental monitoring Due to their low impact and low visibility, wireless sensors are suitable for monitoring floras [308] and faunas [169, 317] in their natural environments, and the environment itself (`http://envisense.org/glacsweb.htm`). WSNs can also be used to complement satellites in monitoring environments [54]. There are naturally concerns about the impact of deployed sensor nodes to the environment. It remains to be seen how large scale deployments of WSNs will be executed in an environment-friendly manner.

Intelligent buildings Rabaey et al. [233] estimate that with the use of sensor network technology, the energy consumption for space conditioning of commercial buildings (the dominant form of energy consumption in commercial buildings) can be reduced by 44%. Since

the cost of installing a wireless sensor is 10 to 100 times less than that of a wired sensor, the majority of which is due to wiring, WSNs provide significant advantages [210]. Systematic monitoring and control of temperature, airflow and lighting top the list of research agenda [60]. As a step towards context-sensitive ubiquitous computing, WSNs can be used to deduce the motion pattern of the inhabitants inside a building [302].

Health/medical WSNs embedded in garments can be used to monitor the ambulatory activities of the elderly, for example to detect when they are going to fall [205]. In Intel's Proactive Health lab, WSNs also help carers make sure the elderly have enough social contact, and take proper medication on time [121]. For emergency response, wearable sensor networks can provide measurements of patients' vital signs like heart rate, oxygen saturation, endtidal CO_2 and serum chemistries, thereby improving pre-hospital and in-hospital emergency care [170]. The same networks can store patients' identification and medical records, as a supplement to the hospital's IT infrastructure [170].

Law enforcement WSNs can be used to detect gunshots. Acoustic sensors can be deployed around areas of high crime rate to capture the sounds of gunshots. The sensors that pick up the sounds can then trigger the camera to zoom in on the shooter and alert the law enforcement authority [265]. WSNs can also make it convenient for officers to access databases, surveillance cameras and reporting tools on the field, without depending on public cellular networks [182].

Transportation Beacons can be installed along roads to broadcast traffic information, allowing commuters to avoid congested roads and to find the fastest routes to their destinations [20]. Multihop inter-vehicular communications facilitate the dissemination of information regarding neighborhood public facilities, gas stations and shops, besides traffic information. Sensor networks can also be found in vehicles for monitoring tire pressure – an application invaluable for road safety and especially useful for road trains that carry heavy freights.

Space exploration Perhaps the most headline-grabbing application of all – NASA's Jet Propulsion Laboratory is planning to use WSNs for exploring other planets and even asteriods in its Sensor Webs initiative (`http://sensorwebs.jpl.nasa.gov`).

1.3 Security

None of the applications mentioned in the previous section would function correctly if appropriate security measures are not taken. Table 1.2 lists some of the potential security threats that can be expected from the absence of security mechanisms.

The security of WSNs can be classified into two broad categories: (1) operational security, and (2) information security. The operation-related security objective is that a network as a whole should continue to function even when some of its components are attacked (the *service availability* requirement). The information-related security objectives are that *confidential* information should never be disclosed, and the *integrity* and *authenticity* of information should always be ensured. These objectives are marked with a cross in Table 1.2 if they are violated in the corresponding scenario.

While it may seem that information security can readily be achieved with cryptography, there are 2 facts that make achieving the above objectives non-trivial in WSNs: (1) sensor nodes operate unattended – they are potentially accessible, both geographically and physically, to any malicious party imaginable; (2) sensor nodes communicate through an open

Table 1.2: Potential security threats, grouped according to application domains. SA=Service availability, C=Confidentiality, I=Integrity, A=Authenticity.

Application domain	Potential security threats	Properties violated			
		SA	C	I	A
Military	• Denial-of-service attacks by means of jamming and/or confusing the networking protocols. • Eavesdropping of classified information. • Supply of misleading information, e.g. enemy movements in the East where in fact they are in the West.	×	×	×	×
Disaster detection and relief	• Supply of misleading information, e.g. bogus disaster warnings, by pranksters, causing huge financial loss as a result of unnecessary large-scale evacuation and deployment of relief equipments.				×
Industry	• Eavesdropping of commercial secrets by business rivals. • Intentional disruption of manufacturing processes as a result of misleading sensor readings caused by disgruntled employees or business spies.	×	×	×	
Agriculture	• The agricultural department might want to deploy WSNs to ensure that farmers do not overuse pesticides or other hazardous chemicals on their crops, but unscrupulous farmers might tamper with the sensor nodes.			×	
Environmental monitoring	• Suppose government-endorsed environmental sensors are installed near a factory to monitor air/water quality to make sure the factory's emission lies beneath the pollution threshold, however by feeding the sensors with wrong information, the factory allows itself to escape detection and let its polluting emission go unchecked.			×	
Intelligent buildings	• Biometrics-based access control mechanisms are compromisable if the biometric sensors can be bypassed or fooled. • Token-based access control mechnisms are compromisable if the token authentication protocol is insecure.	×		×	×
Health/medical	• Providing wrong physiological measurements of a patient to the carer or doctor, a miscreant may cause potentially fatal diagnosis and treatment to be performed on the patient.			×	×
Law enforcement	• If criminals are able to eavesdrop the databases of the police departments, or to misguide the detection of gunshots, or to disrupt the network, public safety will be affected.	×	×	×	×
Transportation	• There is no order in the city when traffic information can no longer be trusted because they can easily be spoofed.			×	×
Space exploration	• Space agencies invest billions into space exploration projects, it is only logical that they want to ensure all commands executed on their space probes are authorized, and all collected data encrypted and authenticated.	×	×	×	×

medium. The first fact makes insider attacks possible, when the soft belly of every node is up for grabs – it is easy for an attacker to gain access to the data (including system states and cryptographic material) and programs that power the devices, and even modify the software to run its own algorithms. The second fact makes network-based attacks [269] easier compared with wired networks, as an open medium can be thought of as an open buffer where an attacker can read from and write to whenever it likes. In the following, we first refine our broad classification of the attacks against WSNs based on the idiosyncrasies of WSNs, then we outline the challenges facing the researchers in this field. Based on this taxonomy and the challenges, we will then lay down the research questions that motivate our research in Section 1.4.

1.3.1 Taxonomy of Attacks

A taxonomy of the potential attacks against WSNs allows us to understand the attacks better and helps us formulate our security strategies accordingly. The attacks are divided exclusively into two types: (1) host-based attacks, and (2) network-based attacks (a WSN consists of hosts and the network and nothing else). We first discuss host-based attacks, which can further be broken down into:

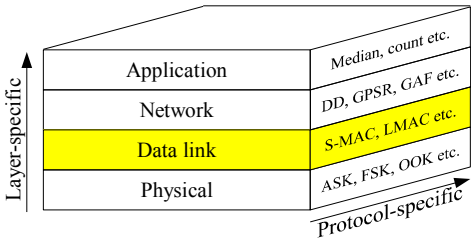
1. **User compromise**: This involves compromising the users of a WSN, e.g. by cheating the users into revealing information such as passwords or keys about the sensor nodes. As sensor nodes do not have or need human operators, this kind of attacks are irrelevant, unless we consider the base station that stores the keys used in the WSN. But even for the case of the base station, a myriad of technologies already exist that can be used to safeguard the security of the base station [13], therefore user compromise is not further investigated in this thesis.

2. **Hardware compromise**: This involves tampering with the hardware [14] to extract the program code, data and keys stored within a sensor node. The attacker might also attempt to load its program in the compromised node. Although sensor nodes are generally known to be lacking in tamper resistance, this kind of attacks are hardware-specific, and compromising a large part of the network typically requires compromising a large number of nodes – a prohibitive cost. Therefore, hardware compromise *by itself* is not an effective attack against WSNs, and it is not further investigated this thesis.

3. **Software compromise**: This involves breaking the software running on the sensor nodes. Chances are the operating system and/or the applications running in a sensor node are vulnerable to popular exploits such as buffer overflows. This is more so for operating systems that support dynamic loading of code module over the network [111]. However we do not address host-based software compromises in this thesis, because we are not dealing with any particular OS or software application – our results are independent of these two aspects. Moreover, writing secure code is an established discipline [101], and techniques are being developed to facilitate the safe execution of untrusted code [199], therefore software compromise is not further investigated in this thesis.

After discussing host-based attacks, we now discuss network-based attacks. We can look at network-based attacks from two orthogonal perspectives: layer-specific compromises, and protocol-specific compromises (Figure 1.2). An attack consists of one or more of the following operations [269]:

1. **Interrupting messages**: This operation threatens service availability. The main purpose of this operation is to launch denial-of-service (DoS) attacks. On the physical layer, this threat is known as radio jamming, i.e. the emission of a radio signal directed at one or more receivers to disrupt the reception of the receivers. In general, the jammer signal power has to be 5 dB to 10 dB times (higher) the received signal power to be effective [252]. On the data link layer, it is called link-layer jamming [300]. Instead of blasting out radio signals blindly, in link-layer jamming, the attacker exploits the characteristics of the MAC protocol (throughout this thesis, 'MAC' alone refers to



Note:

- The application layer sits on top of the protocol stack, and is different from application to application, e.g. it can be a protocol that simply collects readings from neighboring nodes, and sends the average of the readings to a fixed base station.
- The network layer is responsible for routing messages (datagrams) from one node, to one or more other nodes multiple hops away, based on some system-defined parameters such as energy and latency.
- The data link layer defines how messages (packets) are encoded and decoded, how errors are detected and corrected, the addressing scheme as well as the medium access scheme.
- The physical layer defines the actual methods used to transmit and receive messages (frames) through the radio interface, for example the frequency, the data rate, the signal modulation and the spread spectrum scheme, if any, to use.

Figure 1.2: Network-based attacks against WSNs from two perspectives.

message authentication code, but 'MAC protocol' refers to media access control protocol) used by its victims to send out jamming signals energy-efficiently. Interrupting messages on the higher layers is meaningless. For example, on the network layer, the attacker can interrupt messages by refusing to forward them, but this passive approach is less effective than the active approach of link-layer jamming, and for network-layer jamming to work, the attacker has to be able to interoperate with the MAC protocol of its victims. Therefore, link-layer jamming is an immediate threat that needs attention. The worst case is that an attacker can program and deploy a general-purpose link-layer jamming network that is able to identify the MAC protocol used by a WSN and jam the WSN effectively and energy-efficiently. This is to say, from the protocol-specific perspective, we need to investigate the anti-jamming properties of various mainstream MAC protocols.

2. **Intercepting messages**: This operation threatens message confidentiality. The main purpose of this operation is to eavesdrop on the information carried in the messages. From the layer-specific perspective, this operation is usually aimed at the application layer, specifically the payloads of datagrams, where most useful information lies. If the payloads are encrypted and the attacker does not have the corresponding key(s), the attacker can resort to cryptanalysis to infer the original payloads from the encrypted payloads. Key management is an important defense against successful cryptanalysis and will be further discussed as one of the foci of this thesis.

3. **Modifying existing messages**: This operation threatens message integrity. The main purpose of this operation is to confuse or mislead the parties involved in the communication protocol, e.g. by changing reported sensor readings, or by corrupting routing control packets, causing energy to be wasted on erroneous routing. From the layer-specific perspective, this operation is usually aimed at the network layer and the ap-

plication layer, because of the richer semantics of these layers. If the messages are encrypted and the attacker does not have the corresponding key(s), the attacker can, as before, resort to cryptanalysis to infer the key from the plaintext-ciphertext pairs observed in the network, and use the key to re-encrypt (successfully or not) its own modified messages. Furthermore, if the protocol format is undocumented, it is reverse-engineered according to the format of the (successfully or not) decrypted messages. Key management, again, is an important defense against successful cryptanalysis and will be further discussed as one of the foci of this thesis.

4. **Fabricating false messages**: This operation threatens message authenticity. The main purpose of this operation is to confuse or mislead the parties involved in the communication protocol, e.g. by reporting false sensor readings, or by propagating bogus routing errors, causing energy to be wasted on erroneous routing. This operation can also facilitate DoS attacks, by flooding the network, in the hope of overwhelming it with bogus messages, but in this case, this operation becomes the operation of interrupting messages, which has been discussed earlier. If the messages are encrypted and the attacker does not have the corresponding key(s), the attacker can, as before, resort to cryptanalysis to infer the key from the plaintext-ciphertext pairs observed in the network, and use the key to sign (successfully or not) its own fabricated messages. Furthermore, if the protocol format is undocumented, it is reverse-engineered according to the format of the (successfully or not) decrypted messages. Key management, again, is an important defense against successful cryptanalysis and will be further discussed as one of the foci of this thesis.

5. **Replaying existing messages**: This operation threatens message freshness. The main purpose of this operation is to confuse or mislead the parties involved in the communication protocol that is not time-aware. For example, it can be used for man-in-the-middle attacks against cryptographic protocols, and if the attacks are successful, would lead to the violation of confidentiality, integrity and authenticity, and even service availability. Because message freshness *by itself* is meaningless – it only helps safeguard other security objectives – it is not listed in Table 1.2. Time-aware cryptographic protocols is an important defense against this operation and will be further discussed in the context of key management, as one of the foci of this thesis.

6. **Deviating from protocol**: When the attacker is, or becomes an insider of the network, and the attacker's purpose is not to threaten the service availability, message confidentiality, integrity and authenticity of the network, but to gain an unfair advantage for itself in the usage of the network, the attacker manifests selfish behaviors, behaviors that deviate from the intended functioning of the protocol. For example, if the attacker wants to gain an unfair use of the transmission medium by not setting a non-zero contention time-out (which *does not* involve interrupting, intercepting, modifying, fabricating or replaying any message), its neighbors would starve of bandwidth, but when the attacker does not need the medium, its neighbors do get their share of the medium. WSNs are not user-centric, i.e. there is no user behind every sensor node, so unless an application exists where the network is shared between a number of human users, and there are practical motivations for selfish behaviors, this attack operation is an unlikely threat to WSNs.

To summarize the above taxonomy, key management and link-layer security against jamming are identified as two of the most important aspects of WSN security, and this thesis is devoted to them.

1.3.2 Challenges

After looking at the taxonomy of attacks, let us look at the current challenges security researchers are confronted with. The first challenge is the difficulty in distinquishing attacks (especially DoS attacks) from failures. For example, when a node fails to receive acknowledgements from a neighbor. It might be because there is an attacker that has blocked the acknowledgements, or the neighbor has simply moved out of radio range, or there is a temporary interference, or the neighbor has broken down or died of ordinary battery exhaustion. From the outcome alone, it is hard to tell whether a deviation from expected behavior is actually an intrusion or a failure. Worse, an intrusion may be disguised as intermittent failures (e.g. intermittent dropping of packets), and a failure may appear intrusive (e.g. when a series of transmitted packets become corrupted due to environmental effects). The absence of a clear line of distinction between the outward manifestation of attacks and failures is one of the biggest challenges in WSN security.

Another challenge results from the hardware constraints of sensor nodes, and the contradicting fact that security often incurs non-trivial overheads. An ideal security architecture includes (1) preventive measures like encryption and authentication that prevent attackers from tampering with the communication messages, (2) detective measures that detect anomalies in protocol executions and raise alarm if the anomalies exceed a certain tolerance threshold, and (3) reactive measures that trigger workarounds to the attacks, or even launch counter-attacks. All these efforts eat heavily into the resource budget, and under these constraints, achieving a comfortable security/performance *trade-off* becomes a daunting task:

- **Limited computational power** implies that to satisfy the real-time requirement of data processing, a sensor node can only afford little CPU time for security. Algorithms like public-key en/decryption which make extensive use of exponentiation often take too long, and therefore consume too much power.

- **Limited memory** implies that only algorithms with modest code size and runtime footprint can be executed. Algorithms like public-key en/decryption generally require significant amount of code storage and more variables than the RAM can fit. Block ciphers are generally cheaper, but modern ciphers like the AES tend to use tables that are big on a sensor node's scale.

- **Limited data rate** implies that security-related overhead has to compete with normal payloads for bandwidth. Transmission typically uses the most energy in a mote, and the energy for transmitting a packet is proportional to the length of the packet (provided the transmission time dominates the startup time of the radio [262], which is typical of most applications).

- **Limited communication range** implies information often travel multiple hops from the source to the sink. Any security-related overhead should not significantly worsen the already non-negligible latency.

- **Limited energy supply** implies that security-related overhead in computations and transmissions shortens the device's lifetime.

The lesson learnt by WSN security researchers so far is that instead of trying to achieve perfect security, the security paradigm in WSNs should be based on *tolerance* and *resilience* [307]. In other words, when it is not able to prevent attacks, a WSN should at least exhibit *graceful performance degradation* in the face of attacks, and recover as much as possible once the attacks subside. Although there is no official consensus, ideally the performance of the system under attack degrades no faster than a rate proportional to the ratio of compromised nodes to the total number of nodes in the network [137].

The third challenge has more to do with people than technology, namely the fact that in reality experts of communication protocols and security work in their own separate domain without close collaboration. Security designed as a separate module is often not compatible with protocols that typically assume a trusted and cooperative environment. Reasons such as time-to-market pressure or perhaps as simple as short-sightedness have resulted in security threats on a wide range of electronic devices nowadays. In fact, a report by IBM Security Intelligence Services reveals that many mobile phones, handheld computers, wireless networks and embedded computers (increasingly used to run basic automobile functions) are vulnerable to viruses, based on data from 500,000 electronic devices [59].

As a result, while much progress has been made in other areas, security is still widely deemed an open issue, with many problems remaining partially solved or even untackled [136, 214]. On a positive note, this also makes security research in WSNs unique and exciting.

1.4 Research Questions

The previous taxonomy of attacks and challenges allow us to formulate the research questions investigated in this thesis. We start by recalling that we have identified key management and link-layer security as two of the most important security aspects in WSNs. We found that it is best to look at these two problems from two different perspectives. Key management is best approached from a defense viewpoint, because key management is a preventive measure (it prevents crytanalysis). For link-layer security, we look from an attack viewpoint, because jamming is targeted at a specific protocol, and so we need to know more about how a jamming strategy might be devised against a MAC protocol before we know what countermeasure needs to be taken.

Defense From a top-down view, key management requires security architectures, protocols and algorithms. Different systems and different applications however have different defense requirements and resources, and hence require potentially different key management schemes. For example, an industrial application that operates in a well-controlled environment may assume the existence of trusted nodes/servers, but a military application on the battlefield may not. Ideally we should be able to design a key management scheme for a class of applications, and apply the scheme to any application that belongs to the class. The question is,

Question 1: Is it really possible to identify classes of WSNs with respect to security, and if yes what are the design parameters that define a class, and since for every class of applications, there is a matching key management scheme, what are the possible architectures, protocols and algorithms that can be applied to that key management scheme?

Attack As mentioned, we have to have a clearer idea about what link-layer jamming looks like. So the question is,

> Question 2: Which MAC protocols are susceptible to link-layer jamming attacks, and how effective and energy-efficient can these attacks be, and what can be done about them?

1.5 Contributions

Our contributions are the partial solutions we provide to each of the research problems given in the previous section. Contributions 1, 2, 3 and 4 correspond to Question 1. Contributions 5 and 6 correspond to Question 2.

Contribution 1: Design framework In Question 1, we ask if it is really possible to identify classes of WSNs with respect to security and if yes, what parameters define a class. As an answer to this question, we found a resounding 'yes' to the first part of the question, and for the second part, we propose the concept of the *system profile* and *critical system parameters*. A system profile is a set of four boolean-valued critical system parameters that describe a system or application. Any application described by the same set of critical system parameters belongs to the same system profile. Any security architecture designed for a particular system profile can then be directly applied to the applications belonging to that same system profile.

In Question 1, we also ask what the possible architectures, protocols and algorithms that can be applied to a key management scheme are. We tackle this question bottom-up, by first considering cryptographic algorithms in Contribution 2.

Contribution 2: Evaluation of block ciphers Some cryptographic algorithms might be applicable to some sensor nodes but not the others, since not all algorithms consume the same amount of memory and computation, let alone offer the same level of security. Thus, we present a *detailed* benchmark for a selection of block ciphers. Among all cryptographic primitives, we only consider block ciphers because they are the minimum necessary cryptographic primitives (details in Chapter 4). Taking into account the security properties, storage- and energy-efficiency of a set of candidates, we arrive at a systematically justifiable selection of block ciphers and operation modes. The result should serve as a practical guideline on the suitable block cipher to use according to the constraint of available memory and the required level of security. The recommended ciphers are applicable to all system profiles. This is the first systematic evaluation of block ciphers for WSNs in the literature and the first that focuses explicitly on energy.

Next, on top of a suitable set of algorithms, we need key management architectures and protocols. In conventional wired networks, two parties that do not share any secret key beforehand can establish a secure connection using protocols that are based on the Decision Diffie-Hellman assumption [36, 77]. The first problem with these protocols is however that they involve large exponentiations (of at least 1024 bits) [37]. The second problem is that the participants need to exchange public certificates and signatures that require a significant

amount of memory for storage and a lot of energy for transmission. As WSNs generally cannot afford these computational and storage requirements, it becomes a challenging problem to distribute keys such that any two nodes can communicate with each other, without deploying a shared key on *every pair* of nodes. In other words, if we do not deploy a key between every pair of *A* and *B*, how do we make sure *A* and *B* can communicate with each other securely? To answer this question, our first observation is however that we do not actually need to make sure every pair of nodes communicate secure, and in contrast, we only need to make sure nodes of the same security level communicate with each other securely. For example, there are nodes that are meant to carry sensitive information (high-security nodes) and nodes that are not, and we only need to make sure sensitive information circulate securely among the high-security nodes. For this purpose, we have Contribution 3.

Contribution 3: Key management We present ESA1, a lightweight, cluster-based, decentralized key management architecture for WSNs, covering the aspects of key deployment, key refreshment and key establishment. Our architecture uses only symmetric-key cryptography, and is based on a clear set of assumptions and guidelines. The balance between security and energy consumption is achieved by partitioning a system into two interoperable *security realms*: the *supervised realm* trades off simplicity and resources for higher security whereas in the *unsupervised realm* the opposite is true. Key deployment uses minimal key storage while key refreshment is based on the well-studied scheme of Abdalla et al. ESA1 is applicable to the system profile that best describes the class of networks in which all nodes are not tamper-resistant and cannot use public-key cryptography, but in which some nodes are more resource-rich than the others. The novelty of this work lies at the concept of the security realms and the fact that the protocol suite is the first in WSNs to be formally verified.

As a sub-problem of key management, we look at the efficient management of secure multicast groups (i.e. secure communication between more than 2 nodes) in Contribution 4.

Contribution 4: Secure multicast We present LKHW, a secure multicast scheme that is optimized for directed diffusion (a data-centric routing algorithm), and designed to facilitate secure data aggregation, an important operation primitive in WSNs. Like ESA1, the scheme is based on inexpensive symmetric-key cryptography. Due to its LKH heritage, LKHW is scalable, that is, in terms of efficiency, the re-keying overhead (for ensuring backward and forward secrecy) in terms of energy is approximately logarithmic to the group size. LKHW is applicable to all system profiles. This is the first work that aims to harvest the synergy between directed diffusion and LKH.

In Question 2, we ask which MAC protocols are susceptible to link-layer jamming attacks, but instead of considering all known MAC protocols at once, we first investigate the effectiveness and energy efficiency of link-layer jamming attacks on S-MAC [309], a simple and well-known MAC protocol. Our initial investigation assumes that the attacked sensor nodes do not encrypt their packets and the investigation results in Contribution 5.

Contribution 5: Link-layer jamming with detailed knowledge We demonstrate both qualitatively and quantitatively using a set of novel metrics how *listen interval jamming*,

CTRL interval jamming and *data packet jamming* impact S-MAC. Based on our results, we show that data packet jamming is better than the other attacks in censorship rate by a small margin, but CTRL interval jamming has the best lifetime advantage. As a countermeasure to data packet jamming, we propose a new technique based on data blurting and schedule switching.

Our follow-up investigation assumes the attacked sensor nodes encrypt their packets. We realize an attack is more powerful if it requires minimal knowledge of the target MAC protocol, especially when the messages to be jammed are encrypted. Encrypting the packets may help prevent the jammer from taking actions based on the content of the packets, but we found that the temporal arrangement of the packets induced by the nature of the protocol might unravel patterns that the jammer can take advantage of even when the packets are encrypted. We also extend our investigation to classes of protocols instead of just one protocol. Thus, we have Contribution 6.

Contribution 6: Link-layer jamming with minimal knowledge By looking at the packet interarrival times in three representative MAC protocols, S-MAC, LMAC and B-MAC, we derive several jamming attacks that allow the jammer to jam S-MAC, LMAC and B-MAC energy-efficiently. The jamming attacks are based on realistic assumptions. The algorithms are described in detail and simulated. The effectiveness and efficiency of the attacks are analyzed using the metrics introduced by Contribution 5. Careful analysis of other protocols belonging to the respective categories of S-MAC, LMAC and B-MAC reveal that the protocols are also susceptible to our attacks, with minor adaptations. This is a new insight into the security considerations of MAC protocols.

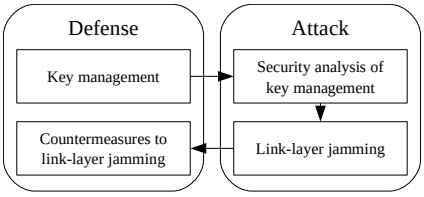


Figure 1.3: Research trail.

The research trail of this thesis is best visualized with Figure 1.3. In effect, our contributions can be classified as follows. Contribution 1 falls in the area of design and implementation. Contribution 2 belongs to the area of cryptographic primitives. Contributions 3 and 4 address key management issues. Contributions 5 and 6 are related to the data link layer. This thesis is thus devoted to the research areas shaded in Figure 1.4. These will be discussed in detail throughout the thesis. The other (unshaded) research areas will be discussed in less detail and only in the next chapter.

1.6 Thesis Overview

The rest of this thesis, with the exception of Chapter 2 and 9, is a compilation of published papers organized into separate chapters.

Chapter 2 presents a survey of the state of the art in the security of WSNs.

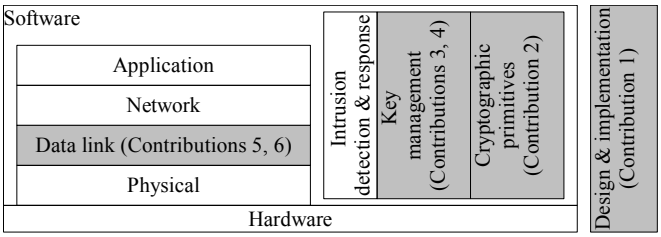


Figure 1.4: Research areas (shaded regions) that are the foci of this thesis.

Chapter 3 presents the concept of system profiles and criticial system parameters for mapping WSN systems to security architectures. This chapter is a minor revision of the paper with the same title published in IFIP SEC 2003.

Chapter 4 presents a survey and detailed benchmark of a selection of block ciphers. This chapter is to appear in the ACM Transactions on Sensor Networks.

Chapter 5 presents a scalable, energy-efficient, decentralized key management architecture, called ESA1, for WSNs. This chapter is a minor revision of the paper with the same title published in IFIP PWC 2003.

Chapter 6 presents a directed diffusion-based secure multicast scheme, called LKHW, for WSNs. This chapter has been published in IEEE ICPP 2003.

Chapter 7 presents three efffective and energy-efficient jamming algorithms against S-MAC, and provides one countermeasure to one of the attacks. This chapter is a minor revision of the paper with the same title published in IEEE EWSN 2005.

Chapter 8 presents several effective and energy-efficient jamming algorithms against three representative protocols: S-MAC, LMAC and B-MAC. Although a countermeasure is yet to be found, some recommendations are given. This chapter is a minor revision of the paper with the same title published in ACM SASN 2005.

Chapter 9 concludes the thesis and proposes future work.

CHAPTER II

State of the Art in Key Management and Link-Layer Security

A survey of the state of the art in key management and link-layer security is given in this chapter. We start by reviewing some basic concepts in cryptography in Section 2.1. The survey itself is divided into 2 sections: key management is discussed in Section 2.2, and link-layer security is discussed in Section 2.3. Finally we go through other security areas briefly in Section 2.4.

2.1 Preliminaries

This section reviews the concept of the one-way key chain. The method of the one-way key chain is believed to be pioneered by Lamport [157], initially for the secure authentication of a user's password. A typical challenge in managing user passwords is that the computer terminal might be compromised. So instead of storing a user password in plaintext, a straightforward solution is to store the hash of the password. A hash is the fixed-length result of applying a *hash function* on a variable-length plaintext. The consensus is that this hash function should be a *collision-resistant hash function* (denoted h), which is a type of hash function that satisfies the following properties [181]:

1. **preimage resistance**: given y, it is computationally infeasible to compute x such that $h(x) = y$.

2. **2nd-preimage resistance**: given x, it is computationally infeasible to compute x' such that $h(x) = h(x')$.

3. **collision resistance**: it is computationally infeasible to find *any* x, x' such that $h(x) = h(x')$ (the freedom of choice makes finding collision easier than finding preimages, so a function that satisfies this property should be more secure than a function that does not).

By definition, *one-way hash functions* are function that only satisfy the first two properties. Stinson [272] shows that the problem of finding collisions can only be reduced to the problem of finding preimages if the (1) hash function is 'close to uniform', and (2) the oracle for finding preimages has a good success probability for every possible input. The fact that none of these two requirements can be verified in practice means collision resistance is a condition in addition to preimage resistance that needs to be satisfied for a hash function to be considered secure. Therefore collision-resistant hash functions, instead of one-way hash functions, are generally assumed in situations where a secure hash function is required.

Back to our problem of storing user password, storing the hash is not enough, when the password can be intercepted when the user inputs it. Given a collision-resistant hash function h, and a user's password x, Lamport's idea is to store the hash $y_n = h^n(x)$, where n is a sufficiently large number. A user is authenticated whenever he is able to submit a

$y_i = h^i(x)$, where $i < n$, because the operation $h^{n-i}(y_i) = y_n$ can easily be performed, while the operation $h^{-i}(y_n) = y_i$ is computationally infeasible. After y_i is used, all hashes $y_{i+1}, \ldots, y_n$ are thrown away, and in the next session, the user would have to present a hash y_j ($j < i$) to be authenticated.

As sensor nodes are an analogy of compromisable terminals, this technique is widely used in WSNs, including some of the work that will be discussed later.

2.2 Key Management

Key management is the process by which cryptographic keys are generated, stored, protected, transferred, loaded, used, and destroyed. There are four principal concerns in a key management framework:

- How many keys are needed and how should the keys be distributed before the nodes are deployed? This is a problem of *key deployment/pre-distribution*.
- How do any pair of nodes, or a group of nodes establish a secure session? This is a problem of *key establishment*.
- How should a node be added to the network such that it is able to establish secure sessions with existing nodes in the network, while not being able to decipher past traffic in the network (i.e. preserving backward secrecy)? This is a problem of *member/node addition*.
- How should a node be evicted from the network such that it will not again be able to establish secure sessions with any of the existing nodes in the network, and not able to decipher future traffic in the network (i.e. preserving forward secrecy)? This is a problem of *member/node eviction*.

The following discusses some of the most important work in the literature in terms of these four concerns.

Basagni et al.'s pebblenets [24] are tailored to sensor nodes that have severe computational and storage limitations but are tamper-resistant.

- Key deployment: Pebblenets uses a typical *network-wide key deployment* scheme where every node shares the same secret key called the group identity key.
- Key establishment: All nodes in the network use the same *traffic encryption key* (TEK) to communicate securely with each other. The TEK is derived as follows in two phases. In the first phase, the network starts with no clusters at all. Then every node broadcasts its weight and when a node finds its weight to be greater than its neighbors', it announces itself as the clusterhead, while its neighbors become cluster members. The formation of the cluster is complete after the clusterhead distributes a cluster key to its cluster members. The first phase ends with the clusterheads forming a backbone of the network. In the second phase, the clusterheads that have a larger weight than its *neighboring clusterheads* elect themselves as the *potential key managers* (PKM). After an exponential backoff period, a PKM generates (using an unspecified algorithm) and distributes a TEK to other clusterheads. More than one PKM might generate a TEK at the same time, so a non-PKM clusterhead might receive more than one TEK. The clusterhead chooses the TEK generated by the PKM with the largest weight, or

the largest ID if the weights are equal. Finally each clusterhead distributes the chosen TEK to its cluster members via the cluster key. At the end of the second phase, any pair of nodes in the network can communicate securely with the TEK.

- Node addition: A new node to be added to the network is imprinted with the group identity key (T_{GI}). After the new node broadcasts a HELLO message, its neighbors will reply with the TEK encrypted with $h^n(T_{GI})$ ($n \geq 1$). Backward secrecy is partly preserved by means of changing the traffic encryption key periodically. A newly added node can only obtain new traffic encryption keys, and not past traffic encryption keys, and hence is not able to decipher past traffic.

- Node eviction: Forward secrecy is preserved by means of tamper resistance. If however tamper resistance is broken and the group identity key is compromised, forward secrecy is breached.

Perrig et al.'s Security Protocols for Sensor Networks (SPINS) [215] is a centralized architecture that assumes a tree-like network topology. At the root of the tree is a base station. Sensor nodes form the rest of the tree. SPINS has two building blocks: Secure Network Encryption Protocol (SNEP) and the micro version of the Timed Efficient, Streaming, Loss-tolerant Authentication Protocol (μTESLA).

- Key deployment: Every node shares a unique *master key* with the base station. Compromising the base station thus compromises the communication in the entire network, although compromising a node only compromises the communication between the compromised node and the base station.

- Key establishment: Two kinds of traffic are secured: node-to-node communications and broadcasts by the base station. SNEP allows two nodes to establish a session key through the base station. μTESLA allows messages broadcast by the base station to be authenticated. μTESLA is based on the one-way key chain introduced in Section 2.1. Before μTESLA begins, the base station generates a key chain $K_0, \ldots, K_n$ ($K_i = h(K_{i-1})$, $1 \leq i \leq n$) and every node synchronizes its time with the base station. To kick start μTESLA, the base station distributes the root of the key chain, K_n, to the sensor nodes. During time interval i, the base station broadcasts (1) a message M, (2) a MAC of M generated with K_i, and (3) a key $K_{i-\delta}$ that authenticates all the messages broadcast *in and before* time interval $i - \delta$. M will similarly be authenticated when the node receives K_i in interval $i + \delta$ *later*.

- Node addition: A new node is imprinted with a unique master key that it shares with the base station.

- Node eviction: The evicted node's master key is removed from the base station.

Eschenauer et al. [81] pioneer random pre-distribution schemes. The basic scheme is best studied from two perspectives, first from the perspective of random graphs, then from the perspective of combinatorics. A random graph $G(n, p)$ is a graph of n vertices (sensor nodes) for which the probability that an edge (a secure link) exists between any two vertices is p. If p is zero, then the graph (network) is disconnected, and if p is zero the graph (network) is fully connected, so there must exist a certain p such that the graph (network) is almost

certainly connected. For a graph that is connected with probability P_c, Erdös et al. [267] show that

$$p = \frac{\ln n - \ln(-\ln P_c)}{n} \tag{2.1}$$

From a combinatorial viewpoint, given a key pool of size P and a key ring size of k (i.e. number of keys in a node), the probability that any two nodes share at least one key is given by

$$p = 1 - \frac{(P-k)!^2}{P!(P-2k)!} \tag{2.2}$$

Combining Equation 2.1 and Equation 2.2, and fixing n and the intended P_c, we get a relationship between P and k. Since typically the key ring size is bounded by the sensor node's storage, k is fixed, and P is calculated accordingly. Di Pietro et al. [221] show that for a small key ring size $K \geq 2$, the network is connected with high probability as long as $P = n/\log n$. This scheme is novel in that it introduces *probabilistic key sharing* but it also leaves a lot of mechanisms unspecified. Note that the number of secure links, or degree from a graph-theoretical viewpoint, a node has is $p(n-1)$. This relation will be used later.

- Key deployment: Every node is imprinted with k keys chosen at random from the key pool of size P.
- Key establishment: Only node-to-node communications are supported. If the two nodes share at least a key, session keys are derived from the shared key(s) (but Eschenauer et al. do not specify exactly how). If the two nodes do not share any key, but have secure links to a common neighbor, then the two nodes can establish a session key through their common neighbor acting as a trusted third party (but Eschenauer et al. do not specify exactly how). Note the two nodes in this context have to be within radio range, otherwise neighbor discovery cannot take place.
- Node addition: When a node is added to the network, the node broadcasts a list of indentifiers that identify the keys it has. The neighbors reply with their lists of key identifiers. By comparing the lists, the new node and its neighbors discover what keys they share. Session keys are then derived from the shared keys.
- Node eviction: To evict a node, a controller first broadcasts a message containing a signed list of k key identifiers for the key ring to be revoked. Then the controller unicasts to each node A the signing key encrypted with key K_A, where K_A is derived from the keys the controller shares with A (but Eschenauer et al. do not specify exactly how).

Chan et al. [53] propose several improvements to Eschenauer et al.'s basic scheme.

- For key deployment, they propose q-composite random predistribution, which is the same as Eschenauer et al.'s original scheme, except in how the key pool size P is derived. As shown earlier, the expected degree (number of secure links) a node has is $p(n-1)$. If the number of expected neighbors within radio range is n', then the probability that a node is securely connected to any of its neighbors at all is $p(n-1)/n'$.

Now P is calculated such that the probability the node shares at least q keys with any of its neighbors is greater or equal to $p(n-1)/n'$. As a consequence, every connection is now secured by a combination of at least q keys.

- For key establishment, there are two options: (1) if the above q-composite scheme is used for key deployement, the session key is derived from an XOR of all shared keys; (2) if the basic scheme is used for key deployment, the session is derived using *multipath key reinforcement*. In multipath key reinforcement, two nodes that share common neighbors, can use their common neighbors to reinforce their secure links at the expense of communication efficiency. The number of such neighbors is calculated to be $0.5865[p(n-1)/n']^2 n'$.

Di Pietro et al. [220] provide further improvements to Eschenauer et al.'s basic scheme.

- Key deployment: Keys are assigned to a node according to the output of a pseudorandom generator with a public seed and the node's ID as inputs. For example, the i-th key assigned to node A is the j-th key from the key pool of size P, if $j = \text{MAC}(\text{MAC}(S, A), i) \mod P$, where $1 \leq i \leq K$, and S is a public seed that every node has a copy of.

- Key establishment: There are two options: (1) direct protocol, (2) cooperative protocol. In the direct protocol, the session key is derived from an XOR of all shared keys between the two participating nodes. This is similar to Chan et al.'s q-composite scheme. The cooperative protocol has the same purpose as Chan et al.'s multipath key reinforcement, i.e. strengthening the security of a link using common neighbors. Suppose A and B have a common neighbor C. A and B share key K_{AB}, B and C share key K_{BC}, and so on. Then the strengthened key between A and B is $K_{AB} \oplus \text{MAC}(K_{BC}, A)$. $\text{MAC}(K_{BC}, A)$ is calculated by C and sent to A encrypted with K_{AC}.

- Note addition: When a node joins the network, it discovers its neighbors' keys by just knowing its neighbors' IDs.

- Node eviction: This is the same as Eschenauer et al.'s basic scheme.

Zhu et al.'s Localized Encryption and Authentication Protocol (LEAP) [320] is a complete key management framework for *static* WSNs, that includes mechanisms for securing node-to-base station traffic, base station-to-nodes traffic, local broadcasts as well as node-to-node (pairwise) communications.

- Key deployment: Each node has to store 4 kinds of keys: (1) an individual key, (2) a group key, (3) cluster keys, and (4) pairwise shared keys. In addition to these keys: a node also has to store a one-way key chain it creates, the commitments of the key chains its neighbors create, and the commitment of the base station's key chain. If a node has d neighbors, the total number of keys the node has to store is 1 (individual key) + 1 (group key) + d + 1 (cluster keys) + d (pairwise shared keys) + d + 1 (key chain commitments) + L (keys in a key chain) = $3d + 4 + L$ keys.

- Key establishment: A node uses its individual key to encrypt messages it sends to the base station. Messages broadcast by the base station are encrypted with the group key but authenticated with μTESLA. A node's cluster key and one-way key chain allow its

neighbors to authenticate its locally broadcast messages. The combination of cluster key and one-way key chain is interesting because

  - if only the cluster key is used, a compromised neighbor would disclose the cluster key;
  - if only the key chain is used, the keys in the key chain would have to be broadcast in the clear, allowing replay attacks to take place;
  - but if used together, the cluster key can be used to hide the keys in the key chain from cluster-outsiders, so that the keys do not need to be disclosed according to a schedule as in SPINS, and the keys in the key chain can be used for authentication as usual.

Pairwise shared keys are for securing messages that require authentication of the sender.

- Node addition: When added to the network, a new node u is imprinted with an individual key for communication with the base station, the group key, the commitment of the base station's key chain, and an initial key K_I. To acquire other keys, node u first discovers its neighbors by sending out HELLO messages. The neighbors reply with their unique identifiers. The pairwise shared key between u and a neighbor v is derived as $K_{uv} = \text{MAC}(\text{MAC}(K_I, v), u)$. Node u and v can then exchange cluster keys and key chain commitments using K_{uv}. v gives u the group key. After a system-defined time-out, K_I is deleted to prevent compromise.

- Node eviction: To evict a node, the base station broadcasts an eviction message using μTESLA. Every node that receives the revocation message updates its own cluster key. When the base station broadcasts a new group key, member nodes receive the new group key via its upstream neighbors' cluster keys, but the evicted nodes would not receive the new group key since its neighbors' cluster keys have already been changed.

To wrap this section up, key management is one of the oldest areas in WSN security, a lot of work has been done and key management schemes are improving. At the beginning, we have Basagni et al.'s pebblenets that require tamper resistance, which is actually something to be avoided in WSNs. Then we have Perrig et al.'s centralized architecture, SPINS. The existence of a single point of vulnerability (i.e. the base station) is again something we want to avoid. Our proposal, ESA1, does not require tamper resistance and does not have a single point of vulnerability (Chapter 5). Probabilistic schemes are attractive because they offer the best flexibility with network topology, however supporting any form of communication that is not pairwise is awkward. On the other hand, ESA1 readily supports group communications. Zhu et al.'s LEAP provides explicit support for all predominant forms of communication typical of WSNs, but it only works for static networks. ESA1 supports mobility explicitly through its key migration protocol. The drawback of ESA1 is that it requires a clustered topology that is composed of resource-rich nodes in addition to normal sensor nodes.

2.2.1 Secure Multicast

Multicast is the sending a single message by a single sender (group controller) to a *select* group of recipients (group members). The problem of secure multicast refers to the problem of ensuring the multicast data remains confidential among the group controller and group members, under the condition of backward as well as forward secrecy. Backward secrecy

refers to the condition that new members should not be able to obtain previously used group keys, whereas forward secrecy refers to the condition that former members should not be able to obtain new group keys (these definitions are consistent with the literature and what Kim et al. [146] define as *weak backward secrecy* and *weak forward secrecy*). Historically speaking, there have been two approaches to realizing secure multicast communications, initially with the distributed approaches, then with the hierarchical approaches.

Distributed Approaches Also called *conference key distribution schemes*, the approaches in this category try to solve the *key agreement problem*, i.e. the problem of deriving a secure common key among n users. Ingermasson et al. [118] are the first to extend the Diffie-Hellman (DH) problem to groups. Burmester and Desmedt [44] correct Ingermasson et al.'s security flaw by using cyclic instead of symmetric functions. Just and Vaudenay [131] patch the key authentication flaw of the Burmester-Desmedt scheme and generalize it. Steiner et al. [270, 271] begin to consider *dynamic peer groups* instead of a fixed number n of users, and introduce CLIQUES, a suite of *contributory* key agreement protocols which requires less communication than the Burmester-Desmedt scheme does. Ateniese et al. [21] imbue CLIQUES with authentication properties. The problem with these conference key distribution schemes is that they require a lot of exponentiation computations that are prohibitively expensive for sensors [50, 110], and re-keying is inefficient. Blundo et al. [35] did the pioneering investigative work on the storage requirements of k-secure t-conference key distribution scheme. The proposed scheme is information-theoretically secure, and does not require exponentiation, but requires $O(n^t)$ amount of keying material per node (where n is the total number of users), which is impractically large. The conclusion is that both DH-based and information-theoretically secure schemes have their share of scalability problems.

Hierarchical Approaches By imposing a hierarchial structure – binary tree being the mainstream – on dynamic peer groups, hierarchical approaches have been able to achieve better scalability than the distributed approaches. There is the pioneering work by Wallner et al. [292] who propose the logical key hierarchy (LKH) model. The LKH model is scalable because the total number of keys in the system is linearly proportional to the number of users, and both the number of keys per user and the number of messages required to manage group dynamics, are logarithmic in the number of users. Wong et al. [299] investigate and compare three re-keying paradigms, i.e. key-oriented, user-oriented and group-oriented re-keying. The re-keying paradigm of LKHW, our proposal, is an improved form of group-oriented re-keying. McGrew et al. [178] introduce the one-way function tree (OFT), where the KDC needs only dispatch $\lceil \lg_a n \rceil$ keys (a is the degree of the tree, n is the number of group members) during re-keying, the same number as LKHW's, instead of the $2\lceil \lg_a n \rceil$ required by the basic LKH model. Canetti et al. [48] replace McGrew et al.'s non-standard cyptographic primitive with pseudorandom generator. The EHBT scheme proposed by Rafaeli et al. [236] uses a non-standard cryptographic primitive similar to OFT's, i.e. one-way function of the form $h(\text{key} \oplus \text{index})$ for key refreshment. In all the schemes mentioned so far, affected group members need to receive refreshed keys from the KDC during user-join events. Members in Perrig et al.'s ELK [213] however avoid that overhead by locally *and* periodically re-generating their keys. Di Pietro et al. [218] provide further improvement by exploiting pseudorandom function and the "level-awareness" of a node in a scheme called LKH++.

LKHW requires the same number of keys for the KDC, i.e. $2n - 1$; and the same number of keys for a member, i.e. $\lceil \lg_a n \rceil + 1$, as OFT, EHBT, ELK and LKH++ do. LKHW's handling of user-join events might seem inefficient since a seed is flooded across the network, but due

to the multihop nature of WSN (where every node behaves as a router) *and* the data caching property of directed diffusion, this method is in fact not only efficient but also robust.

Except for LKHW, the above approaches do not take into account the multihop nature of the network. It is only recently that researchers have started adapting the well-established hierarchical approaches to WSNs. Lazos et al. [159] show that placing the group members on the leaves of the key tree according to their physical location can lead to substantial energy savings, but the scheme is not a complete secure multicast framework because it does not specify how the join and leave events are handled. It can be used however *in conjunction with* LKHW provided the sensor nodes (1) have low mobility and (2) have a means of determining their own locations with high accuracy. Kaya et al. [140] adopt the approach of the Iolus framework [188] by distributing the responsibility of the group controller to group members. This approach reduces the overhead of joins compared to LKHW at the expense of higher resource consumption at the group members. In fact, Kaya et al.'s scheme is targeted at mobile ad hoc networks (MANETs) instead of WSNs, where the group members are typically more resource-rich.

2.3 Link-Layer Security

There is no official definition of link-layer security, but by extending the definition in the IEEE 802.15.4 standard [116] and the definitions by Karlof et al. [138] and Sastry et al. [251], we define link-layer security as the mechanisms for achieving the following objectives:

Objective 1: protecting the *availability* of services against link-layer jamming.

Objective 2: ensuring the *confidentiality*, *integrity*, *authenticity* and *freshness* of messages between neighboring nodes (note: the data link layer itself only concerns neighboring nodes).

Objective 3: prevent *selfish* nodes (nodes that cheat to get their messages delivered with the highest priority possible, and as much use of the communication medium as they want, regardless of the needs of their neighbors) from starving the rest of the network (note: selfishness itself is not limited to the data link layer).

We focus on Objective 1 in this thesis, because Objective 2 can be considered solved using a standard library such as TinySec [138], and Objective 3 is targeted at selfish *insiders* whose aim is on gaining an unfair advantage on the use of the medium, rather than bringing the entire system down (the complete argument has already been presented in Section 1.3.1). In contrast, Objective 1 is a more immediate DoS threat by *outsiders*. Nevertheless, selfishness will be touched on, before jamming is discussed in detail.

2.3.1 Selfishness

In contention-based MAC protocols, a node has to *contend* for the medium for a random period of time, i.e. check if there is any signal in the air, before sending its packet to avoid collision. The contention time, also called random backoff time, is randomly picked from an interval ranging from 0 to a value called the contention window. If the channel is indeed clear, the node that picks the shortest contention time always wins the channel. A node that always pick a near-zero contention time is called a selfish node.

Konorski [152] proposes a scheduling policy that guarantees cooperative parties a high bandwidth share even in the presence of non-cooperative parties that behave rationally and

that seek a Nash equilibrium (a steady state where no player can do better in the game by deviating from it [209]). However his proposal is only applicable to *single-hop* networks, and hence is not applicable to WSNs. Kyasanur et al. [156] propose that instead of letting the sender set the contention time, the receiver sets and sends the time in the CTS and ACK packets to the sender. The sender uses this assigned contention time in the subsequent transmission to the receiver. The receiver can then tell if the sender is being selfish, i.e. using a value smaller than the assigned value, by observing the number of idle slots between consecutive transmissions from the sender. The problem with this approach is that if the receiver misbehaves, the sender is penalized. Cárdenas et al. [49] propose using Blum's coin flipping protocol [34] to ensure that the sender and the receiver choose a random backoff time, such that if *either* the sender *or* the receiver deviates from the random backoff, the other party would know. Čagalj et al. [290] investigate the effect of a group of selfish cheaters from a game-theoretic viewpoint. This thesis focuses on DoS attacks, in which the purpose of the misbehaving party lies in disruption instead of getting more throughput.

2.3.2 Jamming

Link-layer jamming attacks are only feasible if the jammer can somehow get on the same channel used by the victims to communicate. Acquiring the channel and subsequently intercepting messages on the channel are all part of a larger class of activities called *electronic countermeasures* (ECM) [216]. This section begins by looking into the question of feasibility of ECM on WSNs. The rationale of this investigation is that if an attacker can successfully acquire the channel the victims are using, with or without ECM, the attacker can successfully launch link-layer jamming attacks.

The study of ECM has already begun more than 50 years ago and is thus an established field [216, 217]. Well-known countermeasures to ECM, or *electronic counter-countermeasures* (ECCM), include spread spectrum modulations, adaptive antenna systems, error correcting codes and cryptography. We concentrate on spread spectrum techniques because there is not much room to maneuver in the other areas: sensor nodes typically use an omnidirectional antenna, Reed-Solomon codes [164, 203, 252] and standard cryptographic primitives [2]. Among the most well-known spread spectrum techniques are direct-sequence spread spectrum (DSSS) and frequency-hopping spread spectrum (FHSS). Most WSN transceivers support only FHSS if they support spread spectrum at all. The reason is that DSSS requires more circuitry (higher cost) to implement, is more energy-consuming and more sensitive to environmental effects [183, 229]; on the other hand, the hop rate in a FHSS system is typically much lower than the chip rate in a DSSS system, resulting in lower energy usage [183, 283]. Note: hop rate is the number of frequency hops per unit time; chip rate is the number of chips per unit time, where one original signal bit is represented by a fixed number of pseudorandom bits called chips.

There are two types of FHSS systems: (1) slow frequency hopping (SFH) systems that transmit more than one symbol per hop, and (2) fast frequency hopping (FFH) systems that spreads a symbol across multiple hops. SFH systems are more popular because FFH requires highly sophisticated synthesizers [9]. SFH systems commonly use a hop rate between 50 and several hundred hops per second [283], e.g. combat net radios use a hop rate between 50 to 500 hops/s, as hop rates above 1000 hops/s tend to cause internal interference [252]. It is therefore reasonable to expect WSN transceivers to use a hop rate of 50–1000 hops/s, if they support FHSS at all. For FHSS systems moreover, quaternary/binary frequency-shift keying (FSK) is the data modulation scheme of choice [183, 227].

Table 2.1: Comparison of spread spectrum support.

Spread spectrum	Transceiver
DSSS	Chipcon CC2400 [58]
FHSS	Chipcon CC1000 [57], Conexant Systems RDSSS9M [229], MaxStream 9XStream [176], Zeevo Bluetooth [147]
No	Infineon TDA 5250 [117], National Semiconductor LMX3162 [197], RFM TR1000 [242], RFM TR1001 [243]

Table 2.1 compares the spread spectrum support of some common WSN transceivers (which are also previously listed in Table 1.1). The table shows that only one relatively high-end chip supports DSSS, and among the rest, half of the transceivers support FHSS and the other half do not support spread spectrum at all. Our task remains to determine how hard/easy it is to intercept the signals of these frequency-hopping WSN transceivers.

We first look at how frequency-hopping signals can be intercepted, then we shall see what conventional countermeasures there are against interception and if the frequency-hopping WSN transceivers in Table 2.1 readily support these countermeasures. Interception begins by the attacker's estimating the hopping frequencies. In most FHSS systems, the transmitter and receiver hop from frequency to frequency in synchrony according to a frequency table indexed with a pre-distributed pseudorandom seed [283]. The simplest way for an attacker to find out the hopping frequencies is thus to find out the table and the corresponding pseudorandom seed. When this is not possible, the attacker would use an RF receiver (e.g. a scanning superheterodyne receiver on SFH systems [224]) to detect communication signals. The primary function of the receiver is to (1) scan each frequency across the target bandwidth for the presence of signals, and (2) apply a combination of filter(s) (e.g. a bandpass filter in the case of a scanning superheterodyne receiver [281]) and envelope detector(s) to the signal to estimate the carrier frequency. Intuitively, the attacker's difficulty in estimating these frequencies can be compounded using a higher hop rate, and according to Torrieri's analysis [282], an acceptable hop rate is at least 500 hops/s. Torrieri's result is dated 1989 and according to Schleher [252] in 1999, complete protection against a sophisticated interceptor requires a hop rate of the order of 10000 hops/s. In conclusion, frequency-hopping WSN transceivers are in principle (1) interceptable if their hop rate is below 500 hops/s; (2) resistant to most interceptors if their hop rate is 500-1000 hops/s. Since their hop rate hardly exceeds 1000 hops/s, the conclusion is all frequency-hopping WSN transceivers are vulnerable to the most sophisticated interceptors.

So far, we have discussed the *feasibility* of link-layer jamming, in the following we will investigate the state of the art in the *execution* of link-layer jamming. To start with, a number of authors [214, 261, 273, 300] provide a general overview of potential DoS attacks on the data link layer, e.g. intentional collision of messages (jamming), sending RTS packets to request for the channel for an arbitrary amount of time etc.

Negi et al. [200] investigate the throughput of a CSMA/CA system in which the arrivals of data packets are modeled as a Poisson process, under the attack of a probabilistic jammer. The result is intuitive enough: given a fixed number of jammed slots, the longer the data packets, the lower the throughput.

Xu et al. [304] propose two evasion strategies against constant jammers: (1) channel surfing and (2) spatial retreat. Channel surfing is essentially an adaptive form of frequency

hopping. Instead of continuously hopping from frequency to frequency, a node only switches to a different frequency when it discovers the current frequency is being jammed (using an unspecified mechanism). Spatial retreat is an algorithm according to which two nodes move in Manhattan distances to escape from a jammed region. In other words, the algorithm requires the nodes to be mobile. Overall, the algorithms are effective against constant jammers but we are motivated to look at jammers who are more energy-efficient than constant jammers.

In another work, Xu et al. [303] introduce 4 generic jammer models, namely (1) the constant jammer, (2) the deceptive jammer, (3) the random jammer and (4) the reactive jammer. A constant jammer emits a constant noise; a deceptive jammer either fabricates or replays valid signals on the channel incessantly; a random jammer sleeps for a random time and jams for a random time; and lastly, a reactive jammer listens for activity on the channel, and in case of activity, immediately sends out a random signal to collide with the existing signal on the channel. Based on the models, Xu et al. show that either received signal strength indication (RSSI) or carrier sensing time alone is not sufficient in detecting all the 4 types of jammers. Instead, attacks can be detected by measuring (1) both the packet delivery ratio and the signal strength, or (2) both the packet delivery ratio and the location. In our work [4, 5], we look at potential attacks on MAC protocols. Instead of looking at the packet delivery ratio of individual nodes, we look at the effect of distributed jammer motes on the WSN as a whole. In our opinion, apart from jamming effectiveness, a jammer cares about jamming efficiency. For this reason, while Xu et al. provide metrics for measuring the quality of service of individual nodes, we provide metrics for measuring the jamming effectiveness and efficiency of the jammers. Xu et al.'s and our work are complementary in the sense that Xu et al. target jammers at the physical layer, while we target jammers that aim to gain more edge by exploiting the data link layer.

Unlike key management, link-layer jamming is a relatively new field, so not much work has been done in this area. This further highlights the novelty of our work.

2.4 Other Security Areas

While key management and link-layer security are the foci of this thesis, other areas have their influences on our work. This section discusses DoS attacks on the network layer and data protection on the application layer.

2.4.1 The Network Layer

DoS attacks can be realized by disrupting the routing fabric. A disrupted routing fabric also causes energy to be wasted on erratic routing. There are 2 types of routing protocols for WSNs: (1) ID-based protocols, in which packets are routed to the destination designated by the ID specified in the packets themselves; and (2) attribute-based protocols [92], in which packets contain attributes that specify what kinds of data are being requested or provided. These protocols are however inherently insecure due to their assumption that every node is trusted. The following documents the most important 'rescue efforts' in roughly chronological order.

ID-based protocols

A lot of ID-based protocols have been proposed [84], the most well-known being the Ad hoc On-demand Distance Vector (AODV) protocol [212] and the Dynamic Source Routing (DSR) protocol [125].

Marti et al. [171] pioneer the idea of *watchdog* and *pathrater*. In their scheme, every node implements (1) a watchdog that, operating in *promiscuous mode* (which consumes a great amount of energy), constantly monitors the packet forwarding activities of its neighbors, and (2) a pathrater that rates the transmission reliability of all alternative routes to a particular destination node according to the reports of the watchdog. Although proposed as a general mechanism for fortifying any general routing protocol, it is essentially only practical for source routing protocols. Collusion between malicious nodes remains an unsolved problem.

Buttyán et al. [46] and Blažević et al. [32] conceptualize the motivation for nodes not to be selfish as *nuglets*, a sort of virtual currency that nodes have to pay to their neighbors to get their packets forwarded. To guard a node's nuglets against illegal manipulation, a tamper-resistant security module [13] storing all the relevant IDs, nuglet counter and cryptographic materials (but not the code) is compulsory. The cross-certification architecture calls for public-key cryptography, which exerts a high demand on computing resources. The amount of overhead is also a concern.

Yi et al. [312] propose levels of protection as a negotiable metric in route discovery, akin to the way IPsec works [144]. This scheme together with Zapata et al.'s SAODV (for securing AODV) [315] and Sanzgiri et al.'s ARAN (for securing AODV and DSR) [250] require energy-consuming public-key cryptography.

In Zhou et al.'s proposal [319], every message is signed to allow it to be authenticated. The verification process depends on the key management service that is distributed over $t + 1$ servers among n nodes, where $n \geq 3t + 1$, so that at most t may be compromised (i.e. a $(t + 1, n)$-threshold scheme [75]). A key management server not only has to store its own key pair, but also the public keys of all the nodes in the network. The difficulty includes (1) the storage requirement exerted on the servers which must potentially be specialized nodes in the network, and (2) the overhead in signing and verifying routing message both in terms of computation and of communication.

Hu et al.'s Ariadne [112] endows DSR with the following security properties: (1) the target node of a ROUTE REQUEST message can authenticate the initiator of the ROUTE REQUEST message; (2) the target node can be sure that no node ID has been illegitimately removed from the node list in the ROUTE REQUEST; and (3) the initiator can be sure that no bogus node ID has been added to the node list of the corresponding ROUTE REPLY. The downside of the scheme is that it essentially requires a node to share a key with every other node in the network.

Attribute-based protocols

Directed diffusion [93, 119, 120] is a data-centric, attribute-based protocol and is described in detail in Chapter 6. Due to the robust nature of flooding, attacking directed diffusion requires more effort, but is still achievable. If authentication is not used, an outside attacker can easily spoof negative reinforcements to block the flow of information from the sources to the sink. At the same time, the attacker can just as easily spoof positive reinforcements to attract the information flow to itself instead. While Chapter 6 addresses the information security aspect of directed diffusion, operational security is still largely an open issue.

Attacks of other protocols like rumor routing [42] which is derived from directed diffusion, and geographic routing like GPSR [139] and GAF [305] are discussed in detail by Karlof et al [137]. So far there is no integrated approach for countering routing attacks on these protocols.

2.4.2 The Application Layer

The application layer is mostly about managing data. However given that sensor nodes are generally not tamper-resistant, cryptography alone is no longer sufficient for ensuring the trustworthiness of a piece of data in a WSN, there is a natural need for detecting and isolating false information propagated by malicious insiders. Although there are not many proposals in this area, some novel techniques have been introduced.

In Golle et al.'s proposal [98], a node always tries to search for the most plausible explanation for the data it has collected, based on the assumption that malicious nodes may be present. The most plausible explanation for a set of data is the explanation that (1) is consistent with the model of the system, and (2) has the fewest malicious nodes.

Data aggregation is an essential feature of WSNs, but false data produced by malicious nodes can drastically affect the aggregation result (the result of aggregating data, using the aggregation function). Wagner [291] looks at just this problem: which aggregation functions can be secure and meaningfully computed in the presence of a few malicious nodes? For aggregation functions such as minimum, maximum, sum and average, intuition alone tells us they are insecure, e.g. for minimum, a malicious node can claim any value arbitrarily smaller than the real minimum. For other aggregation functions, using the theory of parameter estimation, Wagner shows formally that median, count and t%-trimmed average are more resilient to attacks.

Another approach is to detect if the aggregator is cheating, or more precisely, to guarantee that if the home server accepts an aggregation result from the aggregator, the reported result is close to the true aggregation value with high probability. For this, Przydatek et al. [232] propose a transaction paradigm called *aggregate-commit-prove*, which in effect provides two layers of defense against data corruption. The first defense is commitment (hence the word 'commit' in aggregate-commit-prove): the aggregator commits to the aggregated data, by cryptographic means. The second defense is interactive proofs (hence the word 'prove'): the aggregator proves to the base station the validity of the aggregation result, by statistical means.

2.5 Summary

WSNs pose new technological challenges because they are unlike conventional networks, and as new uses are found, more inventions are needed in the area of security. Historically speaking, research in WSN security started with key management, since it is the basic requirement for protecting information in the network. Secure routing is another area that attracted a lot of attention in the early stage, but a lot of these techniques are borrowed from their counterparts in MANETs and hence are not directly applicable to WSNs, because a WSN node has typically less resources than a MANET node. As WSN network protocols continue to fluorish and materialize into concrete implementations, attention gradually turn to other layers of the protocol stack, notably the data link layer. An attacker that focuses on jamming the data link layer needs only implement the physical layer and the data link layer – the effort is minimal while the reward is high. All in all, the survey reinforces our idea to focus on key management and link-layer security.

CHAPTER III *

Assessing Security-Critical Energy-Efficient Sensor Networks

Abstract We describe why current research in ad hoc networks requires an effective assessment framework, and how our *system profile* proposal can be used for the purpose. With this tool at hand, we have managed to carve out manageable design spaces from the seemingly infinite possibilities of ad hoc mobile wireless networks.

3.1 Introduction

Wireless sensor networks combine the characteristic of ad hoc mobile wireless networks (ad hoc networks in short) at the system level, with the characteristics of sensors at the component level. Ad hoc networks are: (1) **ad hoc**: the network set-up is possibly short-lived; (2) **mobile**: the nodes are not attached to any fixed communications infrastructure as well as fixed energy supply; (3) **wireless**: the nodes communicate wirelessly. These three properties imply a series of constraints, that together with the constraints imposed by sensors (among which energy being the predominant), form the starting point of our security research.

In our opinion, the current research landscape lacks structure. Here, we show how prevalent (proposed) architectures are based on assumptions that are not only implicit but even conflicting at times. As an example of implicit assumptions, to curb selfishness, the Terminodes project introduces a virtual currency called *nuglets* [46]. To protect these nuglets from tampering, every node must be equipped with a tamper-resistant *security module*. At the same time, the proposed cross-certification architecture calls for public-key cryptography. In addition, as part of the Terminodes project, Hubaux *et al.* propose that each node maintains its own *certificate repository* [114]. These repositories store the public certificates the node themselves issue, and a selected set of certificates issued by the others. It is thus concluded, although the Terminodes project does not specify, that the implicit assumptions or requirements of Terminodes are that they have to be (1) tamper-resistant, (2) capable of performing resource-intensive public-key cryptography, and (3) have sufficient storage for a potentially large certificate repository.

In direct contrast to the above approaches are Basagni *et al.*'s *pebblenets* [24] and Perrig *et al.*'s SPINS [215] which specifically avoid public-key cryptography that sensors are ill-equipped to handle. The different assumptions in public-key cryptography present an insurmountable barrier in unifying the earlier proposals with Basagni *et al.*'s or Perrig *et al.*'s. Of further interest is that Basagni *et al.*'s *pebbles* are required to be tamper-resistant to ensure forward secrecy. Building tamper-resistance into a sensor can substantially increase the cost of the sensor depending on the intended FIPS 140-1 level (`csrc.nist.gov`). This presents a case of potentially self-contradicting requirements within a single proposal.

*This chapter is a minor revision of the paper with the same title published in the 18th IFIP TC11 International Conference on Information Security, Security and Privacy in the Age of Uncertainty (SEC 2003), pages 459–463, Kluwer Academic Publishers, ISBN 1-4020-7449-2 [3].

Our contribution to providing structure is *system profiles*. System profiles are an effective means for assessing and categorizing s system according to its actual specification and requirements. Under this framework, we make it possible for architectural designs to relate to a set of fine-grained properties they should comply with, instead of a hypothetical and often arbitrary set of assumptions. Keeping one system profile in perspective at a time helps to prevent oversight and underestimation.

3.2 System Profiles

The inspiration of system profiles comes from Sun's Java™ 2 Platform, Micro Edition (J2ME) (`java.sun.com/j2me`), a Java platform targeted at low-end devices such as PDAs, cellphones etc. As the capabilities of these devices vary tremendously, Sun introduced the concept of *configurations*. Configurations map to Sun's two primary design targets: devices that can be held in the hand and devices that can be plugged into a wall. Hence there are two configurations: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). CLDC targets resource-constrained devices with typically a 16/32-bit processor, and 128~512 KB of memory available for the Java platform and applications; whereas CDC is for more powerful devices. The strategy of J2ME attests to the fact that one size does not fit all. We are adopting a similar profiling strategy. The only difference is what we are profiling are not the nodes themselves, but the systems.

We categorize different ad hoc networks into different system profiles, each of which is defined by a set of boolean *critical system parameters*. (The boolean nature of the papameters is really an abstraction of a more refined scale that we would like to investigate in the future.) The following critical system parameters have been defined:

1. **Message Confidentiality (MC)** specifies the requirement for encrypting all network messages. Rationale: Although encrypting network messages is useful for battling traffic analysis by attackers, not all types of system require this level of security. A brute force approach of encrypting all messages regardless of necessity does not necessarily provide the highest level of security, nor is it energy-efficient. In the case where MC=F(alse), the confidentiality of the payload data is considered an application-dependent (as contrasted to system) requirement .

2. **Tamper Resistance (TR)** speficies whether there is an allowance for using tamper-resistant hardware to protect every node in the network. Rationale: To entrust a node with a key, we have to make sure the node itself does not divulge the key to unauthorized parties upon tampering. If not all nodes in the network can be made tamper-resistant, it is insufficient to rely on cryptography alone to ensure the integrity of any node, since any node can be tampered, with its keys compromised and its program modified. For such networks, cryptographic material cannot be kept at any node for any extended period, and supplementary means are necessary. It is for simplicity that this parameter is represented as a binary scale instead of a multi-level scale as in FIPS 140-1.

3. **Public-Key Cryptographic Capability (PKCC)** refers to the capability of any node in the network to perform public-key cryptography. Rationale: This parameter determines whether public-key cryptographic technology can be employed. Note however that the fact that PKCC is true does not guarantee that public-key cryptography can

or should be used extensively. It only indicates that the technology *can* be used, and the degree of usage depends on the architectural design. In general, to perform public-key cryptography, the processor speed is not the only issue, the deciding factor is the sufficiency of memory, therefore PKCC more or less translates to the sufficiency of memory.

4. **Rich Uncles (RU)** refers to the availability of Rich Uncle nodes, which are resourceful nodes, both in terms of computing resources and energy, that are suitable for the role of certification authorities, for example in the Rich Uncle Protocols [50]. These nodes might be floating or might be gateways to some external wired networks. If all nodes are equally "rich" (i.e. the network is homogeneous), we may assume that either every node is a Rich Uncle or no node is a Rich Uncle. Rationale: the existence of Rich Uncles confirms the possibilities of relegating resouce-intensive tasks and assigning important security roles to them, thereby facilitating the use of certain hierarchical architectures and possibly public-key cryptography.

Message authentication is not included as a critical system parameter because it is considered an essential component of a *secure* system. A system that does not support message authentication is susceptible to denial-of-service attacks because the system can never tell whether a request is genuine. Such a system also cannot attach any confidence to the data it collects, because absolutely no data is authenticated. This selection of parameters is not meant to be exhaustive or definitive and yet we find it an unambiguous way for categorizing the types of system we know so far. We give a few examples below of how we classify some typical ad hoc network systems:

- **Battlefield interpersonal communication** is a scenario where telecommunication devices, carried by vehicles and soldiers, communicate in an ad hoc fashion without the need as well as the danger of using a base station (whose compromise would jeopardize the entire mission). The requirement for MC and TR is obvious. The assumption for PKCC can also be justified, while RU may not be as readily assumed. Hence, MC=T, TR=T, PKCC=T, RU=F.

- **Battlefield sensor surveillance** is the class of systems in which minute wireless sensors (e.g. chemical sensors, seismic sensors etc.) are dispatched in military zones for critical surveillance. Signals Intelligence data are meant to be gathered from Unmanned Aerial Vehicles (UAVs) and relayed to the forward operating base for analysis and correlation. Because of the low cost and disposable design of sensor nodes, TR, PKCC and RU cannot be assumed. MC is undoubtably critical. Hence, MC=T, TR=F, PKCC=F, RU=F.

- **Spontaneous networking** is, as described by Feeney *et al.* [85], a technology that allows people to meet and use their laptops, PDAs, tablets etc. to start collaborating on some tasks through wireless networking, i.e. in the absence of a fixed infrastructure. For the same reason why IPsec is invented, MC is important. TR, as applied to consumer hardware, cannot currently be assumed. PKCC is generally available although the performance varies widely across the classes of device. By the psychological reasoning that nobody wants to spend more energy than the others, we can assume that nobody wants to be a RU. Hence, MC=T, TR=F, PKCC=T, RU=F.

- **SPINS-type sensor networks** refer to the class of networks with sensors deployed around a central base station. The presence of the base station immediately guarantees the existence of a RU. Hence, MC=T, TR=F, PKCC=F, RU=T.

- **EYES networks** is the class of networks we design for the EYES project (`eyes.eu.org`). We envision our prototype to be a congregation of sensor networks for intelligent buildings. The sensors are meant to collaborate to achieve some desired functions, but often in the course of performing such functions, privacy and security-sensitive data are transmitted, so MC is to be enforced. For economic reasons, TR is not assumed. We expect our node to have PKCC because each sensor will carry a 1 MB serial RAM, which is large enough for the purpose. We also do not want to rule out the possibilities of RU, since the office environment is largely under our control. Note that even with these assumptions in place, we are not restricting ourselves to any specific architecture, centralized or decentralized. What we have in mind of the topology is a sea of sensors, some of which mobile and some static, peppered by tiny islands of relatively resource-rich devices. Hence, MC=T, TR=F, PKCC=T, RU=T.

To wrap up, system profile is a means of classification and assessment, it does not dictate what architecture should be adopted for which particular profile.

3.3 Conclusion

Realizing that one size does not fit all, we have introduced a unified assessment framework based on the notion of system profiles, not only to remind ourselves of the valid set of assumptions and requirements, but also to allow ourselves to concentrate on one profile at a time. Our research exercise has testified it to be a useful tool in assessing ad hoc networks.

CHAPTER IV *

Survey and Benchmark of Block Ciphers for Wireless Sensor Networks

Abstract Cryptographic algorithms play an important role in the security architecture of wireless sensor networks (WSNs). Choosing the most storage- and energy-efficient block cipher is essential, due to the facts that these networks are meant to operate without human intervention for a long period of time with little energy supply, and that available storage is scarce on these sensor nodes. However, to our knowledge, no systematic work has been done in this area so far. We construct an evaluation framework in which we first identify the candidates of block ciphers suitable for WSNs, based on existing literature and authoritative recommendations. For evaluating and assessing these candidates, we not only consider the security properties but also the storage- and energy-efficiency of the candidates. Finally, based on the evaluation results, we select the most suitable ciphers for WSNs, namely Skipjack, MISTY1 and Rijndael depending on the combination of available memory and required security (energy efficiency being implicit). In terms of operation mode, we recommend Output Feedback Mode for pairwise links but Cipher Block Chaining for group communications.

4.1 Introduction

A wireless sensor network (WSN) is a network composed of a large number of sensors that (1) are physically small, (2) communicate wirelessly among each other, and (3) are deployed without prior knowledge of the network topology. Due to the limitation of their physical size, the sensors tend to have storage space, energy supply and communication bandwidth so limited that every possible means of reducing the usage of these resources is aggressively sought. For example, a sensor typically has 8~120 KB of code memory and 512~4096 bytes of data memory. The energy supply of a sensor is such that it will be depleted in less than 3 days if operated constantly in active mode [317]. The transmission bandwidth ranges from 10 kbps to 115 kbps. Table 4.1 compares the sensor node used in the EYES project [286] with Smart Dust [110] and the Intel Research mote [147].

Table 4.1: Comparison of the EYES node with Smart Dust and the Intel Research mote.

	Smart Dust	EYES node	Intel mote
CPU	8-bit, 4 MHz	16-bit, 8 MHz	16-bit, 12 MHz
Flash memory	8 KB	60 KB	512 KB
RAM	512 B	2 KB	64 KB
Frequency	916 MHz	868.35 MHz	900 MHz
Bandwidth	10 kbps	115.2 kbps	100 kbps

Karlof et al. [137] made an interesting observation that WSNs will more likely ride Moore's Law *downward*, that is, instead of relying on the computing power to double every 18 months, we are bound to seek ever cheaper solutions. However looking at the current

*This chapter is to appear in the ACM Transactions on Sensor Networks [2].

development of WSNs (Table 4.1), computing power is indeed increasing, though not necessarily at the rate predicted by Moore's Law. Either way, we are conservative and assume that the hardware constraints of WSNs will remain constant for some time to come.

Cryptographic algorithms are an essential part of the security architecture of WSNs. Using the most efficient and sufficiently secure algorithm is thus an effective means of conserving resources. By 'efficient' in this paper we mean requiring little storage and consuming little energy. Although transmission consumes more energy than computation, our focus in this paper is on computation and we can only take transmission energy into account when considering the security scheme as a whole. The essential cryptographic primitives for WSNs are block ciphers, message authentication codes (MACs) and hash functions. Among these primitives, we are only concerned with block ciphers, because MACs can be constructed from block ciphers [230], and hash functions are relatively cheap. Meanwhile, public-key algorithms are well-known to be prohibitively expensive [50].

Our selection of block ciphers is Skipjack [202], RC5 [245], RC6 [246], Rijndael [73], Twofish [258], MISTY1 [172], KASUMI [7] and Camellia [17]. Although Rijndael has been selected by the American National Institute of Standards and Technology (NIST) as the Advanced Encryption Standard (AES) after a five-year long standardisation process that included extensive benchmarking on a variety of platforms ranging from smart cards [102] to high end parallel machines [301], the selection of Rijndael for our platform is *not* obvious. This is because the fact that Rijndael is *on average* the best performer on a range of standard platforms, does not mean that it also performs best on our platform, which is Texas Instruments' 16-bit RISC-based MSP430F149 [279], chosen for its ultra-low power consumption. This microcontroller has 60 KB of Flash memory, 2 KB of RAM, 16 registers, an instruction set of 51 instructions and 5 power-saving modes, and is typically used in sensor systems. The fact that it is not commonly used in smart cards or any mainstream 32/64-bit computing platform, which are the platforms the AES committee mainly focused on, further suggests the necessity of our study. Although with the advent of IEEE 802.15.4 or ZigBee [116], hardware implementations of AES are expected to appear, there would still be customised sensor nodes that are not equipped with such hardware for cost and other reasons (e.g., our nodes), and software implementations offer more flexibility (e.g., in software updates), our benchmark results are therefore still applicable.

The contribution of this paper is three-fold: (1) to identify the candidates of block ciphers suitable for WSNs based on existing literature; (2) to provide a systematic methodology for assessing the suitability of the ciphers, in terms of both security and efficiency; and (3) to select the suitable ciphers for WSNs based on the evaluation results.

Our evaluation framework consists of (1) literature survey, and (2) benchmarking. The rest of this paper is organised as follows. Section 4.2 contains the literature survey of the block ciphers, explaining why they are selected and why others are not, as well as shedding light on their security properties. Section 4.3 details aspects of our benchmarking. Section 4.4 provides our observation and evaluation results. Section 4.5 concludes.

4.2 Literature Survey

A typical cipher consists of three components: (1) the encryption algorithm, (2) the decryption algorithm and (3) the key expansion algorithm (also known as key scheduling or key setup). The key expansion algorithm expands the *user key* or *cipher key* to a larger intermediate key, to allow (ideally) all bits of the cipher key to influence every round of the encryption

algorithm. For most ciphers, key expansion needs only be done once to cater for both encryption and decryption; for other ciphers however, key expansion has to be done separately for encryption and decryption (e.g. for Rijndael). The most important parameters of a block cipher are (1) the *key length*(s) it supports, (2) the *block size* and (3) the *nominal number of rounds*. In the ensuing discussion, for each cipher we give the reasons why we have chosen to evaluate the cipher, as well as provide status quo information on the security strength of the cipher. Table 4.2 explains some of the terms we use in the following discussion. Note that in Table 4.2 we adopt Lenstra et al.'s [161] definition of *security margin*. The definition can be understood as follows. Suppose (1) c_{DES} is the number of computations required to break DES, (2) c_X is the number of computations required to break cipher X, and (3) an attacker that can afford c_{DES} computations starting from 1982 can afford c_X computations starting from year y, then the security of cipher X in year y is computationally equivalent to the security of DES in 1982, or in other words, the security margin of cipher X is y. The year 1982 is chosen as the baseline because DES was standardised in 1977 and set for review in 1982. If the best known attack against a cipher with key length k is exhaustive key search, y can be calculated according to Equation 4.1 [161]:

$$y = 1982 + \frac{30}{23}(k - 56) \tag{4.1}$$

Hence in the discussion below, when a cipher with 80-bit keys has a security margin of 2013, or when a cipher with 128-bit keys has a security margin of 2076, we consider the cipher to be "secure".

4.2.1 Skipjack

We have chosen to evaluate Skipjack because it is used in TinySec [138], SenSec [163] and TinyKeyMan [165]. As TinySec is an optional part of TinyOS, the de facto operating system for WSNs, to the user community of TinyOS, justifying the use of Skipjack seems like a favour waiting to be fulfilled.

Security

Skipjack is a 64-bit block cipher with an 80-bit key. The fact that it is declassified by the NSA raises natural suspicions over its security. However since its declassification in 1998, Skipjack has resisted years of cryptanalysis. The best attack on Skipjack with full 32 rounds is still exhaustive key search. With reduced rounds, the best known attack is Biham et al.'s cryptanalysis with impossible differentials that breaks 31-round Skipjack slightly faster than exhaustive search using 2^{34} chosen plaintexts, 2^{64} memory and 2^{78} encryptions [27]. Reichardt et al. [239] observe that as the nonlinear permutation of Skipjack affects only a quarter of the bits at each round, it is relatively easy to follow differentials across multiple rounds. However, they also show that there are no meaningful truncated differentials for Skipjack with full 32 rounds, providing heuristic evidence that Skipjack *may be* secure against truncated differential distinguishing attacks.

Differential cryptanalysis aside, it is easy to see exhaustive key search as a promising attack against Skipjack due to its relatively short key length of 80 bits, but there is a computationally cheap way to increase the effective key length of any block cipher. This mechanism, originally conceived for DES by Rivest in 1984, is known as the DESX contruction [145]. In fact this is the mechanism Li et al. [163] use to extend Skipjack to what they call Skipjack-X in their SenSec framework. The mechanism works as follows. Denote P as

Table 4.2: Terminology in cryptanalysis.

Term	Definition
Encryption/decryption oracle	An abstraction that is independent of the adversary, but encrypts/decrypts plaintexts/ciphertexts for the adversary on request.
Known-plaintext attack	An attack whereby the adversary, who has no access to the encryption or decryption oracle, uses a number of known plaintext-ciphertext pairs in his analysis to recover the key.
Chosen-plaintext attack	An attack whereby the adversary, who has no access to decryption oracle, feeds a number of plaintexts of his choice to the encryption oracle, in order to get the corresponding ciphertexts that would help in his analysis to recover the key.
Chosen-ciphertext attack	Similar to chosen-plaintext attack, except that the adversary (1) has access to the decryption oracle instead of the encryption oracle, (2) the adversary feeds the oracle with ciphertexts of his choice, and (3) the adversary uses a potentially different kind of analysis.
Linear cryptanalysis	A known-plaintext attack whereby the adversary studies the linear approximation to: Plaintext$[i_1, i_2, \ldots, i_a] \oplus$ Ciphertext$[j_1, j_2, \ldots, j_b]$ = Key$[k_1, k_2, \ldots, k_c]$, where $i_1, i_2, \ldots, i_a, j_1, j_2, \ldots, j_b, k_1, k_2, \ldots, k_c$ denote fixed bit locations [174].
Differential cryptanalysis	A chosen-plaintext attack whereby the adversary encrypts a pool of plaintext pairs with chosen differences and filters those pairs that have the expected ciphertext differences – these pairs reveal internal behaviour of the cipher that helps the adversary determine bits of the secret key [29].
Security margin	The year until which breaking the cipher requires more effort than breaking DES in 1982 [161].

the plaintext, K as an 80-bit key, K_1 and K_2 as two 64-bit keys, then Skipjack-X$_{K,K1,K2}$(P) = $K_2 \oplus$ Skipjack$_K(K_1 \oplus P)$. The new key, of Skipjack-X, is therefore 80+64+64=208 bits long. Applying Kilian and Rogaway's [145] analysis, we can show that an adversary's advantage at distinguishing between a Skipjack-X and a random permutation is bounded by $t/2^{(143-\log_2 m)}$, where t is the total number of trial encryptions/decryptions the adversary can perform, and m is the number of plaintext-ciphertext pairs the adversary can obtain. This means the effective length of Skipjack-X is $143 - \log_2 m$. However, if we apply Biryukov and Wagner's [31] *sliding with a twist* cryptanalysis on Skipjack-X, we will get an attack that requires $2^{(64+1)/2} = 2^{32.5}$ known plaintexts and $2^{80-1+32.5} = 2^{111.5}$ offline trial Skipjack encryptions. This means the 208-bit keys that are used with Skipjack-X are only about as strong as 111-bit keys – such a strategy does not seem to constitute a sound investment of memory. Therefore we do not think of Skipjack-X as a worthy replacement for Skipjack in WSNs.

To conclude, we note that while mainstream cryptography is moving away from keys smaller than 128 bits, Skipjack with full 32 rounds is secure, as of today, with a security margin of 2013.

4.2.2 RC5

We have chosen to evaluate RC5 [245] because RC5 is a well-known algorithm that has been around since 1995 without crippling weaknesses. Although `distributed.net` has managed to crack a 64-bit RC5 key in RSA Laboratories' Secret-Key Challenge after 1757 days of computing involving 58,747,597,657 distributed work units, the standard key length of RC5 is 128 bits and RC5 has managed to withstand years of cryptanalysis.

In terms of design, RC5 is an innovative cipher that uses *data-dependent* rotations and is fully parameterised in word size, number of rounds and key length. Such flexibility is rare among mainstream ciphers. RC5 is also designed to be suitable for hardware as well as software implementations. Lastly, Perrig et al. [215], Xue et al. [306] and Liu et al. [165] choose to use RC5 for WSNs. We would like to find out if their choice is justified.

Security

RC5 is conventionally represented as RC5-*w*/*r*/*b*, where *w* the word size is the number of bits in a word of the target computing platform, *r* is the number of rounds, and *b* the key length is the number of bytes of a key. The attacks found up to year 1998 have been summarised in Kaliski and Yin's technical report [133]. The best attack cited is Biryukov et al.'s [30], which breaks RC5-32/12/16 with just 2^{44} chosen plaintexts (compared with 2^{43} known plaintexts for DES), and RC5-32/16/16 with 2^{61} chosen plaintexts. Biryukov et al. therefore recommend increasing the number of rounds to at least 18.

After Biryukov et al., Borst et al. [41] manage to reduce the storage requirements drastically for attacking RC5 (with 64-bit block size) and RC6 (with 128-bit block size) using the linear hull effect [206]. This is known as the first *experimentally executed* known plaintext attacks on reduced versions of RC5 (with up to 5 rounds).

Shimoyama et al. [264] introduce another route of attack based on statistical correlations derived from χ^2 tests. They have managed to derive the last round key of up to 17 rounds by using a chosen plaintext attack. While Shimoyama et al. use a chosen plaintext attack, Miyaji et al. [189] use a known plaintext attack, which breaks RC5-32/10/16 with $2^{63.67}$ plaintexts at a probability of 90%.

The attacks described so far are theoretical. Ironically, intended as a security feature, data-dependent rotation invites implementation-based attacks. One such attack (on RC5-32/12/16) has been proposed by Handschuh et al. [103] which only needs about 2^{20} encryption timings and a time complexity between 2^{28} and 2^{40}. Kelsey et al. [143] describe another implementation-based attack by observing the processor flags.

To conclude, in view of Biryukov et al.'s [30] recommendation and Shimoyama et al.'s [264] discovery, RC5-32/18/16 (18 rounds) should be secure.

4.2.3 RC6

We have chosen to evaluate RC6 [246] because like RC5, RC6 is parameterised and has a small code size. RC6 is one of the five finalists that competed in the AES challenge and according to some AES evaluation reports [198][301], it has reasonable performance. Lastly, RC6 has been chosen as the algorithm of choice by Slijepcevic et al. [266] for WSNs. We are interested in seeing if their choice is justified.

Security

The technical report of the Cryptography Research and Evaluation Committee (CRYPTREC), Information-technology Promotion Agency (IPA) of Japan [70] has a summary of the attacks

known up to year 2001. The first notable attack is by Knudsen et al. [150]. Their attack is based on statistical correlations derived from χ^2 tests. By observing the nonuniformness of the five least significant bits from each of the two Feistel halves in RC6, they require approximately $2^{13.8+16.2r}$ plaintexts to distinguish $3 + 2r$-round RC6 from a pseudorandom permutation. For RC6 with 17 rounds, they estimate that at most, about 2^{118} plaintexts are required.

At the same time, Gilbert et al. [94] present a theoretical attack of 14-round RC6 based on an iterative statistical weakness found in RC6's round function. The attack requires 2^{118} known plaintexts, a memory size of 2^{112} and 2^{122} operations.

Takenaka et al. [276, 277] propose the Transition Matrix Computation technique for evaluating security against χ^2 attacks, by which they are able to deduce the "weakest key" of RC6 against χ^2 attacks.

Shimoyama et al. [263] show that the 64-bit target extended key of 18-round RC6 with a weak key (which exists once in every 2^{90} keys), can be recovered with a probability of 95% by *multiple linear cryptanalysis* with $2^{127.423}$ known plaintexts, a memory of size 2^{64}, and $2^{193.423}$ computations of the round-function.

To conclude, RC6-32/20/16 (20 rounds) is secure.

4.2.4 Rijndael

We have chosen to evaluate Rijndael [73] because Rijndael is the Advanced Encryption Standard, mandated by the NIST of the United States, chosen after extensive scrutiny and performance evaluation (`http://csrc.nist.gov/encryption/aes`). It is also one of the ciphers recommended by the New European Schemes for Signature, Integrity and Encryption (NESSIE) Consortium (`www.cryptonessie.org`), and Japan's CRYPTREC [71]. Rijndael, as the AES, is well studied and there are efficient implementations on a wide range of platforms (`www.rijndael.com`). We would however like to obtain first-hand experience of evaluating Rijndael on our particular platform, which has never been studied before during the evaluation process of the AES.

Security

Like most modern block ciphers, Rijndael is designed with resistance against differential and linear cryptanalysis in mind, using the latest results in cryptographic research [73]. For example, Cheon et al.'s [55] impossible differential cryptanalysis requires $2^{91.5}$ chosen ciphertexts and 2^{122} encryptions to attack 6-round Rijndael-128 (128-bit keys). Gilbert et al. [95] describe (1) an attack on 7-round Rijndael-196 and Rijndael-256 with 2^{32} chosen plaintexts and a complexity of about 2^{140}, and (2) an attack on 7-round Rijndael-128 with 2^{32} chosen plaintexts and a computational complexity slightly less than exhaustive search. Ferguson et al. [86] report a related-key attack of 9-round Rijndael-256 with time complexity 2^{224}, which is of course far from practical. No attack is known for Rijndael with more than 9 rounds.

On the other hand, although Rijndael has been chosen as the AES, the security of Rijndael has gone through twists and turns of controversy. The algebraic nature of Rijndael [87], has interestingly opened up possible avenues of other non-traditional attacks as summarised in the Crypto-Gram Newsletter of September 2002 [255]. It started with Courtois et al. [67, 66] presenting evidence that the security of Rijndael might not grow exponentially as intended with the number of rounds. Their technique is based on expressing the S-boxes of Rijndael in an overdefined system of multivariate quadratic equations which can be solved by an algorithm called XSL. XSL is in turn based on XL [63]. The security of Rijndael therefore lies on

the computational complexity of XL, which to date remains an open problem [65]. In spite of Moh's dispute [190], whether the technique would *not* work remains to be proven [256]. In the meantime, Murphy et al. [193] derive an alternative representation of Rijndael that is easier to cryptanalyse, by embedding Rijndael in a cipher called BES that uses only simple algebraic operations in $GF(2^8)$. They show that Rijndael encryption can then be described by an extremely sparse overdetermined multivariate quadratic system over $GF(2^8)$, whose solution would recover the key. In another paper, Murphy et al. [192] argue that while XSL does not have estimates accurate enough to substantiate claims of the existence of a key recovery attack, XSL does help solve their $GF(2^8)$ system of equations more efficiently than Courtois et al.'s $GF(2)$ system of equations. Combining Coppersmith's [61] correction of Courtois et al.'s estimates, Murphy et al. further deduce that the security of Rijndael-128 would be reduced from the theoretical complexity of exhaustive key search, 2^{128} to 2^{100}, *if* XSL is a valid technique.

On the other front, Fuller et al. [90] unravel serious linear redundancy in the only non-linear component, i.e. the S-box, of Rijndael: the 8×8 S-box behaves actually like a 8×1 S-box. They, by studying the invariance properties of the local connection structure of affine equivalence classes, discover that the outputs of the S-box are all equivalent under affine transformation. The essence of their discovery can be summarised in the following simple mathematical expression: Let $b_i(x)$ and $b_j(x)$ be two distinct outputs of the S-box, then there exists a non-singular 8×8 matrix D_{ij} and a binary constant c_{ij} such that $b_j(x) = b_i(x)D_{ij} \oplus c_{ij}$.

Independent of the above development, Filiol shocked the scientific community in January 2003 by announcing a break of Rijndael with his plaintext-dependent repetition codes cryptanalysis technique [88]. By detecting bias in the boolean functions of Rijndael, Filio claimed that he was able to obtain 2 bits of a Rijndael key with only 2^{31} ciphertexts and a computational complexity of mere $O(2^{31})$. Fortunately, several independent cryptographers were quick to dismiss the claim [64].

To conclude, the research on Rijndael has entered an interesting era. More and more previously unknown properties are now being discovered and analysed [23][284][314]. Despite the above debate, we are adopting the recommendation of NESSIE [231] and CRYPTREC [71] that Rijndael is secure.

4.2.5 Twofish

We have chosen to evaluate Twofish [258] because it allows a wide range of tradeoffs between size and speed. It is also designed to be efficient on a wide range of platforms. Since Twofish is more efficient than RC6 on some embedded processor platforms, e.g. on 8-bit Z80 [249] and on 8-bit 6805 [141], and since we are investigating RC6, Twofish is also included in our investigation.

Security

One of the earliest cryptanalytic attempts [186] from outside the Twofish design team found flaws in the original security assessment of the key schedule, but results in no practical implication. One of the unprecedented features of Twofish is its key-dependent S-boxes (conventional S-boxes are fixed), but Murphy et al. [195, 194], using primarily (1) the fact that the key-dependent S-boxes are determined only by half of the key, and (2) differential cryptanalysis, show that key-dependent S-boxes, as used in Twofish, provide no additional security over well-designed fixed S-boxes, and may in fact improve the range of options available to attackers. Nevertheless, it is still an open question whether any practical attack will result

from Murphy et al.'s analysis [259][142]. Biham et al.'s [28] impossible differential cryptanalysis needs 1.82×2^{128} one-round computations and 2^{64} chosen plaintexts to break 6-round Twofish with 128-bit keys. The best known attack is Lucks' [167] saturation attack, which requires 2^{127} chosen plaintexts and 2^{126} encryptions (i.e. two times faster than exhaustive search) to break 7-round Twofish with 128-bit keys.

To conclude, 16-round Twofish is secure.

4.2.6 MISTY1

We have chosen to evaluate MISTY1 [172] because MISTY1 is one of the CRYPTREC-recommended [71] 64-bit ciphers and is the predecessor of KASUMI, the 3GPP-endorsed encryption algorithm [7]. MISTY1 is specifically designed to resist differential and linear cryptanalysis. MISTY1 is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. We find MISTY1 to be particularly suitable for 16-bit platforms. MISTY1 is a royalty-free open standard documented in RFC2994 [207].

Security

Babbage et al. [22] demonstrate the possibility of a 7th order differential cryptanalytic attack on 5-round MISTY1. According to them, none of the S-boxes with optimal linear and differential properties has an optimal behaviour with respect to higher order differential cryptanalysis, however as improvement, the number of rounds of the FI function could be increased. As we will see later, KASUMI, derived from MISTY1, incorporated such improvement, in that it has four instead of three S-boxes in its FI function.

Kühn [153] found an impossible differential attack on 4-round MISTY1 using 2^{38} chosen plaintexts and 2^{62} encryptions. In the same paper, a collision-search attack has also been described – the attack requires 2^{28} chosen plaintexts and 2^{76} encryptions. In a later paper, Kühn [154] shows that the FL function introduces a subtle weakness in 4-round MISTY1. This weakness allows him to launch a slicing attack with as few as $2^{22.25}$ chosen plaintexts, with a memory requirement of $2^{34.2}$ bytes and time complexity of 2^{45}, on 4-round MISTY1.

The best known attack on 5-round MISTY1 so far is Knudsen and Wagner's [151] integral cryptanalysis, at a cost of 2^{34} chosen plaintexts and 2^{48} time complexity.

To conclude, MISTY1 with full 8 rounds is secure.

4.2.7 KASUMI

We have chosen to evaluate KASUMI [7] because KASUMI, as the 3GPP-endorsed encryption algorithm [7], is presumably well-suited for embedded applications, and has gone through considerable expert scrutiny.

Security

KASUMI more or less inherits the security and performance benefits of MISTY1.

At the same time reporting attacks on MISTY1 and MISTY2, Kühn [153] is also able to extend the attacks to KASUMI. His impossible differential attack on 6-round KASUMI requires 2^{55} chosen plaintexts and 2^{100} encryptions.

Kang et al. [134, 135] prove that 3-round KASUMI is not a pseudorandom permutation ensemble but 4-round KASUMI is a pseudorandom permutation ensemble. However Tanaka et al. [278] show that 4-round KASUMI without the FL functions can be attacked using

effective 2nd order differentials. The attack can be used to find the 6 sub-keys at the 4th round, at a cost of 1416 chosen plaintexts and $2^{22.11}$ times FO function operations.

To conclude, KASUMI with full 8 rounds is secure.

4.2.8 Camellia

We have chosen to evaluate Camellia [17] because Camellia is one of the NESSIE- and CRYPTREC-recommended [71] 128-bit ciphers. Camellia is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. Aoki et al. [16] claim their hardware implementation of Camellia occupies only 7.875K gates using a 0.11 μm CMOS ASIC library and is in the smallest class among all existing 128-bit block ciphers. Camellia is designed not only to be resistant to differential cryptanalysis, linear cryptanalysis, higher order differential attacks, interpolation attacks, related-key attacks, truncated differential attacks, boomerang attacks and slide attacks, but also with a large safety margin in view of anticipated progress in cryptanalysis techniques [16][173]. Futhermore, Camellia is royalty-free.

Security

He et al. [107] discover that the Square attack [72] is not only applicable to the Square block cipher but also to ciphers with a Feistel structure. The complexity of their attack on 6-round Camellia is 3328 chosen plaintexts and 2^{112} encryptions. Sugita et al. [274] found a non-trivial 7-round impossible differential. Lee et al.'s [160] truncated differential cryptanalysis of 7-round Camellia without the FL/FL^{-1} functions, requires only 192 encryptions but $2^{82.6}$ chosen plaintexts to recover an 8-bit key. Yeom et al. [311] propose a Square attack on 9-round Camellia with 256-bit keys, at a cost of $2^{60.5}$ chosen plaintexts and $2^{202.2}$ encryptions. Hatano et al.'s [106] attack of 11-round Camellia with 256-bit keys, requires 2^{93} chosen plaintexts and $2^{255.6}$ encryptions, which is just a little less than brute force search.

To conclude, 18-round Camellia is secure.

4.2.9 Other Ciphers That Are Not Considered

3-DES [254] is not considered because it is inefficient, involving 3 encryptions. IDEA and SAFER_{++} (of the SAFER family of ciphers) are not considered, because there are concerns in the NESSIE consortium about IDEA's key schedule, and certain structural properties of SAFER_{++} [201]. Among the AES finalists, MARS [45] and Serpent [15] are not considered. For MARS, the reason is its high algorithmic complexity, which results in the highest RAM and ROM usage among the AES finalists [198][249][257], and it has actually got the least number of votes during the last round of voting for the AES. While Serpent is the 1st runner-up during the last round of voting, it performs consistently poorly in software encryption and decryption [198][301] due to its large security margins – researchers who voted for Serpent prioritised security over performance. MARS and Serpent also have not been submitted to, nor considered by NESSIE or CRYPTREC. Blowfish [253] is not considered because it is superseded by Twofish, which improves on Blowfish's key schedule, and en/decryption speed [260]. SHACAL-2 [104] is a new 256-bit hash function-based block cipher recommended by NESSIE, but has not been been studied by NIST or CRYPTREC. The specific reasons it is not chosen for our investigation are that (1) the security offered by a key length of 256 bits is out of proportion with the relative lack of physical security in sensor nodes; (2) using 256-bit keys requires twice the storage that is required by 128-bit keys, causing an unnecessary strain on the available memory that is already scarce – note that 256-bit keys are

not used in TinySec [138] either.

4.3 Methodology and Consideration

For benchmarking, we consider: (1) the cipher parameters, (2) the cipher operation modes, (3) the compiler toolchain, and (4) the implementation sources.

4.3.1 Cipher Parameters

Table 4.3 lists the parameters we have adopted for each cipher (some of them actually have fixed, unadjustable parameters but we list them anyway for clarity's sake). The number of rounds for each cipher is nominal except for RC5, where 18 is used instead of the nominal 12, for security reasons already described in Section 4.2.2. Although RC5 and RC6 allow variable word size, without the backing of relevant cryptanalytic research, we are not sure how many rounds to use if we pick a non-standard word size of 16 bits, which is the word size of our platform. Therefore, for RC5 and RC6, we are using the standard word size of 32 bits.

Table 4.3: Cipher parameters (lengths in bytes).

Cipher	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY	KASUMI	Camellia
Block length	8	8	16	16	16	8	8	16
Key length	10	16	16	16	16	16	16	16
Rounds	32	18	20	10	16	8	8	18

4.3.2 Operation Modes

The naïve approach of encrypting a message longer than one block, by dividing the message into multiple blocks and encrypting the blocks separately, is called the *electronic codebook mode* (ECB) mode. ECB is insecure since an adversary can construct valid ciphertexts from the original ciphertext by arbitrarily rearranging, repeating and/or omitting blocks from the original ciphertext. More secure operation modes in Table 4.4 are used in practice. These operation modes do not only affect the security, but also the energy efficiency of the encryption scheme, which will be measured in the following investigation.

Table 4.4: Comparison of operation modes.

Operation mode	On ciphertext error...	On synchronisation error...
Cipher-Block Chaining (CBC)	An erroneous bit affects the entire current block and the corresponding bit of the next block.	Lost blocks need to be re-transmitted to decrypt the next block.
Cipher Feedback Mode (CFB)	An erroneous bit affects the corresponding bit of the current block and the entire next block.	Lost blocks need to be re-transmitted to decrypt the next block.
Output Feedback Mode (OFB)	An erroneous bit affects the corresponding bit of the current block.	Lost blocks do not need to be re-transmitted.
Counter (CTR)	Similar to OFB.	Similar to OFB.

As stated in Table 4.4, different operation modes also have different fault tolerance characteristics against ciphertext errors (where ciphertext bits are changed during transmission), and synchronisation errors (where whole ciphertext blocks are lost), but we will explain why these characteristics do not really matter.

In case of a ciphertext error, a node would typically either request the corresponding packet to be retransmitted, or ignore the error. If retransmission is requested, regardless of the operation mode used, the same penalty, in the form of energy required for transmitting a packet, applies. Otherwise, using different modes only means putting up with different degrees of error, e.g. CBC and CFB have to put up with more errors compared with OFB and CTR according to Table 4.4.

To put synchronisation error into context, we first need to know how ciphertext blocks are transmitted. Usually, a network packet corresponds to one or more ciphertext blocks, and every packet contains an *initialisation vector* (IV) in the header, allowing it to be decrypted independently from other packets, as is the case with TinySec. So a lost packet does not constitute a synchronisation error, unless the packet is fragmented, *and* one or more of the fragments are lost. When a node discovers it has lost a packet fragment, there are typically 3 options: (1) request the fragment to be retransmitted, (2) abort receiving all ensuing fragments, or (3) wait for the ensuing fragments to arrive naturally until a time-out. If option 1 is taken, then the same retransmission penalty applies to all operation modes. If option 2 is taken, all ensuing fragments are lost and so the same penalty in the form of information loss applies to all operation modes. If however option 3 is taken, only OFB and CTR can decrypt all received fragments (i.e. all fragments that arrived naturally) according to Table 4.4. That is to say, in theory, using option 3 means putting up with different degrees of information loss according to the operation mode used. For example, if 3 fragments are transmitted, each of the fragments contains 4 ciphertext blocks, and if only the 1st and 3rd fragment are successfully received, OFB and CTR would be able to decrypt all ciphertext blocks in the 1st and 3rd fragment correctly, whereas CBC and CFB would only be able to decrypt the ciphertext blocks in the 1st fragment, and the last 3 blocks in the 3rd fragment. In practice, when one fragment is lost, ensuing fragments would likely be lost too [275], in which case the same information loss penalty applies to all operation modes.

Summarising from the above analysis, in the event of a ciphertext error or synchronisation error, using different modes either lead to the same (whether energy or information loss) penalty, or different information loss penalties. In other words, the different fault tolerance characteristics of operation modes do not result in different penalties in energy. We can thus safely consider the energy efficiency of an operation mode solely from an algorithmic point of view.

Some notes about our implementation: our CBC supports ciphertext stealing [254], so padding is not required. For CFB and OFB, we are using a feedback size that is equal to the block size.

4.3.3 Compilers

For compilation, we are currently only using IAR Systems' MSP430 C Compiler V2.20A/W32 (`www.iar.com`) with a patched linker, IAR Universal Linker of version 4.56F. For debugging and profiling, we use IAR Systems' Embedded Workbench 3.2 with the integrated C-Spy Debugger and profiler plug-in. Another viable compiler is the GNU C compiler in the MSPGCC toolchain (`mspgcc.sf.net`), however we are not using it due to the lack of profiling support by the toolchain. That said, we do not rule out the possibility of performing our benchmarks using the toolchain as it continues to mature in the future. Note that the chip supplier itself Texas Instruments offers only the Kickstart version of the toolchain we are using.

In our benchmarks, we compare maximum size optimisation with maximum speed optimisation. The IAR compiler supports 3 levels of optimisation in terms of size or speed: High,

Medium and Low, as well as 4 kinds of transformations: common subexpression elimination (`ELIM`), loop unrolling (`UNROLL`), function inlining (`INLINE`) and code motion (`MOTION`). How these optimisations and transformations are used is described in the next section.

4.3.4 Implementation Sources

To avoid reinventing the wheel, we try to use and improve as much existing source code as possible. The following describes how and from where the implementation of each cipher is adapted.

The implementation of Skipjack is adapted from TinySec. Skipjack consists of executing two shift register algorithms called Rule A and Rule B: 8 rounds of Rule A, then 8 rounds of Rule B, then 8 rounds of Rule A and 8 rounds of Rule B again, resulting in a total of 32 rounds. At each round, 4 bytes of the key are used, and when the key is exhausted, the key is used from the beginning again (i.e. wrapped around). This offers a lot of freedom in the implementation. For example, to minimize storage, we can use exactly as many bytes for the expanded key as there are in the user key – our size-optimised implementation adopts this strategy. For another example, to minimize the number of instructions we can expand the user key to 4×32 bytes so that the key pointer never has to wrap around (Figure 4.1(a)) – TinySec's implementation adopts this strategy. As a compromise, we can also expand the key such that the key pointer only wraps around after 8 rounds of Rule A or Rule B (Figure 4.1(b)) – our speed-optimised implementation adopts this strategy. Measurements show that compared to TinySec's strategy, our strategy not only saves 90 bytes of memory per expanded key but is also slightly more efficient *when speed-optimised*.

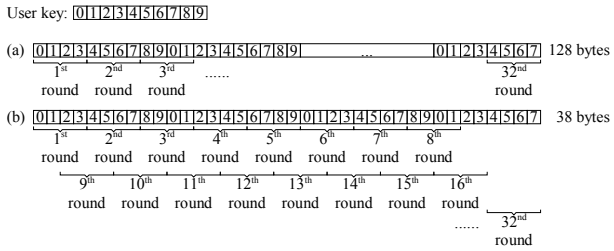


Figure 4.1: Key expansion strategies for Skipjack.

The implementation of RC5 is adapted from that of OpenSSL (`www.openssl.org`). The speed-optimised version is basically the same as the size-optimised version, with some loops unrolled.

RC6 is implemented from scratch according to the specification. The speed-optimised version does not have its loops unrolled because the resultant code size exceeds 10 KB, with marginal increase in energy efficiency.

The implementation of Rijndael is adapted from that of OpenSSL. In OpenSSL, Rijndael is fully accelerated using 10 tables, occupying 10 KB of memory (ROM in our case). Following Hachez et al.'s [102] suggestion, we use only 4 tables – 2 tables for S-boxes, 2 tables for combined S-box lookups, matrix multiplications and rotations, reducing the table space from 10 KB to 2600 bytes, in both our size- and speed-optimised implementations. In principle, we can reduce the code size of the size-optimised version further by doing away with the last 2 tables, but the resultant code size generated by the IAR compiler far exceeds the table space saved. So the number of tables used in our case can be considered minimum.

The implementation of Twofish is adapted from Whiting's [298] optimised implementation. Whiting's implementation by design offers 4 layers of performance tradeoffs, in decreasing en/decryption speed: (1) full keying, (2) partial keying, (3) minimal keying, and (4) zero keying. All options, except zero keying, are however impractical for MSP430F149 as the amount of table space for the key schedule is at least 1 KB, which is half the size of the RAM. Therefore, our size-optimised implementation of Twofish is based on Whiting's implementation with zero keying. Our speed-optimised implementation is also based on Whiting's implementation with zero keying, but with Worley et al.'s [301] optimisation incorporated.

The implementation of MISTY1 is adapted from Matsui et al.'s [173]. In our size-optimised version, only the minimum number of expanded keys are stored in the RAM, while in our speed-optimised version, more expanded keys are stored to speed up en/decryption. Attemps to accelerate MISTY1 further using Botan's (`botan.randombit.net`) table-lookup technique fail, because the IAR compiler fails to terminate on Medium and High-level speed optimisation.

The implementation of KASUMI is adapted from the reference implementation in the specification [7]. The speed-optimised version is basically the same as the size-optimised version, with some loops unrolled.

The implementation of Camellia is adapted from the reference implementation [187]. The size-optimised and speed-optimised versions are the same code, but compiled with different compiler options.

The invocation interface of each cipher follows that of OpenSSL. Table 4.5 lists the levels of optimisation and kinds of transformations we use for each cipher. Furthermore, all code is compiled to use the hardware multiplier. For MISTY1 and KASUMI, the IAR compiler has trouble applying Medium/High size/speed optimisation, hence only Low optimisation is used. Our source code and benchmark results can be found at `http://wwwes.cs.utwente.nl/eyes/crypto_test.zip`.

Table 4.5: Optimisations and transformations.

Cipher	Size optimisation	Speed optimisation
Skipjack	High	High, ELIM, MOTION
RC5, RC6, Rijndael	High, ELIM, MOTION	High, ELIM
Twofish	High	High
MISTY1, KASUMI	Low, ELIM*	Low
Camellia	High, ELIM, MOTION	High, ELIM, MOTION

* Applied only to the key setup routine.

4.4 Results

We have performed our measurements in standalone mode, i.e. without interaction with an OS. We have taken care in making the interface of the cipher implementations as uniform as possible, so that no difference in performance is a result of the difference in the interfaces. Our benchmark parameters are memory and CPU cycles. Given in Section 4.4.1 and Section 4.4.2 are the results in terms of these two parameters, followed immediately by our observation and analysis in Section 4.4.3.

4.4.1 Memory

We refer to two types of memory: (1) code memory, in the form of Flash memory and (2) data memory, in the form of RAM. The memory organisation of MSP430F149 is such that data memory ranges from address 0200h to 09FFh, whereas code memory ranges from 01100h to 0FFFFh. The IAR compiler generates 3 types of segments for MSP430: CODE, CONST and DATA. A typical policy is to put CODE and CONST segments in the code memory, and DATA segments in the data memory. Each segment type is further divided into the following sub-types:

1. CODE: CODE (program code), CSTART and INTVEC;

2. CONST: DATA16_AC, DATA16_C (constants including string literals), DATA16_ID (initial values of static and global variables) and DIFUNCT;

3. DATA: CSTACK (the C runtime stack), DATA16_AN, DATA16_I (initialised variables), DATA16_N, DATA16_Z and HEAP.

Of all the above segment sub-types, we use only the underlined ones. The memory usage of these segments can be read off the list files generated by the compiler, and the results are shown in Tables 4.6 and 4.7.

In Table 4.6, some entries have two figures separated by a comma. The first figure applies to encryption, and the second, decryption, e.g. size-optimised Rijndael needs 14 bytes of RAM for setting up an encryption key, but 30 bytes for setting a decryption key. When an entry has only one figure, the figure for encryption equals the figure for decryption.

In Table 4.7, the code memory for each CBC module takes into account (1) the code memory for key setup, (2) the code memory for barebone encryption and decryption, (3) the code memory for lookup tables, as well as (4) the code memory for CBC-specific parts. CFB, OFB and CTR do not use the decryption function [254], hence the code memory for each CFB/OFB/CTR module is similarly calculated except that the code memory for decryption, decryption key setup and lookup tables for decryption are not included. Note that the storage for plaintext, ciphertext, and cipher key is not included in Table 4.6 nor Table 4.7.

Table 4.6: Data memory requirements.

Component	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimised:								
Expanded key	12	152	176	176	168	32	128	208
Key setup	4	64	60	14, 30	58, 50	4	58	170
CBC	54	58	94	92, 96	88, 92	56	56	142
CFB	38	42	62	60	56	40	40	110
OFB	38	42	62	60	56	40	40	110
CTR	40	44	64	62	58	42	42	112
Speed-optimised:								
Expanded key	38	152	176	176	168	64	128	208
Key setup	6	64	58	10, 26	56, 44	4	38	184
CBC	56	62, 58	98	96, 100	90, 94	60	60	146
CFB	36	42	62	60	54	40	40	110
OFB	36	42	62	60	54	40	40	110
CTR	38	44	64	62	56	42	42	112

Table 4.7: Code memory requirements.

Mode	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimised:								
CBC	1812	2076	2732	8684	9612	7972	9842	19864
CFB	850	1104	1328	4276	7786	4538	5446	12382
OFB	778	1032	1256	4204	7714	4466	5374	12310
CTR	856	1110	1406	4354	7864	4544	5452	12460
Speed-optimised:								
CBC	2610	7502	3174	9984	12538	8596	11078	29516
CFB	1068	4290	1340	4590	9302	4906	5994	17148
OFB	988	4210	1260	4510	9222	4826	5914	17068
CTR	1066	4288	1412	4662	9374	4904	5992	17220

4.4.2 CPU Cycles

The computational complexity of an algorithm translates directly to its energy consumption. Assuming the energy per CPU cycle is fixed (which is justified in the Appendix), then by measuring the number of CPU cycles executed per byte of plaintext processed, we get the amount of energy consumed per byte. For example MSP430F149 draws a nominal current of 420 μA at 3 V and at a clock frequency of 1 MHz in active mode [279] – this means that the energy per instruction cycle (*for the processor alone*) is theoretically 1.26 nJ.

To evaluate the ciphers, we must determine the range of plaintext lengths that is of greatest interest to WSNs. Here is how we arrive at the choice of 8 to 96 bytes. The lower limit of 8 bytes is close to the minimum packet size (usually the size of control packets) used in mainstream link-layer protocols [309][285][226]. The upper limit of 96 bytes is close to the maximum packet size used for simulating and benchmarking WSN protocols [226][285].

Figure 4.2 shows our measurements for CBC encryption. CBC decryption consumes a slightly different number of CPU cycles, but the relative ordering between the various ciphers remains the same. Since an RC5/RC6 en/decryption executes a different number of rotations depending on the key and the plaintext/ciphertex, the figures for RC5 and RC6 in Figure 4.2 are obtained by averaging over 450 encryptions using a different pseudorandomly generated key and plaintext each time. Only 450 encryptions are used because (1) the keys and plaintexts are pseudorandomly generated offline and downloaded to the 60 KB ROM of the MSP430F149, and (2) they can be done in under a minute on the profiler. That the plots for the 64-bit block ciphers: Skipjack, RC5, MISTY1 and KASUMI appear as straight lines should be interpreted as follows: the measurements are only taken at full block lengths for these ciphers. Were the measurements taken at non-full block lengths, the plots for these ciphers would appear jagged-like, just as the plots for other, 128-bit block ciphers do. When size-optimised, the CPU cycles per byte of Rijndael, Skipjack and KASUMI are only a few clock cycles apart, with Rijndael slightly more efficient than Skipjack, and Skipjack slightly more efficient than KASUMI at large packet sizes.

For CFB, OFB and CTR with the exception of RC5 and RC6, we found that the number of CPU cycles consumed per byte, y, can be approximated by

$$y \approx \frac{C_f + xC_b + \left\lceil \frac{x}{B} \right\rceil C_B}{x} \tag{4.2}$$

where x is the plaintext length, B is the cipher's block length (both in bytes); C_f, C_b and

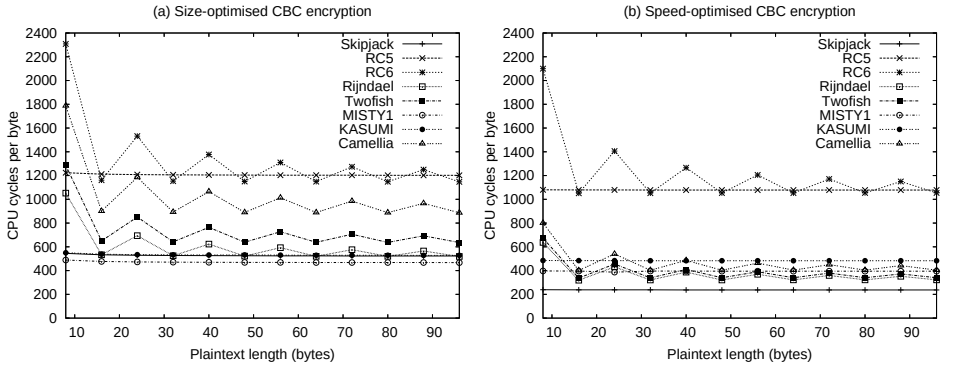


Figure 4.2: CPU usage of CBC encryption when (a) size-optimised, and (b) speed-optimised.

C_B are constants. In Equation 4.2, the first term C_f accounts for the function call overhead, the second term xC_b accounts for the overhead of organising x bytes into B-byte blocks, and the last term accounts for the actual en/decryption process and in CTR's case the increment of a counter. This approximation does not apply to CBC because with ciphertext stealing CBC involves more complicated computation. This approximation does not apply to RC5 and RC6 either because an RC5/RC6 en/decryption executes a different number of rotations depending on the data and the key, resulting in a non-constant value for C_B. Thus for RC5 and RC6, the following equation is more appropriate:

$$y \approx \frac{C_f + xC_b + \left\lceil \frac{x}{B} \right\rceil V_B}{x} \tag{4.3}$$

where V_B is a variable. Table 4.8 lists the values of C_f, C_b, C_B (for Skipjack, Rijndael, Twofish, MISTY1, KASUMI and Camellia) and the average value of V_B (for RC5 and RC6) obtained through least squares fitting. The standard error of each of the constants is less than 10^{-11}. Note that CFB, OFB and CTR consume exactly the same number of CPU cycles for both encryption and decryption.

We now look at the operation modes. Although Figure 4.3 only compares the operation modes in the context of Rijndael encryption, the same comparison applies to all other ciphers. Figure 4.3 shows that in terms of efficiency, OFB > CTR > CFB > CBC when size-optimised ('>' meaning 'is more efficient than'). When speed-optimised, OFB is still the most efficient mode, but CBC > CFB > CTR > at plaintext lengths = integer × block size, and CFB > CTR > CBC at other plaintext lengths – this is when ciphertext stealing is activated, thereby degrading the efficiency of CBC. Anyhow the performance difference between all modes is smaller when they are speed-optimised than when they are size-optimised.

Apart from en/decryption, we are also interested in the efficiency of the key setup algorithms. Table 4.9 has the results. Notice that only Rijndael and Twofish incur overhead when converting an encryption key to a decryption key, and converting a decryption key to an encryption key. For Skipjack, the speed-optimised version performs worse than the size-optimised version because the expanded key used in the speed-optimised version is 38 bytes long compared with 12 bytes in the size-optimised version (difference in expanded key length explained in the next section). Using a longer expanded key allows en/decryptions to execute faster.

Table 4.8: Values of C_f, C_b, C_B (or V_B) for CFB, OFB and CTR.

Mode	Param.	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimised:									
CFB	C_f	94	94	94	94	94	94	94	94
	C_b	51	51	51	51	51	51	51	51
	$C_B(V_B)$	3550	9241	16924	6537	8230	3363	3849	11949
OFB	C_f	82	82	82	82	82	82	82	82
	C_b	27	27	27	27	27	27	27	27
	$C_B(V_B)$	3550	9241	16923	6537	8230	3363	3849	11949
CTR	C_f	90	90	90	90	90	90	90	90
	C_b	33	33	33	33	33	33	33	33
	$C_B(V_B)$	3593	9285	16982	6594	8287	3406	3892	12006
Speed-optimised:									
CFB	C_f	86	86	86	86	86	86	86	86
	C_b	35	35	35	35	35	35	35	35
	$C_B(V_B)$	1594	8322	16279	4573	4866	2859	3561	5885
OFB	C_f	82	82	82	82	82	82	82	82
	C_b	27	27	27	27	27	27	27	27
	$C_B(V_B)$	1594	8322	16278	4573	4866	2859	3561	5885
CTR	C_f	90	90	90	90	90	90	90	90
	C_b	33	33	33	33	33	33	33	33
	$C_B(V_B)$	1637	8366	16328	4622	4915	2902	3604	5934

4.4.3 Observation and Analysis

First about the operation mode. According to the results in the previous section, OFB is the most energy-efficient mode, and CBC is the least energy-efficient mode when there are partial (plaintext/ciphertext) blocks. OFB is the obvious choice if we only consider the case of two communicating parties. However energy-efficient communications in WSNs often require more than two parties to be involved in a secure group, for example in the form of *passive participation* [320]. In passive participation, a node decides whether to transmit its own packets based on the packets it received from its neighbours – not reporting data that are superseded a neighbour's helps save energy.

Using OFB, and for the matter CFB and CTR, effectively in a group setting is problematic. For example in a group composed of nodes *A*, *B* and *C*, all sharing the same key *K* and the same IV *n* (or counter in case of CTR mode), when *A* broadcasts a packet encrypted with *K* and *n*, if *B* succeeds but *C* fails to receive the packet, *C* would not know it should use a different IV than *n*, and if *C* decides to send a packet encrypted with *K* and the same IV *n*, all is lost because the IV in the OFB/CFB/CTR mode can never be re-used. A suggestion to overcome this is to use a longer IV and to partition the IV space by node ID. In this approach, the IV can be variable in length if the group communication protocol is to accommodate a variable number of members dynamically, at the expense of protocol complexity. If the IV is fixed according to the maximum supported group size, bandwidth and energy are wasted whenever the actual group size is lower than the maximum supported group size. Either way, using OFB in the group setting is awkward and not energy-efficient. CBC in contrast is easier and safer to use for group communications [138]. CBC is not the most efficient mode, but we may find comfort in the fact that CBC is actually quite close to other operation modes in efficiency when speed-optimised, and the difference only gets smaller with increasing plaintext length according to Equation 4.2.

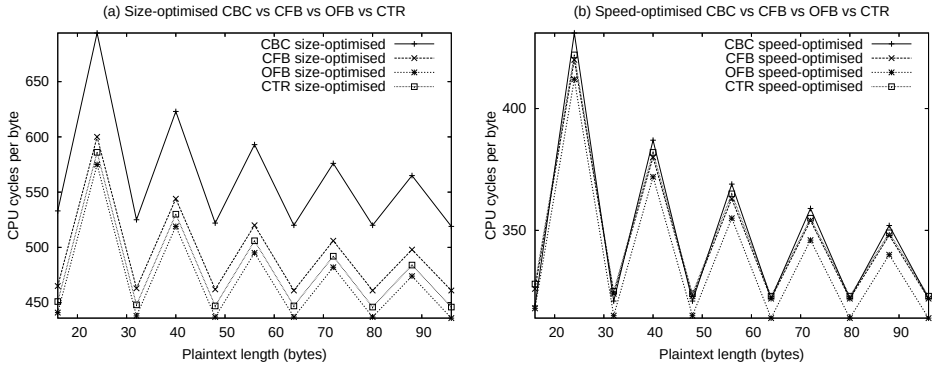


Figure 4.3: CBC vs CFB vs OFB vs CTR for Rijndael encryption when (a) size-optimised, and (b) speed-optimised.

Table 4.9: CPU cycles for key setup (per key).

Operation	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimised:								
Enc	163	44927	45701	*1637	+11187	977	2564	23211
Dec	163	44927	45701	9832	10696	977	2564	23211
Enc→Dec	0	0	0	8195	10696	0	0	0
Dec→Enc	0	0	0	1637	491	0	0	0
Speed-optimised:								
Enc	187	40579	43246	1195	8049	615	1485	15335
Dec	187	40579	43246	5591	7553	615	1485	15335
Enc→Dec	0	0	0	4396	7553	0	0	0
Dec→Enc	0	0	0	1195	496	0	0	0

* Applicable only to CBC mode; 1717 cycles in CFB/OFB/CTR mode.
+ Applicable only to CBC mode; 10707 cycles in CFB/OFB/CTR mode.
Note:

1. Enc (Dec) = setting up an encryption (decryption) key from scatch.
2. Enc→Dec (Dec→Enc) = converting an encryption (decryption) key to a decryption (encryption) key.

Next we analyse Figure 4.2, Table 4.6, 4.7, 4.8 and 4.9 cipher by cipher:

Skipjack: Skipjack produces the shortest expanded key, requires the least code and data memory. When size-optimised, it is slightly less energy-efficient than Rijndael, but when speed-optimised, it is the most energy-efficient cipher.

RC5: RC5 requires little code memory but has poor energy efficiency because multiplication and rotation are the Archilles' heel of MSP430F149: multiplication takes 9 cycles and rotation can only be done one bit at a time. It is for the same reason that speed optimisation does not improve the energy efficiency of RC5 significantly.

RC6: Like RC5, RC6 is lean in code size. The code size is in the range of 1 KB, close to the code size of Sano et al.'s [249] implementation on the ZiLOG Z80 8-bit microprocessor. In terms of CPU cycles, our measurements are not far from Hachez et al.'s [102] measurements on the 8-bit processor Intel 8051: when speed-optimised, key setup takes 43246 clock cycles on MSP430F149, compared to 43200 on 8051; encryption of one block takes 16265

on MSP430F149, compared to 14400 on 8051. For the same reason that applies to RC5, RC6 is poor in energy efficiency, and is barely improved by speed optimisation. The observation that RC6 is a big RAM consumer and performs poorly on 8/16-bit architectures such as the MSP430F149, is confirmed by benchmarks done over a spectrum of architectures [102][198][257].

Rijndael: Rijndael has moderate code size, for the number of tables we use in the implementation. It is the second most energy-efficient cipher both when size-optimised and when speed-optimised.

Twofish: Twofish has larger code size than Rijndael. The tables used by Twofish cannot be economised without seriously degrading its energy efficiency. Twofish is almost as energy-efficient as Rijndael in en/decryption when speed-optimised, but is significantly worse in key agility.

MISTY1: MISTY1 has moderate code size, larger than RC5 and RC6, but smaller than Rijndael. Its RAM usage is only larger than Skipjack's. When size-optimised, MISTY1 is the most energy-efficient cipher. There is no significant speed increase in the speed-optimised version mainly because the IAR compiler has problem applying any level of optimisation above Low.

KASUMI: KASUMI has larger code size than MISTY1 and Rijndael. Although the key setup of KASUMI is linear [78], it takes more than 2 times as long as that of MISTY1 which is non-linear. KASUMI is less energy-efficient than MISTY1 since it is algorithmically more complicated and speed optimisation does not provide significant improvement for the same reason given previously for MISTY1.

Camellia: Although Camellia is more energy-efficient than RC5 and RC6, it has the largest code and data memory requirement. The expanded key occupies more than 10% of the RAM, but computing the round subkeys on the fly would significantly worsen the the energy efficiency in en/decryption that is already lacking.

Table 4.10: Ranking of ciphers*.

Code memory		Data memory		En/decryption efficiency		Key setup efficiency	
CBC	OFB	CBC	OFB	CBC	OFB	Encryption	Decryption
$Skipjack_z$	$Skipjack_z$	$Skipjack_z$	$Skipjack_z$	$Skipjack_s$	$Skipjack_s$	$Skipjack_z$	$Skipjack_z$
$RC5_z$	$Skipjack_s$	$MISTY1_z$	$MISTY1_z$	$Rijndael_s$	$Rijndael_s$	$Skipjack_s$	$Skipjack_s$
$Skipjack_s$	$RC5_z$	$Skipjack_s$	$Skipjack_s$	$Twofish_s$	$Twofish_s$	$MISTY1_s$	$MISTY1_s$
$RC6_z$	$RC6_z$	$MISTY1_s$	$MISTY1_s$	$MISTY1_s$	$MISTY1_s$	$MISTY1_z$	$MISTY1_z$
$RC6_s$	$RC6_s$	$KASUMI_z$	$KASUMI_s$	$Camellia_s$	$Camellia_s$	$Rijndael_s$	$KASUMI_s$
$RC5_s$	$Rijndael_z$	$KASUMI_s$	$KASUMI_z$	$MISTY1_z$	$Rijndael_z$	$KASUMI_s$	$KASUMI_z$
$MISTY1_z$	$RC5_s$	RC5	RC5	$KASUMI_s$	$MISTY1_z$	$Rijndael_z$	$Rijndael_s$
$MISTY1_s$	$MISTY1_z$	$Twofish_z$	$Twofish_s$	$Rijndael_z$	$Skipjack_z$	$KASUMI_z$	$Twofish_s$
$Rijndael_z$	$Rijndael_s$	$Twofish_s$	$Twofish_z$	$Skipjack_z$	$KASUMI_s$	$Twofish_s$	$Rijndael_z$
$Twofish_z$	$MISTY1_s$	$RC6_z$	Rijndael	$KASUMI_z$	$KASUMI_z$	$Twofish_z$	$Twofish_z$
$KASUMI_z$	$KASUMI_z$	$Rijndael_z$	RC6	$Twofish_z$	$Twofish_z$	$Camellia_s$	$Camellia_s$
$Rijndael_s$	$KASUMI_s$	$RC6_s$	$Camellia_z$	$Camellia_z$	$Camellia_z$	$Camellia_z$	$Camellia_z$
$KASUMI_s$	$Twofish_z$	$Rijndael_s$	$Camellia_s$	$RC6_s$	$RC6_s$	$RC5_s$	$RC5_s$
$Twofish_s$	$Twofish_s$	$Camellia_z$		$RC5_s$	$RC5_s$	$RC6_s$	$RC6_s$
$Camellia_z$	$Camellia_z$	$Camellia_s$		$RC6_z$	$RC6_z$	$RC5_z$	$RC5_z$
$Camellia_s$	$Camellia_s$			$RC5_z$	$RC5_z$	$RC6_z$	$RC6_z$

* Subscript *z* means size-optimised, *s* means speed-optimised.

To conclude our observations, we now discuss the ranking of the ciphers in Table 4.10. Skipjack is the winner in all categories. The fact that RC5 and RC6 have small code size does not help them in their overall ranking, as they range from moderate to poor in all other categories. Rijndael and Twofish fare well if only en/decryption efficiency is taken into account.

MISTY1 is excellent in key agility and data memory usage, and is moderate in code size. If the expanded key is not kept in the RAM and must be generated on the fly, for example when the amount of RAM available to security is scarce, MISTY1 has better en/decryption efficiency than Twofish. KASUMI is an average performer in all categories. Camellia is moderate when it comes to en/decryption efficiency, but does poorly in all other categories.

4.5 Conclusion

First about the operation mode. The OFB mode should be used on pairwise secure links, e.g. for mutual authentication between a base station and a node. The CBC mode should be used for group communications, e.g. in passive participation as mentioned in the previous section.

On the most suitable cipher to use, we will reach our verdict by first ruling out the unlikely candidates. First we would like to emphasise that MSP430F149 is one of the most high-end in the Texas Instrument's MSP430 series. In other words, a total code memory of 59.7 KB and data memory of 2 KB is a hard limit in the MSP430 family of processors. For this reason, we first rule out Camellia which occupies 1/5 of the total code memory even after size optimisation, even though it has decent energy efficiency for en/decryption when speed-optimised. The next to be ruled out are RC5 and RC6, which have poor energy efficiency and key agility (the ability to change keys quickly and with a minimum amount of resources). KASUMI lags behind MISTY1 in all categories, so we should consider MISTY1 instead of KASUMI. Rijndael and Twofish require about the same amount of data memory, but Rijndael has a smaller code size and better en/decryption efficiency, so Rijndael instead of Twofish should be considered. Finally the verdict is given in the form of Figure 4.4.

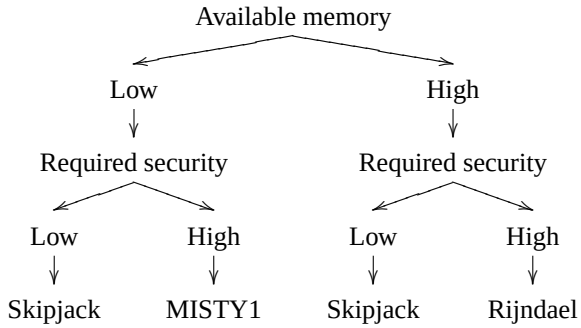


Figure 4.4: Selection of an energy-efficient cipher under the constraints of available memory and required security.

Reviewing some of the proposals in the literature so far, we conclude that there are better options to the use of RC5 and RC6 in WSNs. We note the fact that we use 18 instead of 12 rounds for RC5 based on our security analysis, may explain the difference in performance perceived by us and by other researchers. One discouraging factor against the use of RC5 or RC6 is that most embedded processors do not support the variable-bit rotation instruction like ROL of the Intel architecture [123] which RC5 and RC6 are designed to take advantage of. Another, non-technical, discouraging factor is that they are patented.

In conclusion, we have presented a detailed benchmark for one of the most important cryptographic primitives for WSNs, i.e. block ciphers. Taking into account the security properties, storage- and energy-efficiency of a set of candidates, we have arrived at a system-

atically justifiable selection of block ciphers and operation modes.

Acknowledgement

This work is partially supported by the EU under the IST-2001-34734 EYES project. The authors would like to thank Adrian Perrig and the anonymous reviewers for their inspiring comments which have vastly improved this paper.

Appendix: Per-Cycle Energy Consumption

Different instructions may take different numbers of clock cycles, resulting in different energy consumption per instruction. Even different instructions with the same number of clock cycles may consume different amounts of energy, because of the nature of the instruction itself, for example an instruction that accesses the memory would naturally consume more energy than an instruction that accesses the registers. We will however show that the *energy consumed per cycle* does not vary much from instruction to instruction.

4.6 Methodology

To achieve this, we should ideally measure the energy consumed by each cycle of different instructions. However measuring such energy is difficult without instrumenting the chip, so we measure the current instead. If we fix the voltage V, by measuring the current I, we get the power P. If a cycle is t_c seconds long, and an instruction consists of c cycles, let $e_1, ..., e_c$ be the energy consumed by each cycle, then

$$VI = P = \frac{e_1 + ... + e_c}{ct_c} = \frac{\overline{e}}{t_c} \implies \overline{e} = VIt_c \tag{4.4}$$

where $\overline{e} = \frac{e_1+...+e_c}{c}$ is the average energy consumed per cycle. Since V and t_c are fixed, by measuring I, we are in fact measuring $\overline{e}$. There is a possibility that $e_i \ll \overline{e}$ and $e_j \gg \overline{e}$ for some $i \neq j$ and yet $\overline{e}$ does not vary much from instruction to instruction, meaning that even if $\overline{e}$ is constant, we cannot claim that "energy per cycle" is constant. However this is not a problem, because every instruction is always executed as a whole, with the energy-lean cycle(s) compensating the energy-consuming cycle(s). Worth mentioning is that this method is consistent with Chien and Wen's [56].

The current is measured when an instruction `xxx` is executed in an infinite loop:

```
Mainloop    xxx <some randomised operands>
            xxx <other randomised operands>
            ...100 times
            jmp Mainloop
```

Note that in the above template, one `jmp` instruction after every 100 times of the measured instruction does not affect the measurement much, moreover we can measure the current of `jmp` without the influence of other instructions:

```
Mainloop    jmp Label2
Label2      jmp Mainloop
```

Whenever immediate constant operands, offsets, data are involved, they are randomly generated. In fact, all the test programs are generated by a Perl script.

Since there are 7 addressing modes [280], an instruction like `mov.w` can be used in *at least* 7 modes depending on the type of its operands, e.g. '`mov.w R12, 2(R14)`', '`mov.w @R12+, R14`'

etc. Fortunately not all modes of the same instruction are generated by the compiler. We only test those modes of the instruction generated by the compiler. To find these in-use modes, we have written a Perl script to parse the assembler code of our block cipher algorithms (generated by the compiler). For example, the only mode used for the instruction `and.b` is '`and.b #C,Rn`', and we only measure this particular mode of `and.b` (where '`#C`' stands for an immediate constant, and '`Rn`' stands for a register).

Our test programs are generally divided into 3 parts: (1) the program, (2) the source data, and (3) the destination data. For example, while the instruction '`mov.w @R12, 2(R14)`' itself resides in the program area, '`@R12`' points to a word in the source data area, whereas '`2(R14)`' points to a word in the destination data area. Referring to Table 4.11, $I_{\texttt{RAM}}$ refers to the current when the program, source data and destination data are loaded in the RAM; whereas $I_{\texttt{Flash}}$ refers to the current when the program and the source data are loaded in the Flash *but* the destination data in the RAM. The destination data is always loaded in the RAM because they are meant to be overwritten byte-by-byte, while Flash can only be erased one sector at a time. We do not consider cache because there is none in the processor. Logically $I_{\texttt{RAM}}$ is lower than $I_{\texttt{Flash}}$ since accessing the RAM is cheaper, however we will show that the difference is only about 6% of the mean.

Instead of measuring the current consumed by the processor alone, we have measured the current consumed by the entire EYES sensor node. This is acceptable because the measured instructions do not invoke functions on the peripheral circuits, and assuming the leakage current in the peripheral circuits stays constant independent of the measured instructions.

4.7 Results

There are no entries in Table 4.11 for '`ret`', '`pop.b Rn`' and '`pop.w Rn`'. Instead, '`ret`' is measured along with '`call #L`' or '`call Rn`'; '`pop.b Rn`' along with '`push.b Rn`'; and '`pop.w Rn`' along with '`push.w #C`', '`push.w Rn`' or '`push.w X(Rn)`'. This is fair because in real-world applications, a `pop` is always associated with a `push`, so is a `ret` associated with a `call`.

The average current is 2.93 mA, with a standard deviation of 0.05. From the table, we can see that the most energy-consuming instructions are '`mov.b @Rn+,Rn`' and '`xor.b @Rn+,Rn`', consuming a current of 3.03 mA (when the program and the source data are loaded in the Flash), whereas '`bis.w Rn,Rn`', '`mov.b Rn,Rn`', '`mov.w Rn,Rn`', the rotation instructions, '`swpb Rn`' and '`sxt Rn`' are the cheapest, consuming a current of 2.85 mA (when everything is loaded in the RAM). While it is easy to appreciate why the latter instructions elicit the least current, it is interesting to learn that the instruction mode of '`@Rn+,Rn`' draws a higher current than '`@Rn+,X(Rn)`' (although this does *not* mean that the '`@Rn+,Rn`' over 2 cycles, consumes more energy than '`@Rn+,X(Rn)`' over 5 cycles).

All in all, the difference between the largest and the smallest current is only 6% of the mean, and we conclude that $\overline{e}$ is more or less consistent, or in other words it is safe to assume that "energy per cycle" is more or less consistent for our particular hardware platform.

Table 4.11: Measured currents $I_{\texttt{Flash}}$ and $I_{\texttt{RAM}}$ for each instruction in mA (V=2.994V, t_c=0.22μs, see text for the definition of $I_{\texttt{Flash}}$ and $I_{\texttt{RAM}}$).

Instruction	c	$I_{\texttt{Flash}}$	$I_{\texttt{RAM}}$
add.b #C,Rn	2	2.98	2.89
add.b Rn,Rn	1	2.95	2.87
add.w #C,Rn	2	2.99	2.91
add.w #C,X(Rn)	5	2.99	2.94
add.w Rn,Rn	1	2.96	2.87
addc.w #C,Rn	1	2.99	2.90
addc.w #C,X(Rn)	5	2.97	2.90
addc.w X(Rn),Rn	3	2.99	2.90
addc.w X(Rn),X(Rn)	6	2.99	2.90
and.b #C,Rn	1	2.99	2.89
and.w #C,Rn	1	2.99	2.90
and.w #C,X(Rn)	5	2.97	2.90
and.w @Rn,Rn	2	2.99	2.90
and.w X(Rn),Rn	5	3.00	2.90
bis.w @Rn,Rn	2	2.98	2.89
bis.w Rn,Rn	1	2.93	2.85
bis.w X(Rn),Rn	3	2.98	2.89
bit.b #C,X(Rn)	5	2.96	2.88
bit.w #C,Rn	2	2.98	2.90
bit.w #C,X(Rn)	5	2.96	2.89
br #L	3	2.96	2.87
call #L	5	2.95	2.88
call Rn	4	2.96	2.90
clrc	1	2.97	2.92
cmp.w #C,Rn	2	3.00	2.91
cmp.w #C,X(Rn)	5	2.98	2.90
cmp.w Rn,Rn	1	2.97	2.87
cmp.w Rn,X(Rn)	4	3.00	2.91
cmp.w X(Rn),Rn	3	3.00	2.91
cmp.w X(Rn),X(Rn)	6	2.99	2.90
jc L	2	2.96	2.89
jeq L	2	2.96	2.89
jge L	2	2.97	2.89
jl L	2	2.96	2.89
jmp L	2	2.97	2.89
jnc L	2	2.97	2.89
jne L	2	2.97	2.89
mov.b #C,Rn	2	2.98	2.89
mov.b #C,X(Rn)	5	2.97	2.89
mov.b &L,Rn	3	3.00	2.89
mov.b @Rn,Rn	2	3.00	2.89
mov.b @Rn,X(Rn)	5	3.00	2.90
mov.b @Rn+,Rn	2	3.03	2.91
mov.b @Rn+,X(Rn)	5	2.99	2.91
mov.b Rn,Rn	1	2.94	2.85
mov.b Rn,X(Rn)	4	2.97	2.89
mov.b X(Rn),Rn	3	2.99	2.89
mov.b X(Rn),X(Rn)	6	2.99	2.89
mov.w #C,Rn	2	2.98	2.89
mov.w #C,X(Rn)	5	2.98	2.90
mov.w @Rn,Rn	2	2.99	2.89
mov.w @Rn,X(Rn)	5	2.98	2.90
mov.w Rn,Rn	1	2.93	2.85
mov.w Rn,X(Rn)	4	2.97	2.89
mov.w X(Rn),Rn	3	2.99	2.89
mov.w X(Rn),X(Rn)	6	2.98	2.90
push.b Rn	3	2.99	2.91
push.w #C	4	2.97	2.89
push.w Rn	3	3.00	2.91
push.w X(Rn)	5	2.98	2.90
rla.b Rn	1	2.93	2.85
rla.w Rn	1	2.94	2.85
rlc.b Rn	1	2.94	2.85
rlc.w Rn	1	2.94	2.85
rra.b Rn	1	2.93	2.85
rra.w Rn	1	2.93	2.85
rrc.b Rn	1	2.93	2.85
rrc.w Rn	1	2.93	2.85
sub.b Rn,Rn	1	2.95	2.86
sub.w #C,Rn	2	3.00	2.91
sub.w @Rn,Rn	2	3.02	2.89
sub.w Rn,Rn	1	2.95	2.87
sub.w X(Rn),Rn	3	3.01	2.90
subc.b #C,Rn	2	2.99	2.89
subc.b Rn,Rn	1	2.95	2.86
subc.w Rn,Rn	1	2.95	2.87
subc.w X(Rn),Rn	3	3.00	2.91
swpb Rn	1	2.94	2.85
sxt Rn	1	2.93	2.85
xor.b #C,Rn	2	2.99	2.90
xor.b @Rn,Rn	2	3.00	2.90
xor.b @Rn,X(Rn)	5	2.99	2.91
xor.b @Rn+,Rn	2	3.03	2.91
xor.b Rn,Rn	1	2.94	2.86
xor.b Rn,X(Rn)	4	2.98	2.92
xor.b X(Rn),Rn	3	3.00	2.91
xor.b X(Rn),X(Rn)	6	2.99	2.90
xor.w #C,Rn	2	3.00	2.90
xor.w #C,X(Rn)	5	2.99	2.91
xor.w @Rn,Rn	2	3.00	2.91
xor.w Rn,Rn	1	2.95	2.87
xor.w Rn,X(Rn)	4	3.00	2.91
xor.w X(Rn),Rn	3	3.00	2.90

CHAPTER V *

A Formally Verified Decentralized Key Management Architecture for Wireless Sensor Networks

Abstract We present a decentralized key management architecture for wireless sensor networks, covering the aspects of key deployment, key refreshment and key establishment. Our architecture is based on a clear set of assumptions and guidelines. Balance between security and energy consumption is achieved by partitioning a system into two interoperable *security realms*: the *supervised realm* trades off simplicity and resources for higher security whereas in the *unsupervised realm* the vice versa is true. Key deployment uses minimal key storage while key refreshment is based on the well-studied scheme of Abdalla et al. The keying protocols involved use only symmetric cryptography and have all been verified with our constraint solving-based protocol verification tool CoProVe.
Keywords: Wireless sensor networks, key management, protocol verification.

5.1 Introduction

Wireless sensor networks (WSNs) are open architectures, where any potential intruder can easily intercept, eavesdrop and fake messages. Therefore, to guarantee any level of security (confidentiality, authentication etc.) one has to employ cryptographic protocols.

Key management is the process by which cryptographic keys are generated, stored, protected, transferred, loaded, used, and destroyed. Key management is a challenging problem for WSNs due to the hardware constraints of the sensors and the dynamic nature of WSNs themselves. The hardware constraints of sensors are in terms of:

1. **Cost**: Sensors are generally not tamper-resistant.
2. **Space**: They can only store as many keys as are usually allowed by the storage left over by the operating system and application code, which is not much.
3. **Energy**: It is necessary to optimize the use of cryptography since cryptographic operations tend to be resource-intensive.
4. **Time**: Public-key cryptography should be avoided or at least limited to applications which are not time-constrained, because they are a few order of magnitude more resource *and* time consuming than symmetric-key cryptography [43].

The security and functional requirements of most WSNs are such that under the above constraints, the following guidelines need to be taken into account:

1. NO_SINGLE_KEY: The system should not operate on a system-wide key (or keys). Due to the sensor nodes' lack of tamper-resistance, having a vulnerable system-wide key is *not* better than having no key at all.

*This chapter is a minor revision of the paper with the same title published in the 4th IFIP TC6/WG6.8 International Conference on Personal Wireless Communications (PWC 2003), pages 27–39, Springer-Verlag, ISBN 3-540-20123-8 [1].

2. NO_SINGLE_POINT: The system should not have a single point (e.g. node) of failure.

3. SCALABILITY: The system should be scalable in the sense that the addition of new nodes should not cause excessive rise in computation, communication and administrative overhead in the network.

As of writing, we are not aware of any key management architecture that satisfies all the above hardware constraints as well as the guidelines. Take for example Basagni et al's proposal [24]. In this proposal, sensors are assumed to be tamper-resistant (contrary to our hardware constraints) and share a network-wide system key. Although repeated key refreshment (aka re-keying) would thwart cryptanalytic attampts on the traffic encryption key, the compromise of a single node compromises the system key and thus the whole network, leading to the violation of guideline NO_SINGLE_KEY and NO_SINGLE_POINT. In Perrig et al's proposal [215], the testbed architecture for the SPINS protocol suite possesses a single point of vulnerability that is the base station, violating guideline NO_SINGLE_POINT in addition to SCALABILITY.

The contributions of this paper are:

1. A decentralized key management architecture for WSNs, covering the aspects of key deployment, key refreshment and key establishment. Our decentralized key management architecture, which we call EYES Security Architecture 1 (ESA1), satisfies all the above hardware constraints and architecture guidelines. In addition, authenticated routing is directly implementable on top of the architecture, allowing an integrated security architecture to be specified.

2. Verification of all the protocols employed in our architecture using CoProVe [62]. The verification results imply that our protocols are secure in our standard set of test scenarios.

Overview We start by giving the preliminaries in Section 5.2, which covers the notation that we use, and the overview and security objectives of our architecture. Section 5.3 proceeds to describe key deployment, i.e. the strategic distribution of keys on sensor nodes during the bootstrap phase; and key refreshment, i.e. the refreshment of keys for increasing the encryption threshold. Section 5.4 describes keying protocols, i.e. the establishment of common session keys to secure communication channels; and their verification. Section 5.5 discusses important related work. Section 5.6 concludes and wraps up with a list of future work.

5.2 Preliminaries

Notation To simplify the discussion below, we are using the following standard notation:

- $E_K(M)$ represents the ciphertext as a result of encrypting plaintext M with key K.
- $\text{MAC}_K(M)$ represents the message authentication code (MAC) of message M using key K.
- If $A, B \in \{0,1\}^*$, then $A|B$ is their concatenation.

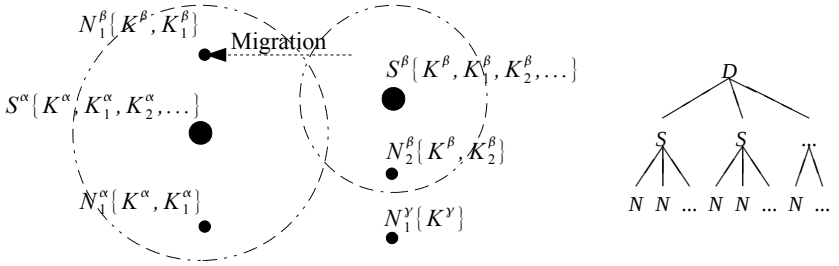
We also use the following non-standard notation:

- $A \rightsquigarrow B$: A's radio range reaches B, or equivalently A's transmission reaches B in a single hop; but not necessarily vice versa.
- $A \leftrightsquigarrow B$: $A \rightsquigarrow B$ and $B \rightsquigarrow A$.

Architecture In ESA1, we employ a structure of supervised and unsupervised clusters. A *supervised* cluster consists of a supervisor node (e.g. S^α of cluster α, S^β of cluster β in Figure 5.1(a)) and a group of supervised nodes (e.g. N_1^α of cluster α, N_1^β and N_2^β of cluster β in Figure 5.1(a)). While the supervised nodes are normal sensor nodes, the supervisor is usually assumed to be tamper-resistant, to have higher energy and computational resources (like the Rich Uncles in [50]), and to have wider radio coverage compared with normal sensor nodes. An unsupervised cluster consists only of unsupervised nodes (e.g. N_1^γ of cluster γ in Figure 5.1(a)), which usually are just normal sensor nodes.

In our architecture, supervisors are needed to combine security (confidentiality and authentication) with energy efficiency and scalability. Nodes that are only sporadically involved in security-critical tasks requiring confidentiality and authentication can be left unsupervised.

Another motivation for having supervisors as specified is to support Di Pietro et al's secure selective exclusion [219] and intrusion detection, the details of which is however outside the scope of this paper. In other words, the above specification of the supervisors is *not* a limitation of our model. We allow as many or as few supervisors as desirable from a cost point of view.



(a) (b)

Figure 5.1: (a) Key deployment in ESA1 (where $N_i^\alpha\{K^\alpha, K_i^\alpha\}$ means node N_i^α of cluster α with keys K^α, K_i^α; dashed circles represent supervisors' radio range). (b) Three-tier hierarchy for the supervised realm: D=Domain, S=Supervisor, N=Node.

Security Objectives First, we define two *security realms*: the set of all supervised clusters constitute the supervised realm, while the set of all unsupervised clusters constitute the unsupervised realm.

- In the supervised realm, confidential and authenticated *node-to-node* communication within a single supervised cluster, and between nodes residing in different supervised clusters are realized through *intra-supervised cluster keying* (cf Section 5.4.2) and respectively *inter-supervised cluster keying* (cf Section 5.4.3). In these modes of communication, the effect of any node's compromise is isolated, i.e. if a supervised node N_1^α is compromised, only sessions that involve N_1^α are compromised.
- In the unsupervised realm, confidential and authenticated *group* communication within a single unsupervised cluster is realized through *intra-unsupervised cluster keying* (cf

Section 5.4.4). In this mode of communication, the compromise of a node compromises the security of the whole cluster. Between unsupervised clusters, communication is open and unauthenticated, i.e. no two unsupervised clusters trust each other.

- Between the supervised realm and the unsupervised realm, confidential and authenticated communication is realized through *unsupervised-to-supervised cluster keying* (cf Section 5.4.5). Since the supervised realm has a higher security level than the unsupervised realm, we do not think it makes sense to secure a communication that originates from a supervised node to an unsupervised node (drawing an analogy from the "no write-down" policy of the Bell LaPadula model). However the need for unsupervised-to-supervised node communication is believed to be quite possible, e.g. in the upload of sensitive data.

Independent of the confidentiality and authentication requirements, message integrity can and should always be ensured by using universal hash functions [51]. Universal hash functions are highly resistant to collisions and have also been recently shown to be resistant to algorithmic complexity-based denial of services attacks [68].

5.3 Key Deployment

We can now describe the main keys employed in ESA1 and their roles.

In ESA1, every supervised node N_i^α ($1 \leq i \leq n$) of cluster α is equipped with a unique *node key* K_i^α and the supervisor S^α is equipped with the node keys of all the supervised nodes of its cluster. Additionally $S^\alpha, N_1^\alpha, \ldots, N_n^\alpha$ share the same *cluster key* K^α. The cluster key is a backup key that is only used when the supervisor is not available, in which case the supervised cluster downgrades to an unsupervised cluster. Therefore in contrast to what Figure 5.1(a) might suggest, the cluster membership of a node N in cluster α depends solely on (1) whether there is a unique shared key between N and S^α; and (2) whether N possesses the cluster key K^α. When a node, say N_1^β (cf Figure 5.1(a)), roams out of the radio range of its original supervisor S^β, to the radio range of another supervisor S^α, it will for efficiency initiate a process called *migration* that migrates its node key from S^β to S^α (cf Section 5.4.6). It is important to note that when we mention "migrating a key", we actually mean "migrating a derivation of the key", so that even if the migrated key is compromised backward secrecy is preserved.

In an unsupervised cluster γ, every node shares the same cluster key K^γ, and cluster membership depends solely on the possession of K^γ.

To establish trust between supervisors, we leave two options open. The first option involves a further domain tier higher in the hierarchy (cf Figure 5.1(b)), and it is to connect all the supervisors at bootstrap to a *domain controller*. The domain controller partitions the set of supervisors into *domains*. Only supervisors that belong to the same domain are given the same *domain key* (a symmetric key) and thus share the common trust. This scheme uses only symmetric cryptography but can only support a static and coarse notion of trust – the compromise of any supervisor compromises the whole domain. The second option is to resort to SSL/TLS (or similar public key-based mechanisms) to secure communication between supervisors. This scheme supports flexible and fine-grained access control at the expense of higher overhead and energy consumption. All in all, the suitability of each of these two options depends on the construction of the supervisors and the actual application scenario. We focus on the node/cluster tiers of the hierarchy.

5.3.1 Key Refreshment

The effectiveness of a key is not only bounded by its length but also by the number of encryptions it has been used for, because with enough encryptions an intruder is able to launch birthday attacks (or cryptanalytic attacks which however typically require more encryptions [8]). Therefore, as part of any key management architecture, we need a scheme to refresh every now and then the deployed keys. For this, we refer to the well-studied scheme of Abdalla et al [8]. Given an initial shared key K_0 between two nodes A and B, the encryption and message authentication code (MAC) keys are derived *serially* at each i-th re-key as follows:

$$K_{enc,i} = \mathrm{MAC}_{K_i}(1) \qquad \text{and} \qquad K_{\mathrm{MAC},i} = \mathrm{MAC}_{K_i}(2) \tag{5.1}$$

where $K_{i+1} = \mathrm{MAC}_{K_i}(0)$ and $i = 0, 1, 2, \ldots$. A convention is that a pair of encryption key and MAC key derived from key K is written as K' and K''.

In a supervised cluster, re-keying is message-count driven, i.e. when the encrypted message count approaches the encryption threshold (e.g. $2^{2k/3}$ for CBC mode, where k is the key length [8]), re-keying is initiated. In a supervised cluster, if the shared key is between a node and the superviser, or between a node and another node, re-keying can be initiated by either end of the secure channel. In an unsupervised cluster, re-keying of the cluster key can proceed in a similar fashion described by Basagni et al [24].

5.4 Verified Keying

In our end-to-end communication model, session establishment has two phases:

1. Keying: The initiator engages the node it wants to communicate with in a key establishment (keying for short) protocol, using an underlying, not necessarily secure routing protocol.

2. Communication: Then, by using the session key established, the two endpoints start communicating with authenticated and optionally encrypted messages.

The following discusses the keying protocols and their formal verification.

5.4.1 Formal Protocol Verification

The keying protocols described in this section are verified using the protocol verifier CoProVe (available at `http://wwwes.cs.utwente.nl/24cqet/coprove.html` [62]). The verifier works by taking as input a specification of the protocol and a *system scenario* describing the roles involved in the protocol, e.g. the initiator, the responder or a server of the protocol. The system scenario is analyzed by the verifier in search of possible interleavings of the roles. Attacks manifest as interleavings not anticipated by the protocol designer.

CoProVe is used to verify these properties:

- Confidentiality: A session key must only be known to the communicating nodes, and the supervisors involved in the keying protocol. This is how the verification works. In CoProVe, we 'encode' the desired security property in the system scenario, along with the protocol roles. For confidentiality, this is done by creating a special 'secrecy' role, which attempts to *receive* the secret we want to guard. If there is no corresponding role in the specification that sends the secret, but the 'secrecy' role finishes (i.e. is able to receive the secret), then we can deduce that it can only be the intruder who sent it, and therefore the intruder *knows* the secret.

- Authentication: A keying protocol must end with every party properly authenticating the other parties it is communicating with. In other words, it must be impossible for any intruder *I* to impersonate another node *A* whose keys (used in the keying protocol) *I* does not have. To verify authentication, we fix some role, say a responder, but we exclude the corresponding initiator. If the responder is able to run to completion, we can regard the protocol as having authentication failure, since there exists the possiblity of an intruder impersonating the initiator, thereby finishing the protocol with the responder.

- Replay resistance: The meaning of replay attack on a role *R* is the possibility of unauthenticated parties to cause *R* to run, i.e. for *R* to process replayed messages. If *R* happens to maintain state of every run, then it would be maintaining incorrect states. In CoProVe, we verify replay resistance of *R* by making sure that two instances of *R* *cannot* complete regardless of the other roles.

After verifying these properties, i.e. confidentiality, authentication and replay resistance, we can be sure that the following network-based attacks described in Section 1.3.1 would not succeed:

- Intercepting messages: An attacker that does not have the key shared between *A* and *B* cannot eavesdrop on the messages exchanged between *A* and *B*.

- Modifying existing messages: An attacker that does not have the key shared between *A* and *B* cannot modify the messages exchanged between *A* and *B* without being detected.

- Fabricating false messages: An attacker that does not have the key shared between *A* and *B* cannot fabricate messages to *A* or *B* because the fabricated messages would fail to be authenticated.

- Replaying existing messages: An attacker that does not have the key shared between *A* and *B* can replay messages previously sent by *A* or *B* but it would not be able to finish any protocol with either *A* or *B*.

For the following attacks however, key management is generally not (meant to be) a solution:

- Interrupting messages: Any attacker can jam the network as long as it has a radio transmitter and knows which frequency or frequencies to transmit on.

- Deviating from protocol: Any attacker that has become an insider of the network, can deviate arbitrarily from the protocol.

Limitations To ensure termination, system scenarios must be finite – a restriction shared by every model-checking-based approach. (Other approaches like theorem proving allow an unbounded number of sessions, but sacrifice termination guarantee, or need human intervention.) As such, a protocol is verified secure only in the finite tested scenarios. There is no assurance that an attack would not actually be found in a larger scenarios.

So far, we have tested all protocols in the context of a scenario containing a single session. Such tests turned out to be extremely useful, as they have allowed us for example to spot authentication failures resulting from the inappropriate use of unauthenticated nonces. Here, we also want to stress that these protocols are considerably more complex than many of the

usual "toy" security protocols protocol verifiers are tested agains. In fact, considering that one of them involves nine message exchanges, we were ourselves pleasantly surprised to see that CoProVe could deal with them within a reasonable time-span (ranging from a fraction of a second to 2.5 days for the most complex protocol, on a standard Linux-i686 architecture). We are now carrying out extra tests involving larger scenarios.

5.4.2 Intra-Supervised Cluster Keying

We start with the keying protocol that is used within a supervised cluster. Concerning the notation, N_A represents a nonce emitted by A, N_B represents a nonce emitted by B and so on; strings such as Ack and Success are constants. For the sake of notation simplicity, we leave out the superscripts we used in Figure 5.1. In addition, recall that the encryption key and the MAC key derived from key K are respectively denoted by K' and K''.

Suppose that in a cluster supervised by S, a node A (which shares key K_A with S) wants to initiate a session with B (which shares key K_B with S). The protocol is:

Protocol 1

1. $A \rightarrow S$: $N_A, B, \mathrm{MAC}_{K''_A}(N_A|B)$
2. $S \rightarrow A$: $E_{K'_A}(E_{K'_B}(N_S)|K_{AB}), \mathrm{MAC}_{K''_A}(N_A|B|E_{K'_B}(N_S)|K_{AB})$
3. $A \rightarrow B$: $A, E_{K'_B}(N_S)$
4. $B \rightarrow S$: $B, N_B, A, \mathrm{MAC}_{K''_B}(N_S|B|N_B|A)$
5. $S \rightarrow B$: $E_{K'_B}(K_{AB}), \mathrm{MAC}_{K''_B}(N_B|A|E_{K'_B}(K_{AB}))$
6. $B \rightarrow A$: $\mathrm{Ack}, \mathrm{MAC}_{K''_{AB}}(\mathrm{Ack})$

where K_{AB} is the final established session key. This protocol has been verified (1) secure in the confidentiality of K_{AB}, (2) secure in the mutual authentication between A and B, and (3) secure against replay attacks on S. Before we proceed to describe this protocol, we would like to state that an alternative keying protocol *could have been* derived from the node-to-node key agreement protocol of SPINS [215]:

Protocol 2

1. $A \rightarrow B$: N_A, A
2. $B \rightarrow S$: $N_A, N_B, A, B, \mathrm{MAC}_{K''_B}(N_A|N_B|A|B)$
3. $S \rightarrow A$: $E_{K'_A}(K_{AB}), \mathrm{MAC}_{K''_A}(N_A|B|E_{K'_A}(K_{AB}))$
4. $S \rightarrow B$: $E_{K'_B}(K_{AB}), \mathrm{MAC}_{K''_B}(N_B|A|E_{K'_B}(K_{AB}))$
5. $A \rightarrow B$: $\mathrm{Ack}, \mathrm{MAC}_{K''_{AB}}(\mathrm{Ack})$

Here, the first four messages are from SPINS and we added the fifth message to prevent authentication failure. This protocol has been verified (1) secure in the confidentiality of K_{AB}, (2) secure in the mutual authentication between A and B, and (3) secure against replay attacks on S. The drawback of this protocol is that while there is no attack, it is far too easy for an intruder to generate random bogus requests (at step 1) and hence random bogus protocol instances, that would cause B to wait at step 5 for the bogus protocol instances to finish.

A design principle behind Protocol 1 is that A should do more work compared with B to initiate the protocol. Now it takes three messages for A to get B to respond, and S is able tell B whether A's request is genuine at step 5. Let's also have a look at the energy budget.

We only consider the energy consumed in radio communication, since it generally dominates that used for computation. Assuming the following sizes: node names (IDs) and nonces are 64 bits long, MACs are 160 bits, keys are 128 bits, acknowledgements are 8 bits (to cater for other status codes), encryption is length-preserving; and assuming the energy consumption for receiving to be 40% that of sending [243], we found the energy consumption of Protocol 1 compared with Protocol 2 to be 52% higher for A, 6% for B and 21% for S. The fact that A has to consume more energy in Protocol 1 cannot be helped by our requirement that A authenticates itself with S first, but this is necessary to overcome the drawback of Protocol 2. Moreover, the energy cost can be amortized over A's period of operation, provided such keying is conducted infrequently.

Special attention is given to the MAC at step 6 of Protocol 1 because MAC'ed acknowledgements are susceptible to replays depending on the MAC mode used. By assuming that combined CTR and CBC mode (CTR + CBC-MAC) [129] is used, we should be able to minimize this risk. This also applies to other instances of MAC'ed acknowledgement in the other protocols in the rest of this paper.

5.4.3 Inter-Supervised Cluster Keying

After solving the problem of keying two nodes in the same cluster, let us consider keying two nodes from two different clusters in the supervised realm. The protocol is:

Protocol 3

1. $A \rightarrow S_A$: $N_A, B, \mathrm{MAC}_{K''_A}(N_A|B)$
2. $S_A \rightarrow A$: $E_{K'_{SS}}(N_{S_A}), \mathrm{MAC}_{K''_A}(N_A|B|E_{K'_{SS}}(N_{S_A}))$
3. $A \rightarrow B$: $A, B, E_{K'_{SS}}(N_{S_A})$
4. $B \rightarrow S_B$: $B, N_B, A, E_{K'_{SS}}(N_{S_A}), \mathrm{MAC}_{K''_B}(B|N_B|A|E_{K'_{SS}}(N_{S_A}))$
5. $S_B \rightarrow S_A$: $N_{S_B}, A, B, E_{K'_{SS}}(K_{AB2}), \mathrm{MAC}_{K''_{SS}}(N_{S_A}|N_{S_B}|A|B|E_{K'_{SS}}(K_{AB2}))$
6. $S_A \rightarrow S_B$: $E_{K'_{SS}}(K_{AB1}), \mathrm{MAC}_{K''_{SS}}(N_{S_B}|A|B|E_{K''SS}(K_{AB1}))$
7. $S_A \rightarrow A$: $E_{K'_A}(K_{AB}), \mathrm{MAC}_{K''_A}(N_A|B|E_{K'_A}(K_{AB}))$
8. $S_B \rightarrow B$: $E_{K'_B}(K_{AB}), \mathrm{MAC}_{K''_B}(N_B|A|E_{K'_B}(K_{AB}))$
9. $B \rightarrow A$: $\mathrm{Ack}, \mathrm{MAC}_{K_{AB}}(\mathrm{Ack})$

where K_{SS} is intuitively the shared key between S_A and S_B; and $K_{AB} = K_{AB1}|K_{AB2}$ is the final established session key. To divide the cost of generating the whole key, the key is generated as a concatenation of two sub-keys K_{AB1}, K_{AB2} respectively produced by S_A and S_B. Note that A does not need to know who B's supervisor is – that is S_A's (and B's) job. This protocol has been verified (1) secure in the confidentiality of K_{AB}, (2) secure in the mutual authentication between A and B, and (3) secure against replay attacks on S_A and S_B.

5.4.4 Intra-Unsupervised Cluster Keying

When two nodes in an unsupervised cluster want to communicate with each other, the only key they can use is the cluster key. To state the obvious, the cluster key is never used directly, instead, for each session a new pair of encryption and MAC keys are derived from the cluster key using the mechanism of Section 5.3.1.

5.4.5 Unsupervised-to-Supervised Node Keying

As mentioned, there is by default no trust between any supervised cluster and unsupervised cluster, i.e. there is no *direct* shared key an unsupervised node can use to communicate with a supervised node. It is here that the idea of *visitor certificate* is employed. A visitor certificate takes the form $E_{K_g}(ID|K_v)$, where K_g is the *grant key*, K_v is the *visitor key* and ID is the visitor's name. K_g is held by a supervisor while K_v and the respective visitor certificate are held by an unsupervised node.

To start a session, the unsupervised node, say U (with visitor key K_v), submits its visitor certificate $E_{K_g}(U|K_v)$ to the supervisor, say S (with grant key K_g). Upon reception, S decrypts U's certificate with K_g to obtain K_v, and sends a new session key K_{US} to U, thereby establishing a secure session with U:

Protocol 4

1. $U \rightarrow S$: $N_U, S, E_{K_g}(U|K_v), \mathrm{MAC}_{K_v''}(N_U|S|E_{K_g}(U|K_v))$
2. $S \rightarrow U$: $N_S, E_{K_v'}(K_{US}), \mathrm{MAC}_{K_v''}(N_U|N_S|E_{K_v'}(K_{US}))$
3. $U \rightarrow S$: $\mathrm{Ack}, \mathrm{MAC}_{K_{US}''}(N_S|\mathrm{Ack})$

Protocol 4 has been verified (1) secure in the secrecy of K_{US}, and (2) secure in the mutual authentication between U and S. From this point onwards, the keying protocol for the communication between U and any node supervised by S, let's say A will be similar to the intra-supervised cluster keying between U and A (cf Section 5.4.2).

The merit of this approach is that K_g and K_v can be pre-deployed, while the visitor certificate can be generated and distributed at runtime by a *visit granting agent* (VGA). A VGA keeps an application-determined set of visitor and grant keys. Under guideline NO_SINGLE_POINT (cf Section 7.1), no single VGA should possess all the visitor and grant keys of the whole network, therefore there should be several VGAs throughout the network, each safeguarding a preferrably non-intersecting set of visitor and grant keys. A VGA works with supervisors. Upon an unsupervised node U's request for a visitor certificate to visit supervisor S, a VGA

1. checks with S and U's neighbouring supervisors if U is a green node (i.e. a good node [219])
2. checks with itself if it has U's K_v and S's K_g

If all checks are positive, then the VGA will proceed to issue the requested visitor certificate to A. Otherwise the request would be denied. In this scheme, the only pre-deployed information is the visitor keys on the unsupervised nodes and grant keys on the supervisors, instead of (x = number of supervisors $\times$ number of unsupervised nodes) key pairs. Note that it is straightforward to extend the visitor certificate to include the expiry time of the certificate.

5.4.6 Node Migration

Intra-supervised cluster keying and inter-supervised cluster keying work most efficiently when $S_A \rightsquigarrow A$. If it happens that $S_A \not\rightsquigarrow A$ as a result of node mobility or other factors that break the radio link, A may 'employ' another supervisor, say S_B as its temporary supervisor if $S_B \rightsquigarrow A$, by migrating its node key from S_A to S_B, provided S_A and S_B have mutual trust. If A is not able to find such a supervisor, then no migration takes place. Note that A *never* becomes a member of S_B's cluster (which implies receiving the cluster key from S_B),

because then a compromised node would be able to roam to every cluster and collect their cluster key.

Suppose A migrates from the cluster supervised by S_A to the cluster supervised by S_B, the security objective of this protocol is to prevent unauthorized migrations, i.e. if A did not authorize the migration, the migration should not occur. The protocol is:

Protocol 5

1. $A \rightarrow S_A$: $N_A, S_B, \mathrm{MAC}_{K''_A}(N_A|S_B)$
2. $S_A \rightarrow A$: $E_{K'_A}(E_{K'_{SS}}(N_{S_A})), \mathrm{MAC}_{K''_A}(N_A|S_B|E_{K'_{SS}}(N_{S_A}))$
3. $A \rightarrow S_B$: $A, S_A, E_{K'_{SS}}(N_{S_A})$
4. $S_B \rightarrow S_A$: $S_B, N_{S_B}, A, \mathrm{MAC}_{K''_{SS}}(N_{S_A}|S_B|N_{S_B}|A)$
5. $S_A \rightarrow S_B$: $E_{K''_{SS}}(K_{A2}), \mathrm{MAC}_{K''_{SS}}(N_{S_B}|A|E_{K''_{SS}}(K_{A2}))$
6. $S_B \rightarrow A$: $\mathrm{Sucess}, \mathrm{MAC}_{K''_{A2}}(\mathrm{Sucess})$

where $K_{A2} = \mathrm{MAC}_{K_A}(0)$ is the migrated key. Note that as mentioned, it is a derivation of A's node key that is migrated. This protocol has been verified (1) secure in the confidentiality of K_{A2}, (2) secure in the authentication of A to S_A and S_B, of S_A to A and S_B, of S_B to A and S_B, and (3) secure against replay attacks on S_A.

Suppose A migrates to S_C's cluster, if S_A, S_B and S_C trust each other, but $S_A \not\rightsquigarrow S_C$ and $S_B \rightsquigarrow S_C$, then S_B should for efficiency take over S_A's role of migrating A's key to S_C.

Supervisors like S_B and S_C are not intended to keep A's key when A leaves their cluster, however if A consistently returns, then it would not be efficient to purge A's key immediately once A leaves. It is best to have a mechanism for predicting A's movement so that A's key can be optimally cached, otherwise there should at least be a hysteresis period where A's key is cached. Note that S_A *never* purges A's key. This is to ensure that there is always one supervisor holding A's key.

5.5 Related Work

There is currently a lack of literature detailing key management architectures for WSNs. We mentioned Basagni et al's pebblenets and Perrig et al's SPINS in Section 5.1. We have also made a comparison between our intra-supervised cluster keying protocol with SPINS' node-to-node key agreement protocol in Section 5.4.2. Now we look at some other proposals.

Slijepcevic et al propose location-based keys [266]. All nodes start out sharing a set of master keys, and the network is divided into non-overlapping hexagonal cells. At runtime, a node selects and activates a key from the set of master keys based on the output of its pseudorandom generator *and* on which hexagonal cell the node is in, so that the compromise of a key in one cell does not affect the other cells. Seen another way, the problem of session establishment reduces to the problem of determining the location; and node mobility in this scheme does not necessitate the equivalent of migration in our proposal, because it is readily 'solved' by having all nodes sharing the same keys. However, this scheme does not fit our requirements, because of our assumption that sensor nodes are not tamper-resistant – the set of master keys, the corresponding pseudorandom number generator and the seed are easily exposed and hence the active key of any cell can be found out. The requirement of nodes being able to discover their coordinates is also too demanding for our case.

Luo et al's localized trust model [168] is not exactly a key management architecture proposal, but it can be regarded as such. This model is based on (k, n)-threshold secret sharing,

wherein a system secret key (of a public-private key pair) is shared among every n nodes, and a node is only trusted if it acquires a certificate issued by k neighbours. Session establishment is presumed to proceed through Diffie-Hellman-derived protocols. This solution forgoes the need for imposing structure (e.g. clustering) on the network, but it also means that if a node cannot get certified by k neighbours, it would have to roam to another place where it can. More importantly, the extensive use of public-key cryptography is also too demanding within the resource constraints of WSNs.

5.6 Conclusion and Future Work

We have presented a decentralized key management architecture for WSNs, covering the aspects of key deployment, key refreshment and key establishment.

Motivated by the assumption that sensors are not tamper-resistant, we have devised an architecture in which two security realms namely the supervised realm and the unsupervised realm provide two levels of security. Communication in the supervised realm entails various protocols to implement intra- and inter- supervised cluster keying, which we have verified with our tool CoProVe. In the unsupervsied realm, security suffers from the fact that a cluster key is shared by all members of the same cluster. The trade-off provides WSN implementors with an option to partition their system according to the actual security requirements, and the network can be expanded by easily adding one node at a time, one cluster at a time, or even one domain at a time. To conclude, our architecture meets our hardware constraints, and the guidelines NO_SINGLE_KEY, NO_SINGLE_POINT and SCALABILITY of Section 5.1.

We have adopted a two-step design process by getting the protocols right first before proceeding to optimize them. Our next step is to validate our model through simulation and implementation. We are also motivated to extend our architecture to support directed diffusion (which implies multicast communication) [119]. As of now, our assumption is that a node knows exactly the name of the other node it wants to communicate with, which is not the case in directed diffusion.

5.7 Acknowledgement

The authors would like to thank Roberto di Pietro, Paul Havinga and the anonymous reviewers for their insightful comments.

CHAPTER VI *

LKHW: A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks

Abstract We present a mechanism for securing group communications in Wireless Sensor Networks (WSN). First, we derive an extension of Logical Key Hierarchy (LKH). Then we merge the extension with directed diffusion. The resulting protocol, LKHW, combines the advantages of both LKH and directed diffusion: robustness in routing, and security from the tried and tested concepts of secure multicast. In particular, LKHW enforces both backward and forward secrecy, while incurring an energy cost that scales roughly logarithmically with the group size. This is the first security protocol that leverages directed diffusion, and we show how directed diffusion can be extended to incorporate security in an efficient manner.

6.1 Introduction

Imagine sensor networks being deployed in public areas to detect SARS viruses. The sensors spend most of their time in a dormant state and only report their measurements when requested. Health officers gather readings from the network by posting requests such as "start sending me your readings five times a minute if you are currently detecting the virus". Assuming the network is heterogeneous (in that apart from SARS virus sensors, there are other types of sensors in the network), there are generally two ways of executing this request: (1) remember the IDs/names of the virus sensors and send the request explicitly to the sensors using an ID-based routing protocol (like AODV [212] or DSR [125]), or (2) never remember any ID, and instead, flood the network with the request.

Method (1) is most effective when the number of SARS sensors is fixed and known beforehand, and sensors are static (so that a new route does not need to be set up everytime a send is requested). However these conditions may not be practical, because these sensors exist in large quantity; the bookkeeping overhead would be unmanageable if at the same time we allow arbitrary addition (or removal) of sensors to (or from) the network. On the other hand, if all the SARS sensors share a common ID/address, ID-based routing does not work. We would not go as far as suggesting setting up directory services for sensor lookup, as the problem of keeping these directory services up-to-date and the problem of locating these directory services only elevate the original problem to a higher level.

Method (2) may seem inefficient, however with the help of *directed diffusion* [119] that facilitates *attribute-based naming* and *in-network processing*, significant traffic reduction is possible (Heidemann et al. report a reduction of 42% [108]). In a nutshell, the energy expended in flooding the network with the request, is (more than) compensated by savings obtained from the exploitation of *local interaction* and *caching*. Furthermore, with cached information, after the initial flood, further flooding is often not necessary.

All is well until the health officers require that the communication between the requesting

*This chapter has been published in the 32rd International Conference on Parallel Processing Workshops (ICPPW 2003), pages 397–406, IEEE Computer Society Press, ISBN 0-7695-2018-9 [6].

device (i.e. sink) and the SARS sensors (i.e. sources) to be secure. In other words, directed diffusion does not cater for secure group communication. By secure we mean just three aspects: (1) data confidentiality, (2) data integrity, and (3) data authentication. And by group communication, we mean a secure communication channel *shared* by the sources and the sink – the naïve approach of using pairwise secure channels between each individual source and the sink rules out in-network processing, which is essential for the success of directed diffusion. However our communication model is *specific* for WSN in that messages flow in a certain direction: interest (or query) messages from sinks to sources, and data messages from sources to sinks. We do not cater for general *n*-to-*n* communication, which is not common for WSN.

We propose LKHW (Logical Key Hierarchy for Wireless sensor networks), a secure group communication scheme based on directed diffusion and LKH. Our contribution is two-fold:

1. **Integration of security and routing**: Our scheme integrates security and routing in a single framework, by leveraging secure multicast techniques and the tried and tested concepts of directed diffusion. Such integration allows our security protocols to be optimized for directed diffusion in terms of energy efficiency, reducing the overhead that would otherwise be necessitated by a routing-unaware alternative.

2. **Efficiency**: We present a performance evaluation model that, unlike the conventional evaluation model for secure multicast schemes, is centered around energy consumption and that takes into account the dynamic nature of the topology of WSN. And with the model, we show that the energy efficiency of LKHW scales roughly logarithmically with the group size.

This paper is organized as follows. Section 6.2 introduces directed diffusion. Section 6.3 contains the essentials of LKHW. It starts by introducing LKH, then proceeds to discuss the initialization aspects of LKHW, and the central problems of user-leave and user-join operations. We back up our theories with performance evaluation in Section 6.4 , while we discuss related work in Section 6.5. Finally, Section 6.6 gives the conclusion and some ideas for future work.

6.2 Overview of Directed Diffusion

Basic Terminology Picture the classic directed diffusion scenario in which a WSN is deployed in a wilderness refuge to track animals [108, 119]. A tracking request represents an *interest*. The node that broadcasts the initial *exploratory* interest is the *sink*, i.e. the final destination of the requested data. In directed diffusion, the adjective "exploratory" indicates unoptimized flow, and that the flow will cease if it is not reinforced. Interest and data are *named* using attribute-operation-value tuples. (i.e. `attr1 op1 val1; attr2 op2 val2`...). For example, an interest *I* might look like "`class IS interest; x GT 0`", where `GT` is a *formal operator* meaning "greater than"; a data *D* might look like "`class IS data; x IS 1`", where `IS` is an *actual operator* meaning "equals". Obviously this particlar data *D matches* this particular interest *I* because *D*'s value '1' is greater than *I*'s value '0' as required by *I*'s formal operator. Other formal operators are described by Intanagonwiwat et al. [119]. Every node tries to match every data message it receives with every interest in its interest cache, and if a match is found, the data message will be sent to whoever originated the matching interest. It is important to note that *an interest does not contain any attribute-operation-value tuple*

that describes the sink that originates it, nor does a data describe the source that produces it. An "interest about interests" is just a nested interest, e.g. an interest about interests in four-legged animals. The restriction is that interests cannot be further nested, i.e. there is no "interest about interest about interests" and so on. The concept of "interest about interests" is used extensively in LKHW, as shall be seen later.

Directed diffusion consists of three phases:

1. **Interest diffusion**: As the interest is *diffused* across the network, every sensor that receives the interest remembers the neighbour(s) from which the interest came, using their *interest cache* (which are essential for suppressing duplicate messages and preventing loops), before re-broadcasting the interest to all their neighbours. Moreover, every sensor node is *task-aware* and any node that matches the traveling interest will apart from forwarding the interest, reply with the relevant data and thus become a *source*. The traveling interest sets up *gradients* along the paths it has taken. A direction is *downstream* if it is from a source to the sink, and *upstream* if otherwise.

2. **Exploratory reply**: The first replies from the sources are exploratory. These replies, containing data, flow downstream along the gradients set up by the interest. Along the gradients, each node caches the data message in their *data cache* (which, similar to the interest cache, is used to suppress duplicate messages and to prevent loops). The fact that movements through the gradients are actually merely *time progressions of data-interest matching events along the communication links* cannot be stressed enough.

3. **Gradient reinforcement**: On receiving the exploratory data messages, the sink stores them in its data cache, and according to some system-defined parameters (e.g. latency), the sink *reinforces* its neighbours which satisfy these parameters (e.g. neighbours which delivered the exploratory data messages with the lowest latency). These neighbours in turn reinforce their upper-stream neighbours, according to the same principle. The effect of reinforcement is system-defined, e.g. possibly bumping up the data output rate of the sources connected to the reinforced gradients, and hence the data transfer rate along the reinforced gradients. Note that the meaning of "x reinforces y" is equivalent to the meaning of "x reinforces the downstream gradient from y to x". Future data messages from the sources only travel down these reinforced gradients, and with priority given to the more heavily reinforced gradients. That said, gradients can also be *negatively reinforced* for increasing performance, for load-balancing, or for gradient maintenance in response to topological changes, node failures, environmental effects etc.

All in all, the directed diffusion model provides the basic primitives for data communications in WSN. It uses caches for data-interest matching, to suppress duplicate messages and to prevent loops. It uses data aggregation to optimize bandwidth usage. As a result of performing only local interaction, nodes require little local storage and the resulting network is capable of self-repairing. However this also implies a trade-off for robustness and scalability with energy efficiency.

6.3 The LKHW Model

After an overview of directed diffusion, we describe LKHW in this section. There are two basic aspects to a tree-based multicast model like LKHW: (1) the key tree structure, and (2) the

re-keying scheme. As the name implies, LKHW adopts the key tree structure of Logical Key Hierarchy (LKH) [292]. The re-keying scheme of LKHW is based on Wong et al's group-oriented re-keying scheme [299] and an improvement of ours (described in Section 6.3.3). In Section 6.3.1, we describe the key tree structure. We then proceed to describe our strategy for group initialization in Section 6.3.2. Group dynamics and the associated aspects of re-keying are detailed in Section 6.3.3.

6.3.1 Key Tree

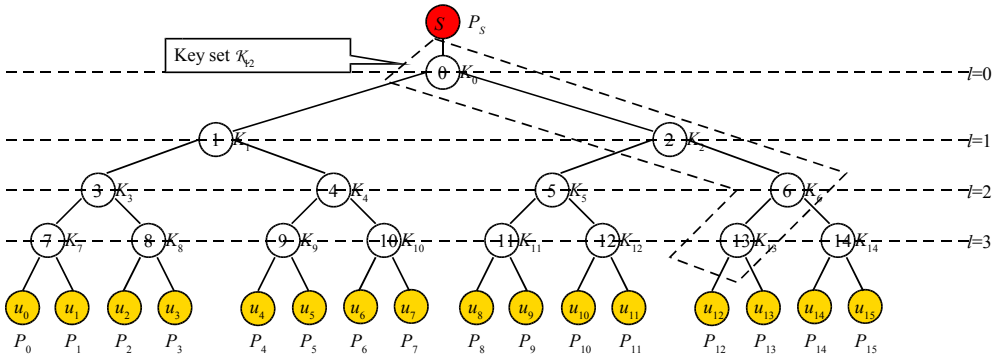


Figure 6.1: Logical key hierarchy ($a = 2, n = 16$).

n	Number of users in a multicast group
a	Degree or a-rity of a LKH tree
h	Height of a LKH tree, i.e. $\lceil log_a n \rceil$
$\mathscr{L}$	The set of hierarchical levels of a LKH tree, i.e. $\mathscr{L} = \{0, \ldots, h-1\}$
K'	Refreshed version of a key K
$E_K(M)$	Encryption function that takes key K and plaintext M
$\text{MAC}_K(M)$	Message Authentication Code (MAC) function that takes key K and plaintext M

Table 6.1: Notation.

In LKH, (symmetric) keys, e.g. $K_0, \ldots, K_{15}$, $P_0, \ldots, P_{15}$ in Figure 6.1, are logically distributed in a tree rooted at the *key distribution center* (KDC). The leaves of the tree correspond to the users, e.g. $u_0, \ldots, u_{15}$ in Figure 6.1. By 'user', we mean a user process on a sensor node. Every user stores all the keys on its *key path*, i.e. the path from the leaf node (corresponding to the user) up to the root. These keys comprise the user's *key set*,

$$\mathscr{K}_i = \{K_j \mid j = S_a(l) + \left\lfloor \frac{i}{a^{h-l}} \right\rfloor, \forall l \in \mathscr{L}\} \tag{6.1}$$

where i is the index of user u_i; keys of the form K_j are as illustrated in Figure 6.1; a, n, $\mathscr{L}$ are as defined in Table 6.1; and $S_a(l)$ is the sum of the first l terms of a geometric progression with ratio a.

For an example using Equation 6.1, u_0 stores key P_0, K_7, K_3, K_1, K_0, where in particular P_0 is the unique *individual key* u_0 shares with the KDC; K_0 is the *group key* that is shared by all users in the group and is used to encrypt all group communication traffic; and K_7, K_3, K_1

are so-called *key encryption keys* which serve the sole purpose of encrypting new keys during re-keying (cf Section 6.3.3). The KDC maintains the structure of the key tree and stores all the keys in the key tree.

Note that the tree illustrated here is for simplicity binary and balanced, but in reality key trees need not be. Therefore in general, if we use the notation in Table 6.1, then the total number of stored keys per user is $h + 1$, and the total number of stored keys by the KDC is $\frac{a^{h+1}-1}{a-1}$, or $\frac{an-1}{a-1}$ if the tree is balanced.

The main reason for using such a key tree compared with more traditional structure-less approaches such as Blundo et al's [35] is that *re-keying* can be more efficiently executed. Re-keying is the operation that refreshes a subset of the keys in the key tree, when one or more users join or leave the group, in such a way that ensures added users are unable to decrypt past traffic, while evicted users are not able to decrypt future traffic – or in other words, ensures *backward secrecy* and respectively *forward secrecy*. Up to this point, only the basic LKH model has been described. Details of LKHW-specific group initialization and re-keying follow.

6.3.2 Group Initialization

Continuing with our previous example of tracking animals, now suppose that the sensor results have to be protected from potential poachers, confidential and authenticated group communication has to be established among the sources and the sink. To achieve this efficiently, we apply LKHW: directed diffusion sources play the role of multicast group members, whereas the sink plays the role of the KDC.

Protocol Before the normal directed diffusion process can begin, a secure group has to be established first, with the *group initialization* process. The rationale is that the confidentiality of the query a node posts to the other nodes is just as important as the confidentiality of the data supplied by the nodes. From a system point of view, the protocol is:

1. $S \rightarrow *$: interest about interests to join
2. $u_i \rightarrow S$: interests to join
3. $S \rightarrow u_i$: data for joining
4. $S \rightarrow u_i$: encrypted normal interest, i.e. secure interest
5. $u_i \rightarrow S$ encrypted normal data, i.e. secure data

where $i = 0, \ldots, n-1$. Since a source in the directed diffusion sense can become a sink in the group initialization phase by emitting an "interest to join", we are careful to not refer u_i ($i = 0, \ldots, n-1$) as sources, but as users/members, and S as the KDC in the discussion below to avoid confusion. The protocol is described in detail below:

1. **Interest about interests to join**: By sending out an exploratory "interest about interests to join" at step 1, S finds out which among the nodes that are capable of the task(s) specified, are 'interested' in joining its secure group. This "interest about interests" would be cached in the nodes that received it, and would match any future interest expressed by whichever nodes that want to join the group. This "interest about interests to join", like normal interests, creates an initial, exploratory set of gradients that diffuse across the network.

2. **Interests to join**: 'Interested' nodes reply with an interest to join, by declaring the task(s) they are capable of, the ID of the group they are joining, their key ID, a nonce

and the MAC of the entire message (Figure 6.2b). These "interests to join" travel down the gradients created by the the previous "interest about interests to join". The interesting observation here is that by originating interests, these 'interested' nodes have actually become both sources and sinks – they are sources for "interest about interests to join" but sinks to which S is going to dispatch their respective "data for joining" (i.e. keying material that includes a member's assigned index and key set). Similarly, S is simultaneously a sink for "interest to join" and a source for "data for joining".

3. **Data for joining**: After collecting enough requests or after a timeout, S proceeds to supply u_i ($i = 0, \ldots, n-1$) with their assigned index (i.e. the i in u_i) and key set. Note that prior to this stage u_i does not know it is user i; u_i is just a name by which we call a node consistently. From this point onward, S can start dispatching normal interests encrypted with the group key K_0, called *secure interests* (Figure 6.2d); and u_i ($i = 0, \ldots, n-1$) can reply with data, encrypted too with the same group key (Figure 6.2e), called *secure data*. Consequently, in-networking processing and data aggregation can start to take place, securely.

```
class EQ interest
task EQ ...
(more "task EQ ..." tuples)
groupID IS ...
```

(a) Interest about interests to join

```
class IS interest
task IS ...
groupID EQ ...
id IS ...
nonce IS ...
mac IS ...
```

(b) Interest to join

```
class IS data
groupID IS ...
id EQ ...
nonce IS ...
index IS ...
keyset IS <encrypted>
reinforce IS true
mac IS ...
```

(c) Data for joining

```
class IS interest
groupID IS ...
task EQ ...
task-specific1
    EQ <encrypted>
task-specific2
    EQ <encrypted>
(other encrypted task
 -specific tuples...)
mac IS ...
```

(d) Secure interest

```
class IS data
groupID EQ ...
task IS ...
task-specific1 IS <encrypted>
task-specific2 IS <encrypted>
(other encrypted task
 -specific tuples...)
mac IS ...
```

(e) Secure data

Figure 6.2: Message formats for group initialization where encrypted entries are marked as <encrypted>.

Required Extension of Directed Diffusion It is important to note that secure interests and secure data are not encrypted as a whole. Instead, only task-specific values as depicted in

Figure 6.2 are encrypted. This is to allow data-interest matching to be carried out on non-task-specific fields so that non-member nodes know how to forward secure data. Specifically, when a node receives a secure data:

- If the node sees that it is not in the group specified by the `groupID` of the data, the node would perform the usual matching algorithm on the `groupID` and the `task`; *ignoring* the encrypted task-specific tuples.
- Otherwise, if the node sees that it is in the the group specified by the `groupID`, the node will decrypt the task-specific tuples, and perform the usual matching algorithm on the `groupID`, the `task` *and* the task-specific tuples.

As a result of this extension, members of group `groupID` would potentially receive data that do not match their interest. However this is just an efficiency issue, the security of the data is not compromised.

As mentioned, since S is both a sink and a source, the keying materials S dispatches, double as data and reinforcements. To be precise, S as a sink decides which neighbour to send the "data for joining" to, i.e. decides on which neighbour to reinforce based on some system-defined parameters. These reinforcements limit the direction/paths of transmissions, and hence increase efficiency. They also provide another advantage that will become clear in Section 6.3.3.

The catch is that in the original directed diffusion model, reinforcements are actually refined versions of the initial interest. In our case, "data for joining" do not qualify as reinforcements per se, but re-sending "interest about interests" as reinforcements is expensive. Therefore LKHW has to add on top of directed diffusion an extra layer of logic that treats "data for joining" as reinforcements. This explains the `reinforce` tuple in Figure 6.2c.

6.3.3 Group Dynamics

In this section, we describe the protocols for leave and join operations, starting with the leave operation.

Leave

When a source leaves the group, it can either be due to voluntary leave or forced eviction. Voluntary leave maybe the result of load control, or the leaving node's self-awareness that it is exiting the region of sensing interest, or that it is in the process of being compromised. On the other hand, forced eviction maybe a result of intrusion detection that decides the node in question is no longer trustable. Or it may just be that the sink is no longer interested in the readings of the particular node.

Example Regardless of the reason, after a node, say u_v has left the group, to ensure forward secrecy all the keys on the path N_v-root need to be refreshed, where N_v is the key tree node from which u_v is detached. Evicting $u_v = u_{12}$ in Figure 6.1 implies that all the keys in key set $\mathscr{K}_{12}$, i.e. K_0, K_2, K_6 and K_{13} have to be refreshed. Our scheme is to refresh K_0, K_2, K_6 according to Equation 6.2:

$$K'_0 = H(K'_2) = H^{(2)}(K'_6) = H^{(3)}(K'_{13}) \tag{6.2}$$

We also need to forward-securely generate K'_{13}. We call the set of K'_0, K'_2, K'_6 and K'_{13} the *refreshed key set*, denoted $\mathscr{R}_v = \mathscr{R}_{12}$. The next step is to deliver these refreshed keys to the appropriate members other than u_{12} securely and efficiently. In LKHW,

- $u_0, \ldots, u_7$ would receive $E_{K_1}(K'_0)$;
- $u_8, \ldots, u_{11}$ would receive $E_{K_5}(K'_2)$, and compute K'_0 according to Equation 6.2;
- u_{14}, u_{15} would receive $E_{K_{14}}(K'_6)$, and compute K'_2, K'_0 according to Equation 6.2;
- u_{13} would receive $E_{P_{13}}(K'_{13})$, and compute K'_6, K'_2, K'_0 according to Equation 6.2.

Note that according to Figure 6.1, K_0 is at level 0, K_2 is at level 1 and so on, that is, only one key is refreshed at a level. If we denote $\mathscr{T}_v^l$ as the *transmitted key set* at level l on the eviction of node u_v, then $\mathscr{T}_{12}^0 = \{E_{K_1}(K'_0)\}$, $\mathscr{T}_{12}^1 = \{E_{K_5}(K'_2)\}$, $\mathscr{T}_{12}^2 = \{E_{K_{14}}(K'_6)\}$, $\mathscr{T}_{12}^3 = \{E_{P_{13}}(K'_{13})\}$.

Similarly we define $\mathscr{U}_v^l$ as the *recipient set* at level l on the eviction of node u_v , i.e. the set of members who receive $\mathscr{T}_v^l$. So $\mathscr{U}_{12}^0 = \{u_0, \ldots, u_7\}$, $\mathscr{U}_{12}^1 = \{u_8, \ldots, u_{11}\}$, $\mathscr{U}_{12}^2 = \{u_{14}, u_{15}\}$, $\mathscr{U}_{12}^3 = \{u_{13}\}$.

Protocol The only difference between voluntary leave and forced eviction is that for voluntary leave, the node, say u_v starts by flowing down a leave message to the sink. From then on, both voluntary and forced eviction are the same. From a system point of view, the protocol is:

1. $S \rightarrow *$: interest about interests to re-key
2. $\mathscr{U}_v \rightarrow S$: interests to re-key
3. $S \rightarrow \mathscr{U}_v^l$: $\mathscr{T}_v^l, \forall l \in \mathscr{L}$

The following is a step-by-step description of the protocol:

1. **Interest about interests to re-key**: S broadcasts an "interest about interests to re-key", the format of which is depicted in Figure 6.3a, where `evictIndex` specifies the index of the evicted node. This interest diffuses across the network, establishing a new set of gradients.

2. **Interests to re-key**: All members except u_v, upon receiving the "interest about interests", set off a timer with a random time-out value $\rho_i\Delta$, where ρ_i is the time-out ratio, $0 \leq \rho_i \leq 1$ and Δ is the maximum time-out value. The rationale of using a random time-out value is basically to facilitate data aggregation but the details are explained later. Denote l_i as the `level` at which node u_i has to re-key, i.e. the *re-key level* of u_i. Then for every member u_i, after the time-out $\rho_i\Delta$ has elapsed, u_i replies with an interest that indicates the transmitted key set, $\mathscr{T}_v^{l_i}$, it should receive. See Figure 6.3b for the message format and notice the use of the `level` tuple, which specifies l_i and essentially $\mathscr{T}_v^{l_i}$. Notice further that there can be more than one `level` tuple, to cater for data aggregation. These interests travel down the gradients that have previously been established by S's "interest about interests".

   Now we justify the use of random time-outs with Proposition 1:

   Proposition 1 Given a member u_i, which has $N \leq n-2$ upstream neighbours, and given that the N neighbours are also group members, assuming the the link latency is negligible, the probability that there exists at least one u_j among the N neighbours such that $l_j \neq l_i$ and $\rho_j < \rho_i$ is given by

$$\frac{N}{N+1}\left[1 - \sum_{l=0}^{L_{max}} \binom{(a-1)a^{h-l-1}}{N+1} \Big/ \binom{n-1}{N+1}\right]$$

where $L_{max} = \lfloor h - \lg_a \lceil \frac{a(N+1)}{a-1} \rceil \rfloor$.

This is the probability that node u_j is able to aggregate at least one re-key level from one of its upstream neighbours (see proof in 6.6). To see how significant the probability is, let us use a binary key tree ($a = 2$), and suppose there are 16 members in the group before eviction ($n = 16$, $h = 4$). Suppose for the particular node u_i we are considering, u_i has 3 neighbours ($N = 3$), then according to Proposition 1, u_i has a probability of 71% for performing at least one aggregation.

To further illustrate the details, we borrow the help of a sample topology depicted in Figure 6.4, where node u_{12} is in the process of being evicted. Pick a random node, say u_7 in Figure 6.4. According to Figure 6.1, $l_7 = l_0 = 0$, and $l_8 = 1$. When u_8 receives u_7's "interest to re-key", u_8 will cache *and* re-broadcast u_7's interest because $l_7 \neq l_8$. Similarly when u_0 receives u_7's and u_8's interest, u_0 will cache both interests, but only re-broadcast u_8's interest because $l_0 = l_7 \neq l_8$. Eventually, S receives $l_0 = 0$ and $l_8 = 1$ from u_0 (ignoring the `level` of other nodes than u_7, u_8, u_0 for this example).

Since u_8 needs to send u_7's and its own interest, and similarly, u_0 needs to send u_8's and its own interest, both u_8 and u_0 should ideally aggregate the interests they have to send, before sending them. Proposition 1 suggests that this will happen with a high probability.

3. **Data for re-keying**: Continuing with the example, S then knows $\mathscr{T}_{12}^0$ and $\mathscr{T}_{12}^1$ are needed and would deliver them upstream in the direction of u_0. u_0, remembering that $\mathscr{T}_{12}^0$ and $\mathscr{T}_{12}^1$ are needed, would likewise deliver the key sets upstream. Finally u_8 would send only $\mathscr{T}_{12}^0$ upstream since it is only u_7 who needs it (but of course u_8 does not know it is u_7 who needs it – u_8 only knows some node needs it). The importance of the data cache cannot be emphasized more here. For example, if u_0 has sent and got $\mathscr{T}_{12}^0$ *before* u_7's "interest to re-key" arrives, u_0 can easily retrieve $\mathscr{T}_{12}^0$ from its data cache and hand it over to u_7 without going through S again.

```
class EQ interest
groupID IS ...
evictIndex IS ...
mac IS ...
```

(a) Interest about interests to re-key

```
class IS interest
groupID EQ ...
evictIndex EQ ...
level IS ...
(possibly more
 level tuples...)
mac IS ...
```

(b) Interest to re-key

Figure 6.3: Message formats for leaving.

Join

For a system to be scalable in general and for sensing tasks that require dynamic adjustment of sensing resolution in particular, the capability to add computing power dynamically to the network is essential. The join operation of LKHW makes this possible.

Protocol As it happens in many secure multicast schemes, the join protocol and the leave protocol are asymmetrical with the join protocol being more efficient because the joining node does not know any existing key of the system. Similar to a leave operation, a join operation can either be initiated by S or by whichever node u_v that wants to join. Suppose

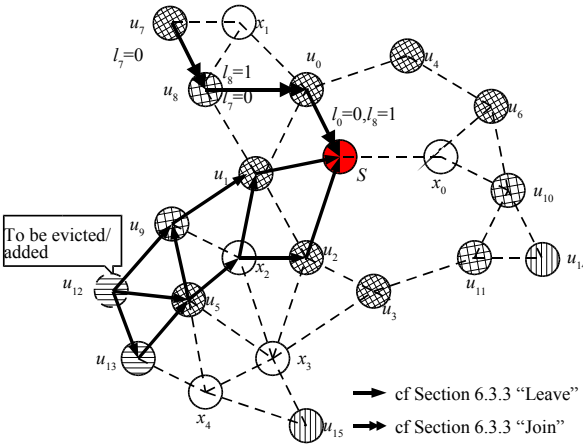


Figure 6.4: An arbitrary topology where S denotes the sink/KDC; members are named u_i ($i = 0, \ldots, 15$); non-members are named x_j ($j = 0, \ldots, 4$); nodes with similar hatch pattern belong to the same recipient set $\mathscr{U}_{12}^l$ ($l = 0, 1, 2, 3$).

.

S requires a higher sensing resolution, S can start sending "interest for interests to join" again. Moreover, recall that in Section 6.3.2, S's "interest about interests to join" is cached in the network, so for any u_v which sends out its "interest to join", its "interest to join" would syntactically match the cached "interest about interests to join" – with a catch: The format of Figure 6.2b specifies the ID of the group a node wants to join. As it is unrealistic for a node to find out the group ID beforehand, it should just omit the `groupID` tuple in its "interest to join" and the protocol will still work. From a system point of view, the protocol is:

1. $u_v \rightarrow *$: interest to join
2. $S \rightarrow u_v$: $\mathscr{K}_v$
3. $S \rightarrow u_i$: seed, $\forall i \in \{0, \ldots, n-1\} \setminus \{v\}$

where "seed" is the key regeneration seed. Detailed explanation follows:

1. **Interest to join**: Section 6.3.2 mentioned the dual role of "data for joining", i.e. as data cum reinforcements. We will clarify the advantage of doing so here. First, it is clear that when u_v broadcasts its "interest to join", in the directed diffusion model, the interest has to diffuse across the network. Now the good news is that since S has planted the seed of reinforced gradients in the group initialization phase, u_v's "interest to join" can simply take advantage of the reinforced gradients, i.e. any node that receives the "interest to join" would flow it down the reinforced gradients if it happens to be on one or more of the reinforced gradients.

   See for example in Figure 6.4, the possible reinforced gradients that take $u_v = u_{12}$'s "interest to join" to S. In this case, u_{12}'s neighbors u_{13}, u_5, u_9 send u_{12}'s "interest to join" only along the reinforced gradients.

2. **Data for joining**: Once S receives u_v's message, S would, as in the group initialization phase, dispatch $\mathscr{K}_v$ (among other things) to u_v as data cum reinforcement.

3. **Seed diffusion**: The key regeneration seed is used by existing members to refresh their respective key sets. The seed does not need to be encrypted but must be authenticated with the group key. Every member u_i ($i \neq v$) that receives the seed would derive its new key set $\mathscr{K}_i'$ from its existing key set $\mathscr{K}_i$ as follows:

$$\mathscr{K}_i' = \{K' | K' = \text{MAC}_K(\text{seed}), \forall K \in \mathscr{K}_i\} \tag{6.3}$$

The forward security of the seed can be ensured by Bellare and Yee's construction [25].

6.4 Performance Evaluation

In this section, we perform a theoretical evaluation of the performance of LKHW. The performance criterion for WSN is energy efficiency instead of throughput. We note that while it is a convention to evaluate computational as well as communication cost as benchmarks for secure multicast schemes, for WSN however communication cost dominates computational cost by typically three orders of magnitude [50], therefore we only consider communication cost. For large messages, energy consumption is proportional to the message length, however for small messages, the transmission overhead and hence the number of messages has a more significant effect on the energy cost. Therefore we will take both message length and number of messages into account. Moreover, the energy for reception is not insignificant, at least for the transceiver we are using, RF Monolithics TR1001 [243], it can be as high as 40% of the energy required for transmission. We denote the ratio of reception power to transmission power as r for the discussion below. We will discuss the leave and join operation separately.

6.4.1 Leave

Let us consider the sink S and the sources u_i ($i = 0, \ldots, n-1$, $i \neq v$, v is the index of the evicted node) separately. In LKHW, for S, the number and the total length of messages sent, and hence energy cost is network topology-dependent. For example, in Figure 6.4, S would potentially receive requests for the transmitted key set at level 0, $\mathscr{T}_v^0$, from all its neighbours, i.e. u_0, u_1, u_2 and x_0. Directed diffusion dictates that S would send, instead of broadcast, $\mathscr{T}_v^0$ as data replies to u_0, u_1, u_2 and x_0 individually. So S would send $\mathscr{T}_v^0$ four times. On the other hand, requests for $\mathscr{T}_v^3$ would potentially only come from u_1 and u_2, and hence S would send $\mathscr{T}_v^3$ two times. Intuitively, we can certainly do better than what directed diffusion allows us to. Observe that instead of unicasting to each requesting neigbhour, we might be able to save energy by broadcasting all the key sets $\mathscr{T}_v^l$, $l = 0, \ldots, h-1$ at once because each key set $\mathscr{T}_v^l$ would have to be sent at least once anyway. To formalize our argument, suppose S receives "interests to re-key" from N' out of a total of N neighbours. Using unicasts, the energy cost for S to dispatch the requested key sets is

$$E_{unicasts} = (1+r)\sum_{i=1}^{N'} C_i E_{ks} + (1+r)N'E_o \tag{6.4}$$

where C_i is the number of key sets requested by neighbour i, $i = 1, \ldots, N'$; E_{ks} is the energy associated with sending a key set; and E_o is the energy associated with the overhead of a single "data for re-keying" message. Notice that apart from considering the energy cost for S, we also consider the energy cost incurred on S's neighbours for listening, hence the terms containing r. The energy cost for broadcasting all key sets at once is

$$E_{broadcast} = (1+Nr)hE_{ks} + (1+Nr)E_o \tag{6.5}$$

The terms N', N, $\sum_{i=1}^{N'} C_i$ are entirely topology-dependent, hence there is no way of determining whether $E_{unicasts}$ is larger or smaller than $E_{broadcast}$ without considering the topology. Now, we can make S adopt this adaptive policy: after collecting enough "interests to re-key", compute $E_{unicasts}$ and $E_{broadcasts}$, if $E_{unicasts} > E_{broadcasts}$, then aggregate and unicast the requested key sets to each requesting neighbour, otherwise broadcasts all key sets at once.

In practice, we expect $N' \approx N$, and the term $(1+r)\sum_{i=1}^{N'} C_i$ to often be larger than $(1+Nr)h$, in which case, S will choose to broadcast. Since N is related to the network density which does not vary much for the same type of applications, intuitively then, $E_{broadcast}$ increases with the number of levels h, which is logarithmic to the group size. Therefore $E_{broadcast}$ scales logarithmically with the group size.

This adaptive policy does not break directed diffusion, because it needs only apply to S. In the event that S decides to use broadcast instead of unicasts, the neighbours of S which have not submitted any request for key sets but have received S's broadcast, would just drop the broadcast data, since there is no matching interest.

Now we consider the sources. The upper bound of the energy cost for a source u_i applies when the N upstream neighbours of u_i time out later than u_i, *and* the re-key levels received from these N neighbours span the set $\mathscr{L}$. The upper-bound energy cost is therefore:

$$\begin{aligned} E_{upper} &= r\sum_{i=1}^{N}(C_i E_l + E_o) + (E_l + E_o) + (h-1)(E_l + E_o) \\ &= (h + r\sum_{i=1}^{N} C_i)E_l + (h + Nr)E_0 \\ &\approx (h + Nr)E_0 \end{aligned} \tag{6.6}$$

where C_i is the number of re-key levels sent by neighbour i, $i = 1, \ldots, N$; E_l is the energy required for sending a re-key value; and E_o is the energy associated with the overhead of a single "interest to re-key" message. Note that the number of bits for representing a re-key level is potentially very small compared with the the overhead of an "interest to re-key" message (Figure 6.3) which is taken into account by E_o, or in other words $E_l \ll E_o$. N is related to the network density which does not vary much for the same type of applications. Using the same logic as before, E_{upper} scales logarithmically with the group size. On the other hand, the lower bound applies when u_i receives no request from its neighbours, and needs only to send its own re-key level:

$$E_{lower} = (E_l + E_o) \approx E_0 \tag{6.7}$$

6.4.2 Join

The case of join is much simpler because it simply involves dispatching the relevant key set to the new member, and flooding the network with the key regeneration seed. The transmission energy cost for S alone is independent of the topology:

$$E = E_{ks} + E_{seed} + E_{o(unicast)} + E_{o(broadcast)} \tag{6.8}$$

where E_{seed} is obviously the energy for broadcasting the seed; $E_{o(unicast)}$ and $E_{o(broadcast)}$ are associated with the overhead of a single unicast and a single broadcast respectively. For the new joining member, the cost involves dispatching its "interest to join" and receiving the allocated key set. The existence of reinforced gradients ensures that the energy cost for

propagating the "interest to join" across the network to the sink is low. For existing members, the cost primarily involves receiving the seed.

6.5 Related Work

LKHW is, as far as we know, the first secure group communication scheme to reap the benefits of directed diffusion. We have already reviewed directed diffusion [108, 119] in Section 6.2 in details. In the following, we will compare LKHW with other group communication schemes from two perspectives, the distributed approaches and the hierarchical approaches.

Distributed Approaches Also called *conference key distribution schemes*, the approaches in this category try to solve the *key agreement problem*, i.e. the problem of deriving a secure common key among n users. Ingermasson et al. [118] are the first to extend the Diffie-Hellman (DH) problem to groups. Burmester and Desmedt [44] correct Ingermasson et al.'s security flaw by using cyclic instead of symmetric functions. Just and Vaudenay [131] patch the key authentication flaw of the Burmester-Desmedt scheme and generalize it. Steiner et al. [270, 271] begin to consider *dynamic peer groups* instead of a fixed number n of users, and introduce CLIQUES, a suite of *contributory* key agreement protocols which requires less communication than the Burmester-Desmedt scheme does. Ateniese et al. [21] provide CLIQUES with authentication properties. The problem with these conference key distribution schemes is that they require a lot of exponentiation computations that are prohibitively expensive for sensors [50, 110], and re-keying is inefficient. Blundo et al. [35] did the pioneering investigative work on the storage requirements of k-secure t-conference key distribution scheme. The proposed scheme is information-theoretically secure, and does not require exponentiation, but requires $O(n^t)$ amount of keying material per node (where n is the total number of users), which is impractically large. The conclusion is that both DH-based and information-theoretically secure schemes have their share of scalability problems.

Hierarchical Approaches By imposing a hierarchial structure – binary tree being the mainstream – on dynamic peer groups, hierarchical approaches have been able to achieve better scalability than the distributed approaches. There is the pioneering work by Wallner et al. [292] who propose the logical key hierarchy (LKH) model. The LKH model is scalable because the total number of keys in the system is linearly proportional to the number of users, and both the number of keys per user and the number of messages required to manage group dynamics, are logarithmic in the number of users. Wong et al. [299] investigate and compare three re-keying paradigms, i.e. key-oriented, user-oriented and group-oriented re-keying. The re-keying paradigm of LKHW is an improved form of group-oriented re-keying. McGrew et al. [178] introduce the one-way function tree (OFT), where the KDC needs only dispatch $\lceil \lg_a n \rceil$ keys during re-keying, the same number as LKHW's, instead of the $2\lceil \lg_a n \rceil$ required by the basic LKH model. Canetti et al. [48] replace McGrew et al.'s non-standard cyptographic primitive with pseudorandom generator. The EHBT scheme proposed by Rafaeli et al. [236] uses a non-standard cryptographic primitive similar to OFT's, i.e. one-way function of the form $h(\text{key} \oplus \text{index})$ for key refreshment. In all the schemes mentioned so far, affected group members need to receive refreshed keys from the KDC during user-join events. Members in Perrig et al.'s ELK [213] however avoid that overhead by locally *and* periodically re-generating their keys. Di Pietro et al. [218] provide further improvement by exploiting pseudorandom function and the "level-awareness" of a node in a scheme called LKH++.

LKHW requires the same number of keys for the KDC, i.e. $2n - 1$; and the same number of keys for a member, i.e. $h + 1$, as OFT, EHBT, ELK and LKH++ do. LKHW's handling of user-join events might seem inefficient since a seed is flooded across the network, but due to the multihop nature of WSN (where every node behaves as a router) *and* the data caching property of directed diffusion, this method is in fact not only efficient but also robust.

6.6 Conclusion and Future Work

We have presented a secure group communication scheme that is optimized for directed diffusion. The scheme is independent of the underlying key management architecture. We have given the details of handling group dynamics. In terms of efficiency, the re-keying overhead in terms of energy cannot be concretely quantified without considering the topology, but it is found to be approximately logarithmic to the group size. In fact, the conventional evaluation methodology is no longer apt in the WSN context. In view of this, we have presented a new evaluation methodology that is entirely based on energy efficiency. As part of our future work, we would address multiple join and multiple leave operations, as well as the efficient re-balancing of the LKH tree in the directed diffusion context.

Appendix: Proof of Proposition 1

Denote X as the event that u_i can perform at least one aggregation, i.e. that there exists at least one u_j among the N neighbours such that $l_j \neq l_i$ and $\rho_j < \rho_i$. Further denote A as the event that all N neighbours have the same re-key level as u_i's; and B as the event that all N neighbours have a time-out greater or equal to u_i's. Then,

$$\begin{aligned} Pr[X] &= 1 - Pr[A] - Pr[B] + Pr[AB] \\ &= 1 - Pr[A] - Pr[B] + Pr[A]Pr[B] \end{aligned} \tag{6.9}$$

since events A and B are mutually independent. Recall that ρ_i is as defined in Section 6.3.3, the time-out ratio of u_i. We first derive $Pr[B]$, assuming ρ_i is uniformly distributed between 0 and 1:

$$\begin{aligned} Pr[B] &= \int_{\rho_i=0}^{1} (1 - \rho_i)^N \\ &= \frac{1}{N+1} \end{aligned} \tag{6.10}$$

reducing (6.9) to the simpler form of (6.11).

$$Pr[X] = \frac{N}{N+1}(1 - Pr[A]) \tag{6.11}$$

As for $Pr[A]$, we observe that the number of nodes, n', having the same re-key level l to be:

$$n'(l) = (a-1)a^{h-l-1}$$

expressed as a function of l. For example, in Figure 6.1, the nodes sharing the re-key level of $l = 1$ during the eviction of u_{12} are $u_8, \ldots, u_{11}$, or equivalently $n' = 4$. Observe that n' decreases as l increases, so there exists a maximum value of l, L_{max} such that $n'(L_{max}) = N+1$, or

$$L_{max} = h - \lg_a \left\lfloor \frac{a(N+1)}{a-1} \right\rfloor$$

Constraining L_{max} to integer values, we have

$$L_{max} = \left\lfloor h - \lg_a \left\lceil \frac{a(N+1)}{a-1} \right\rceil \right\rfloor \tag{6.12}$$

When $l > L_{max}$, at least one of the N neighbours of u_i will have a different re-key level, a condition precluding event A. Therefore,

$$Pr[A] = \frac{\sum_{l=0}^{L_{max}} \binom{n'}{N+1}}{\binom{n-1}{N+1}} \tag{6.13}$$

where n is the total number of nodes in the group *before* eviction. Substituting $Pr[A]$ in (6.11) with (6.13), we get Proposition 1.

□

CHAPTER VII *

Link-Layer Jamming Attacks on S-MAC

Abstract We argue that among denial-of-service (DoS) attacks, link-layer jamming is a more attractive option to attackers than radio jamming is. By exploiting the semantics of the link-layer protocol (aka MAC protocol), an attacker can achieve better efficiency than blindly jamming the radio signals alone. We investigate some jamming attacks on S-MAC, the level of effectiveness and efficiency the attacks can potentially achieve, and a countermeasure that can be implemented against one of these attacks.

7.1 Introduction

Radio jamming is potentially the most direct, non-destructive and yet disruptive form of DoS attack on sensor networks. There are two reasons why an attacker might favor radio jamming over other DoS attacks: (1) it is trivial to execute – the jammer only needs to emit an arbitrary constant signal at a power roughly equal to the signal power of its victims [273], (2) sensor nodes are typically limited to single-frequency use [300], depriving them of the possibility to use standard countermeasures such as direct sequence spread spectrum and frequency hopping spread spectrum, or some combination of these two techniques [238]. On the other hand, blind radio jamming is not nearly as energy-efficient as link-layer jamming. Let us elaborate in the paragraph below.

Suppose there is a sensor network that has no fixed base stations as points of control (e.g. in a high-security application). A radio jammer, having *no* base stations to focus its jamming signals on, would have to target the sensor nodes themselves. Assume that every sensor node has a signal power of P_s. To jam an area of radius r using an omni-directional antenna, each jamming device needs to draw a power that is proportional to $r^\beta P_s$ ($3.5 \leq \beta \leq 5$ for outdoor environments) [238]. That is, the wider the area, the exponentially more power a jamming device needs. If the energy to these devices cannot be replenished, with high probability, the devices would exhaust themselves much faster than the sensor nodes would. A solution for the attacker is to reduce the jamming power and increase the number of jamming devices, to such an extent that the attacker would reduce to using a network of jamming motes, that is, a network of nodes having comparable capability with the sensor nodes. Taking this a step further, instead of blind radio jamming, the attacker can save energy with link-layer jamming, by taking advantage of the MAC protocol. Note that we are proposing a general attack strategy where

1. there are no fixed *sinks* (nodes that request for, and hence sink information) to attack, e.g. in directed diffusion [120] where ID-less interests propagate through the network, there is no way for a node to tell where the real sinks are;

2. it is infeasible for the attacker to deploy its nodes strategically, except perhaps to air-drop them on top of the target network;

*This chapter is a minor revision of the paper with the same title published in the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005), pages 217–225, IEEE, ISBN 0-7803-8801-1 [4].

3. the attacker can only estimate where and not how the target network is or would be deployed – knowing the location allows the attacker to plan an ambush though.

We will show how an attacker may take advantage of the MAC protocol, using S-MAC [309], one of the earliest protocols specifically designed for sensor networks, as our example. The contributions of this paper are (1) to show 3 easy-to-implement attacks on S-MAC, and (2) to show an equally easy-to-implement countermeasure against one of these attacks, thereby (3) delivering the message that MAC protocols should be designed with prevention against DoS attacks in mind.

The paper is organized as follows. Section 7.2 summarizes the S-MAC protocol. The attacks are described in Section 7.3, followed by a countermeasure in Section 7.4. Related work is discussed in Section 7.5 before the conclusion is given in Section 7.6.

7.2 The S-MAC Protocol

The prime focus in the design of MAC protocols for sensor networks is minimizing energy use. S-MAC reduces energy wastage by avoiding collisions, overhearing, control packet overhead and idle listening. In S-MAC, every node follows one or more *schedules*. The node that first sets and broadcasts a schedule is the *synchronizer*. Other nodes that receive and adopt the schedule are *followers*. The synchronizer and followers thus form a *virtual cluster*, listening and sleeping at about the same time.

A schedule indicates the times t_0, $t_0 + T_P$, $t_0 + 2T_P$, ... at which the synchronizer and followers go to sleep, where T_P is the length of a cycle, or *period* (Figure 7.1). A complete cycle consists of a listen interval T_L seconds long and a sleep interval T_S seconds long, i.e. $T_P = T_L + T_S$. The *duty cycle* is defined as the ratio T_L/T_P, therefore the lower the duty cycle, the more energy nodes get to conserve, albeit at the price of incurring higher network latency. A listen interval is further divided into a synchronization (SYNC) interval, and a control (CTRL) interval.

The SYNC interval is for sending SYNC packets that allow neighboring nodes to synchronize their schedules:

1. When a node first joins a network, it listens for a whole period. If the channel is clear, the node broadcasts its schedule in a SYNC packet in the SYNC interval. A SYNC packet consists primarily of the address of the sender and the time the sender goes to sleep, t_s, relative to the sender's current time. This node becomes a synchronizer.

2. If a node receives a schedule from a neighbor before choosing its own schedule, it adopts the schedule, and re-broadcasts the schedule after a random delay t_d, indicating that it will sleep at $t_s - t_d$. This node becomes a follower.

3. If a node receives a different schedule after broadcasting its own, it adopts both schedules [310].

Every node broadcasts its schedule periodically (every *SYNC period*) even if it has not received any other node's schedule, so that when new nodes join the network, they are able to synchronize with existing schedules.

The CTRL interval is for medium reservation, to avoid collision during node-to-node data transmissions, by means of exchanging RTS and CTS packets, akin to the IEEE 802.11 Distributed Coordination Function (DCF). The CTRL interval is also for data broadcast, in which case no exchange of RTS/CTS packets is involved.

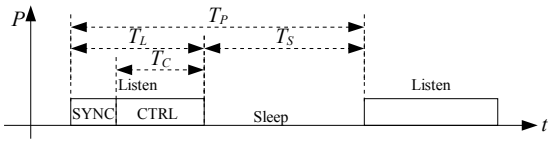


Figure 7.1: S-MAC schedule (T_P = period, T_L = length of listen interval, T_S = length of sleep interval, T_C = length of control interval).

7.3 Attacks on S-MAC

In this section we discuss our assumptions of the sensor network and the attacker (as a network of jamming motes), details of our attacks and countermeasure, coupled with simulation results.

7.3.1 Assumptions

We assume an attacker has 2 goals: the primary goal is to disrupt the network by preventing messages from arriving at the sink node, and the secondary goal is to increase the energy wastage of the sensors. While a sink node is a node that requests information from other sensor nodes, the nodes that supply the requested information are called source nodes [120].

We assume the sensor nodes exchange messages that are not encrypted although they might or might not be cryptographically signed. We make this assumption because: (1) encrypted messages introduce an additional cost of at least 200 cycles or 400 nJ per byte using the fastest symmetric algorithm on an ultra-low-power processor [2]; (2) we know of no key management scheme that protects the link layer with more than one encryption key, and consequently the compromise of a single key compromises the whole network; (3) by listening on the channel for some time an attacker can still learn about the S-MAC parameters and attack accordingly. The third point deserves elaboration: if we assume data packets to be typically larger than SYNC/RTS/CTS/ACK packets (which are roughly of the same length), an attacker might be able to build a prediction model of when data packets are transmitted, by just collecting statistics of the times when longer packets are transmitted. In fact, we are working on such a prediction model.

However we do not assume the attackers know the values of the S-MAC system parameters (e.g. the period, the duty cycle etc.) used by the sensor nodes. Our attacks include a method of estimating these parameters just by listening.

7.3.2 Simulation and Evaluation Model

All protocols, attacks and countermeasures have been simulated on Pentium 4 PCs with at least 768 MB of RAM using the OMNeT++ simulator (`www.omnetpp.org`). In the case of no attackers, 100 normal nodes (1 sink node + 99 source nodes) are pseudorandomly distributed in a square area whose length l is derived from the intended network density D [316] and transmission range r using Equation 7.1:

$$l = \sqrt{\frac{100\pi}{D}}r \tag{7.1}$$

The normal nodes use S-MAC with *adaptive listening* [310] and a duty cycle of 10% at the link layer, and TinyDiffusion [196] at the network layer, both faithfully ported from

TinyOS (`tinyos.sf.net`). For a total of T_{sim} virtual seconds, the sink node broadcasts an interest every 10 seconds, while the source nodes each broadcast a matching data every 2 seconds, as an approximation of a network with moderately fast-changing data. T_{sim} is set to 600 and 1200, where the longer time is for seeing the effect of longer simulation time on the results. For simulating mobility, the improved random waypoint model [313] is used, with a minimum speed of $r/20$ and a maximum speed of $r/5$ (giving the average speed as $r/8$). We believe that the choices above reflect the typical workload of a small-scale sensor network [54]. Every experiment is run 20 times, using 32 new seeds for the 32 pseudorandom number generators in OMNeT++ each time. Attacks are simulated by adding 50 or 100 jamming motes, of the same transmission range, energy supply and power consumption as the normal nodes, to the network. The ratio between the power consumption in sleep, Tx and Rx mode is 1:2400:960 [243]. The source code is available at `wwwes.cs.utwente.nl/eyes/smac_attack.zip`.

One of the things that are not simulated is the interference resulted from jamming, which would in practice cause more data packet loss than in simulation. However this is one of the many simplifications that are conventionally applied in simulations, as is the case that simulated radio ranges are circular/spherical by convention – a departure from actual measurements [240].

We evaluate how *effective* an attack is by measuring primarily the *censorship rate* R_c and secondarily the *attrition rate* R_a. R_c measures the percentage of messages blocked. Let M be the number of messages arriving at the sink in the absence of attacks, and M' be the number of messages arriving at the sink in the presence of attacks, then

$$R_c = (M - M')/M \times 100\% \tag{7.2}$$

R_a measures the percentage of additional energy the sensor network has to spend in the presence of attacks. Let E be the amount of energy spent when there is no attack, and E' be the amount of energy spent when there is attack, then

$$R_a = (E' - E)/E \times 100\% \tag{7.3}$$

To evaluate how *energy-efficient* an attack is, we use the *effort ratio* R_e, defined as the ratio of the attacker's per-node energy expenditure to the sensor network's per-node energy expenditure when not under attack. Using R_a and R_e, we can calculate the *lifetime advantage* R_l of a jamming mote over a sensor node, that is how long a jamming mote can live compared with a sensor node. Let T_{att}, T' be the lifetime of a jamming mote and the lifetime of a sensor node under attack respectively. Assuming the power is constant during the lifetime of a jamming mote or sensor node, i.e.

$$E_{\text{total}}/T_{\text{att}} = E_{\text{att}}/T_{\text{sim}} \qquad E_{\text{total}}/T' = E'/T_{\text{sim}}$$

then the attacker's lifetime advantage

$$R_l = T_{\text{att}}/T' = E'/E_{\text{att}} = (1 + R_a)/R_e \tag{7.4}$$

7.3.3 Details of Attacks

Since S-MAC uses a periodic listen-sleep schedule, the simplest attack is to jam the entire listen interval. If the duty cycle is 50%, such an attack ideally allows the attacker to use 50%

less energy compared with using radio jamming. Ironically, the more energy the network wants to conserve with a lower duty cycle, the more energy-efficient the attacker becomes. A jamming mote can use Equation 7.5 to 7.7 to estimate the period T_P, the length of a listen interval T_L and the length of a CTRL interval T_C, by just listening to SYNC packets (Figure 7.2):

$$T_P = t_1 + t_{s1} - t_0 - t_{s0} \tag{7.5}$$

$$T_L = \max(t_{s0}, t_{s1}, t_{s2}, ...) + \delta \tag{7.6}$$

$$T_C = \min(t_{s0}, t_{s1}, t_{s2}, ...) - \delta \tag{7.7}$$

where t_{s0}, t_{s1}, t_{s2} are sleep times obtained from SYNC packets, δ is a small estimated constant that compensates for the DCF Inter-Frame Space (DIFS) [115] and carrier-sensing time spent by the sender.

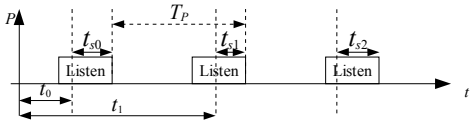


Figure 7.2: Estimating S-MAC parameters.

Once a jamming mote gets an estimation of T_P and T_L, it can adopt a periodic jam-sleep schedule, jamming when its neighbors are listening, and sleeping when its neighbors are sleeping. Note that since a jamming mote must watch out for a pair of consecutive SYNC packets sent by the same sender, it will be some time before it can synchronize with the schedule.

On speculation, the attack can be improved by jamming just the SYNC or CTRL interval. Consider first the case of jamming the SYNC interval and sleeping for the rest of a cycle (i.e. the CTRL interval and the sleep interval). The idea of this attack is that if a sensor node fails to receive the SYNC packets of its neighbors, it would not know it has neighbors that it can transmit data packets to. However some SYNC packets do slip through especially before the jamming motes enter their jam-sleep cycle, and when they do, the sensor nodes that receive the SYNC packets know they themselves are connected. Connected sensor nodes continue to use the CTRL interval to send data packets without problems. So jamming the SYNC interval is ineffective, as verified by simulations.

Now consider the case of jamming the CTRL interval. Jamming the CTRL interval prevents RTS/CTS handshakes, and thus data transmissions from taking place. Speculating further, instead of jamming the whole CTRL interval, we can *listen* in the CTRL interval for CTS packets, and jam the ensuing data packets instead. Since listening consumes less energy than transmission, we might be able to use less energy compared with blindly jamming the CTRL interval.

Table 7.1 shows the censorship rates of the 3 attacks discussed so far: jamming of the listen interval, the CTRL interval, or data packets. Figure 7.3 shows the attrition rates and effort ratios, with standard deviations mostly below 10%. In our simulation, the network density D ranges from 3 to 7. A network with $D = 1$ or 2 is too sparse to be realistic, while we have problem going beyond $D = 7$, being currently limited by the computational resources required to simulate networks that dense. After all, networks sparser than our simulated ones exist in practice [317]. We discuss our observation below in several aspects.

Table 7.1: Average censorship rates of various attacks (standard deviations in parenthesis).

(a) T_{sim} = 600s

Jammed target	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
Listen interval	50	62 (25)	72 (12)	73 (28)	81 (14)	81 (16)
	100	86 (12)	90 (7)	89 (13)	87 (18)	83 (13)
CTRL interval	50	72 (19)	76 (24)	81 (16)	84 (10)	82 (26)
	100	88 (9)	93 (7)	94 (8)	94 (6)	90 (9)
Data packets	50	78 (13)	85 (11)	87 (9)	95 (3)	93 (7)
	100	96 (3)	97 (3)	97 (4)	99 (1)	99 (3)

(b) T_{sim} = 1200s

Jammed target	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
Listen interval	50	58 (20)	77 (16)	77 (20)	85 (12)	88 (9)
	100	89 (7)	94 (4)	92 (8)	93 (9)	91 (7)
CTRL interval	50	70 (16)	78 (17)	84 (9)	87 (11)	87 (13)
	100	90 (6)	95 (5)	93 (10)	95 (6)	95 (4)
Data packets	50	77 (10)	84 (8)	87 (6)	94 (3)	92 (12)
	100	94 (5)	96 (3)	96 (4)	99 (1)	99 (1)

Network density

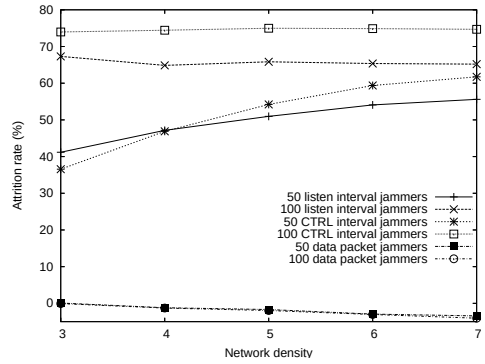
As expected, the lower the density of the jamming motes, or the density of the entire network, the lower the censorship rate of the motes is, since there would be more regions outside the influence of the motes. Although there are simulation runs that record 100% censorship rate, this should not in general be expected, due to the unpredictability and the dynamic nature of the simulated topology.

Censorship rate

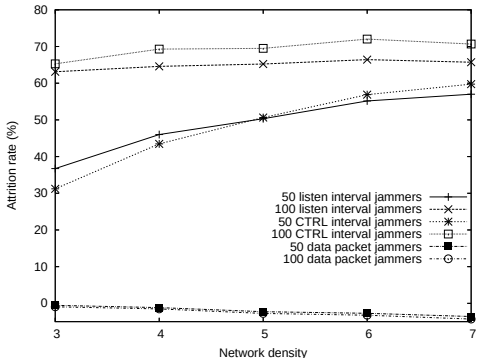
Data packet jamming has on the average the highest censorship rate for the following reason: before a jamming mote is able to synchronize with the S-MAC schedule, it can already listen for CTS packets and jam the ensuing data packets. In contrast, listen/CTRL interval jammers only listen for SYNC packets, allowing data packets to slip through, *before* starting their jam-sleep cycle. Between listen and CTRL interval jammers, CTRL interval jammers censor more data because they settle into their jam-sleep cycle earlier than listen interval jammers do.

Denote $F(t)$ as the fraction of jamming motes that still have not entered their jam-sleep cycle at time t. Figure 7.4 plots $F(t)$ against t, for the three types of jammers. It shows that by 200s, all CTRL interval jamming motes and all data packet jamming motes have already settled into their jam-sleep cycle, while it may take indefinitely long for all listen interval jamming motes to do so. Concluding from the preceding analysis, if we disregard the results for the first 200s (the 'settling time'), CTRL interval jammers and data packet jammers have about the same censorship rate, however listen interval jammers would still have a lower censorship rate. And because of the shorter settling time, the censorship rates measured in T_{sim} = 1200s (Table 7.1(b)) do not differ much from those measured in T_{sim} = 600s (Table 7.1(a)) for CTRL interval jammers and data packet jammers. In comparison, for listen interval jammers, the corresponding differences are larger.

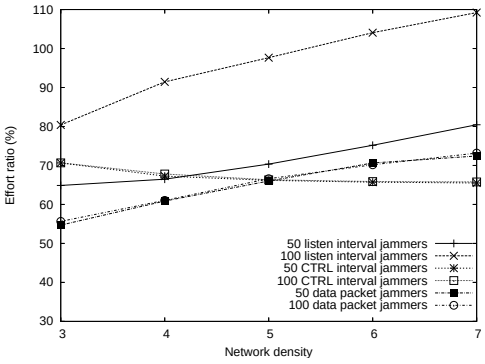
In practice, since listen/CTRL jamming relies on overlapping the jamming period with the listen/CTRL interval as closely as possible, incomplete overlapping might also allow



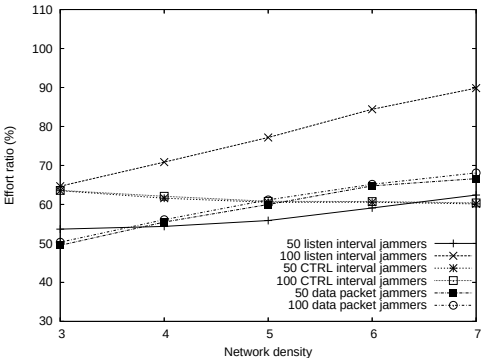
(a) Attrition rate (T_{sim} = 600s)



(b) Attrition rate (T_{sim} = 1200s)



(c) Effort ratio (T_{sim} = 600s)



(d) Effort ratio (T_{sim} = 1200s)

Figure 7.3: The attrition rates and effort ratios of jamming attacks.

some packets to slip through – recall in Equation 7.6 and 7.7 the use of an estimated constant δ. Note that in the controlled environment of our simulation, we are using a good estimate of δ, so this is *not* the reason for the lower censorship rate of listen/CTRL jamming in our case.

Attrition rate

CTRL interval jamming is more successful in causing its victims to consume more energy than the other attacks are. This is because it allows its victims to listen and send SYNC packets in the SYNC interval (consuming energy for both Rx and Tx), and forces its victims to listen for the whole of the CTRL and sleep interval (consuming energy for Rx). Note that in the original implementation of S-MAC, nodes recovering from the BACKOFF state in the sleep interval (when the jamming stops) continue to listen. In comparison, listen interval jamming only forces its victims to listen for the entire period (consuming only energy for

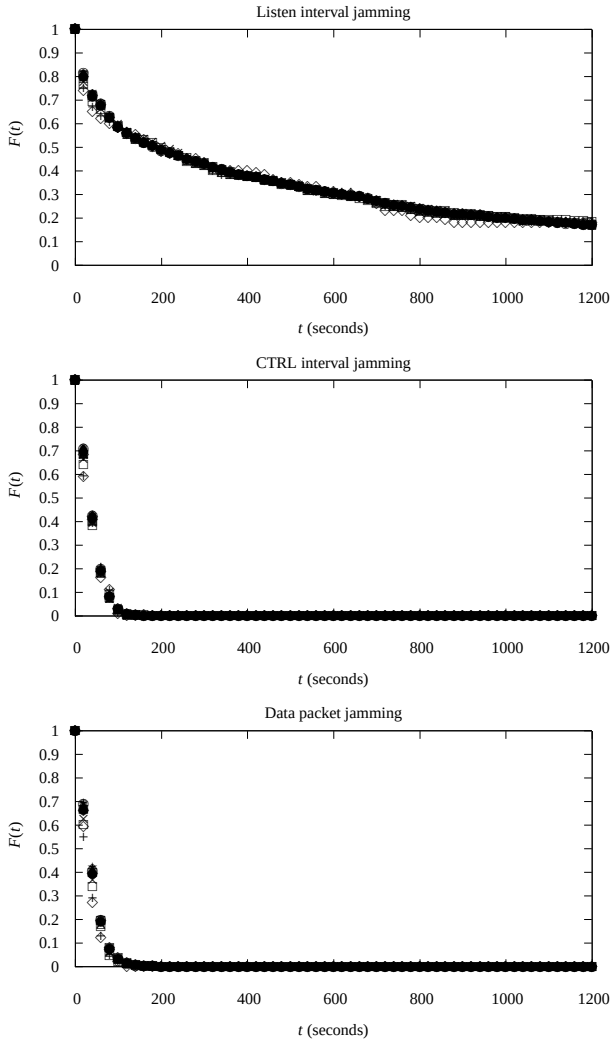


Figure 7.4: $F(t)$, the fraction of jamming motes that are still unsynchronized at time t, plotted against t, where number of runs = 20, number of jamming motes = 100, $D = 7$.

Rx). In the case of data packet jamming, sensor nodes not only waste *less* energy compared with when they are attacked by listen/CTRL interval jammers, they also waste less energy compared with when they are not under attack (negative attrition rates in Figure 7.3(a) and Figure 7.3(b)). There are two reasons:

1. The energy for retransmissions is typically less than the energy for listening for the entire sleep interval. To see this numerically, let P_{Rx} be the power consumed by listening, then the power consumed by transmitting is $(2400/960)P_{Rx} = 2.5P_{Rx}$. The energy consumed by listening for the entire sleep interval is $T_S P_{Rx}$. This is typically less than the energy consumed by sending a data packet t_p seconds long which is $t_p \times 2.5P_{Rx}$,

because $T_S \gg t_p \times 2.5$.

2. Sensor nodes go to sleep whenever they fail to receive acknowledgements. This uses less energy than if they continue listening (as would happen if they are attacked by listen/CTRL jammers), or transmitting (as would happen if they are not under attack and they have data to send).

It is for this reason that increasing the network density does not boost the attrition rate of 50 data packet jammers like it does for 50 listen/CTRL interval jammers, by bringing more jamming motes within the radio range of the sensor nodes (Figure 7.3). Notice that increasing the network density also does not boost the attrition rate of an attacker network whose density is already the same as the sensor network's (e.g. 100 jamming motes vs 100 sensor nodes), due to the saturation of the effect of attacks.

Effort ratio

It is not surprising that listen interval jamming has the highest effort ratio, since it transmits for the longest period of time. Notice that the effort ratio of listen interval jamming increases with network density. This is because as the network gets denser, some jamming motes are able to synchronize themselves with the schedule sooner, but as these motes start jamming the listen interval, other motes would not have the chance to capture SYNC packets, to synchronize themselves with the schedule. It is also for this very reason that the difference between 50 and 100 listen interval jammers is larger than the those between 50 and 100 CTRL interval jammers, and between 50 and 100 data packet jammers. In contrast, since CTRL interval jamming does not prevent the transmission of SYNC packets, not only are more jamming motes able to start their jam-sleep cycle sooner, but also are more jamming motes able to follow the schedule. As a result, the effort ratio of CTRL interval jamming decreases with network density. For data packet jammers, as the network gets denser, each jamming mote gets more neighbors, and hence more data packets to jam (while CTRL interval jamming motes get the same CTRL interval to jam). This explains the upward trend of the curves for data packet jammers.

It is perhaps surprising that the effort ratio of data packet jamming would exceed that of CTRL interval jamming starting from $D = 5$ (for both $T_{\text{sim}} = 600$s and 1200s) since data packet jamming is a reactive approach. One reason is that data packet jamming motes, as mentioned, already start jamming before entering their jam-sleep cycle. Another reason has to do with adaptive listening [310]. In the original S-MAC protocol [309], after a node *B* receives a data packet from its neighbor *A*, it has to wait until the CTRL interval of the next cycle, before it can relay the data packet to another neighbor *C*. In the improved version with adaptive listening, the node *C*, upon overhearing *A*'s RTS packet, would as usual sleep when *A* transmits to *B*, but *additionally* schedule to wake up right after the transmission is over, thereby giving *B* the opportunity to relay the data packet to itself in the same cycle. During this adaptive listening phase, when *C* listens for *B*'s potential RTS, *C* might also send an RTS packet if it has any data to send, thereby triggering another wave of adaptive listening. The key point is that since data packet jamming allows RTS packets to pass through, a data packet jammer needs not only to block the data packet from *A* to *B*, but also any potential data packet from *C* and other nodes triggered to perform adaptive listening. As long as there are RTS packets, adaptive listening may well extend deep into the sleep interval. In the case of listen/CTRL interval jamming, most RTS packets are either prevented from being sent or are corrupted by jamming signals. Nonetheless our implementation of data packet jamming

has not been optimized according to Lin et al.'s error correction code-based technique [164], otherwise the effort ratio is expected to be lower.

As expected, comparing Figure 7.3(d) with Figure 7.3(c) shows that when the settling times of the jamming motes are amortized over a longer simulation period, the effort ratios become lower. The amortization effect is most pronounced with listen interval jammers as they have much longer settling times. The important point is that the increasing/decreasing trend of the effort ratios with network density remains the same for different simulation lengths.

All in all, from Equation 7.4, CTRL interval jamming has the highest lifetime advantage, followed by listen interval jamming, and lastly data packet jamming.

7.4 Countermeasure

The most straightforward solution is probably to encrypt every packet, so that jammers cannot deduce the schedule as well as differentiate the type of packets. However if a single encryption key is used for every node, then we have a single-key vulnerability. If we have more than 1 key, each node will have to send a packet encrypted with different keys each time, resulting in drastically higher energy usage. If we use just one schedule, no matter how many keys we use to encrypt the schedule, the attacker would know that schedule once it knows 1 key. If we use more than one schedule, and encrypt a different schedule with a different key, then some nodes would never learn about some schedules, resulting in some inevitable packet collisions. All in all, doing the encryption right is hard.

To counter data packet jamming which has the highest censorship rate, our proposal is to use what we call *data blurting* and *schedule switching*. The idea is that after n times timing out waiting for an ACK, the sender 'blurts' out its data packet without going through the RTS/CTS/data/ACK sequence. Upon receiving the blurted packet, the receiver announces a switch of schedule in the next cycle. After some finite number of cycles, the receiver and its neighbors would synchronize with the new schedule, which is a right shift of the old schedule along the time axis, by the length of 1 listen interval and 1 CTRL interval. We let the receiver and not the blurter announce the schedule switch because letting the blurter do it would result in too many schedule switches, which in turn would cause a vast drop in throughput. The receiver might or might not receive the blurted data packet, but once it receives a blurted data packet, it knows for sure that a schedule switch is necessary. Compare Table 7.1 and Table 7.2 to see how our technique alleviates the effect of data packet jamming, e.g. when $T_{\text{sim}} = 1200$s and $D = 7$, the censorship rate is on average reduced from 99% to 52%. Table 7.2 also shows that the alleviation effect gets better over time, as the jamming motes loose sync with the sensor nodes for a longer period of time. The standard deviations are large because blurted data packets might or might not be successfully received, and it takes a varying number of cycles to synchronize with a new schedule. Note that the technique does not interfere with the normal operation of the network in the absence of attacks.

7.5 Related Work

Link-layer attacks are not new but are certainly not actively studied either. Wood et al. acknowledged in 2002 that no effective defense was yet known against link-layer jamming [300]. Both Ståhlberg [273] and Wood et al. [300] quote how an attacker might keep sending RTS packets to elicit CTS packets from its victims, as an example of *sleep deprivation torture attacks* [268]. We think the attack is irrelevant in the case of S-MAC where nodes go to sleep

Table 7.2: Average censorship rates of data packet jamming with countermeasure implemented (standard deviations in parenthesis).

T_{sim}	Jammer count	Censorship rate R_c (%)				
		D=3	D=4	D=5	D=6	D=7
600	50	43 (31)	50 (29)	57 (27)	64 (31)	61 (28)
	100	64 (21)	55 (20)	66 (29)	68 (32)	72 (23)
1200	50	35 (26)	34 (26)	27 (38)	54 (32)	47 (33)
	100	50 (18)	40 (29)	50 (22)	56 (36)	52 (49)

periodically – it would be long before the energy of the sensor nodes is exhausted, and in contrast it would not be too long before the attacker exhausts its own energy first.

Negi and Perrig [200] have investigated the attack of a jammer that detects and jams RTS packets, as well as sends RTS packets to reserve the largest time interval possible, using the Poisson arrival model. In reality, RTS packets are typically too short to react to in time, jamming the ensuing CTS or data packets is more feasible. The attacker also has to send RTS packets that can pass the integrity check by the normal sensor nodes. Our attackers are not bound by such an assumption.

Konorski [152] has proposed a scheduling policy that addresses the selfish behavior of using small or no contention windows in contention-based protocols, in the context of *single-hop* networks. We are only interested in multi-hop networks. Kyasanur and Vaidya [156] targets selfishness in IEEE 802.11, specifically in the random backoff mechanism. Čagalj et al. [290] investigate the effect of a group of selfish cheaters from a game-theoretic viewpoint. We focus on DoS attacks, in which the purpose of the misbehaving party lies in disruption instead of getting more bandwidth. Different countermeasures are thus required.

7.6 Conclusion

Among all attacks on network protocols of sensor networks, we argue that link-layer attacks are the most relevant, on the grounds of energy efficiency, and given the fact that the correct functioning of any higher-level protocol depends on a working link layer. We have demonstrated both qualitatively and quantitatively how listen interval jamming, CTRL interval jamming and data packet jamming affect S-MAC. Based on our results, we have shown that data packet jamming is better than the other attacks in censorship rate by a small margin. We have also shown that CTRL interval jamming has the best lifetime advantage. As a countermeasure to data packet jamming, we have proposed a new technique based on data blurting and schedule switching. We realize that we need a better approach to reduce the standard deviations of this technique, and also a countermeasure to the other attacks. At the same time, we also plan to report more attacks on S-MAC, particularly in how the attackers might analyze encrypted S-MAC traffic and respond to changes in the schedule, and validation of our simulations in a later paper.

CHAPTER VIII

Energy-Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols

Abstract A typical wireless sensor node has little protection against radio jamming. The situation becomes worse if energy-efficient jamming can be achieved by exploiting knowledge of the data link layer. Encrypting the packets may help prevent the jammer from taking actions based on the content of the packets, but the temporal arrangement of the packets induced by the nature of the protocol might unravel patterns that the jammer can take advantage of even when the packets are encrypted. By looking at the packet interarrival times in three representative MAC protocols, S-MAC, LMAC and B-MAC, we derive several jamming attacks that allow the jammer to jam S-MAC, LMAC and B-MAC energy-efficiently. The jamming attacks are based on realistic assumptions. The algorithms are described in detail and simulated. The effectiveness and efficiency of the attacks are examined. Careful analysis of other protocols belonging to the respective categories of S-MAC, LMAC and B-MAC reveal that those protocols are, to some extent, also susceptible to our attacks. The result of this investigation provides new insights into the security considerations of MAC protocols.

8.1 Introduction

Jamming-style DoS attacks on the physical and data link layer of WSNs have in these few years attracted some attention [4, 289, 303, 304]. In particular, Xu et al. [303] propose 4 generic jammer models, namely (1) the constant jammer, (2) the deceptive jammer, (3) the random jammer and (4) the reactive jammer. A constant jammer emits a constant noise; a deceptive jammer either fabricates or replays valid signals on the channel incessantly; a random jammer sleeps for a random time and jams for a random time; and lastly, a reactive jammer listens for activity on the channel, and in case of activity, immediately sends out a random signal to collide with the existing signal on the channel. According to Xu et al. [303], the constant jammers, deceptive jammers and reactive jammers are effective jammers in that they can cause the packet delivery ratio to fall to zero or almost zero, if they are placed within a suitable distance from the the victims. However these jammers are also *energy-inefficient*, meaning they would exhaust their energy sooner than their victims would if given comparable energy budgets. Although random jammers save energy by sleeping, they are less effective.

Our contribution is to develop jamming attacks that (1) work on encrypted packets, (2) are as effective as constant/deceptive/reactive jamming, and (3) at the same time more energy-efficient than random jamming or reactive jamming. We implement such jamming attacks by exploiting the semantics of the data link layer and show the results quantitatively. The fact that our attacks are applicable to three representative MAC protocols suggests the same attacks are applicable to a wide range of other protocols belonging to the same categories as these protocols'. Our analysis of the attacks provides new insights into the timing consider-

*This chapter is a minor revision of the paper with the same title published in the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005) [5].

ations of MAC protocols with regards to security, and provides hints on which category of protocols provides the best protection against our attacks so far, in the absence of effective countermeasures. The motivation of this work stems from the concern that if an attacker can program and deploy a general-purpose link-layer jamming network that is able to jam any WSN effectively and energy-efficiently, and if a high entry barrier is not maintained for such a low cost attack, a WSN can never in any practical sense be secure. Moreover, link-layer jamming can be used to render higher-level attacks like the sleep deprivation attacks against topology maintenance protocols more energy-efficient [91]. For example, link-layer jamming can be used to block the beacon messages periodically exchanged by neighboring nodes in CCP [294] energy-efficiently, to prevent the victim nodes from entering the SLEEP state. A counter-argument might be that energy efficiency is no concern to powerful attackers, but even powerful jammers come with a finite energy supply and they would advertise their presence and location if they simply blast away with an exorbitant amount of radio waves – this is something a sensible attacker would avoid.

The paper is organized as follows. We start by stating the assumptions on which our attacks are based in Section 8.2. We then describe the attack algorithms in Section 8.3. Section 8.4 describes how the protocols and the corresponding attacks are simulated, and how the results are evaluated. The results are given in Section 8.6. Implications of our work to other protocols are discussed in Section 8.7. Section 8.8 explores some potential countermeasures. Related work is discussed in Section 8.9. Finally Section 8.10 concludes.

8.2 Assumptions

We assume an attacker has two goals: the primary goal is to disrupt the network by preventing messages from arriving at the sink node, and the secondary goal is to increase the energy wastage of the sensors. A sink node is a node that requests for, and hence sinks, information. Our attacks depend on three assumptions: (1) the jammer motes know the preamble used by the victim nodes, (2) the jammer motes can measure the length of a packet, and (3) the jammer motes know what MAC protocol that victim notes are running. A preamble is a bit sequence, usually consisting of alternating 1's and 0's, for training the receiver, and its length depends on the data coding scheme used [244]. Requirement (1) and (2) should be easy to satisfy in practice. Requirement (3) is more demanding but not impractical to satisfy – as our future work, a strategy will be devised to map observed traffic to specific classes of protocols. Note that the jammer motes do not need to know the content of the packets, so our attacks work even if the packets are encrypted. Adding to the significance of our attacks is that the attacker does not need to capture and compromise any existing sensor nodes.

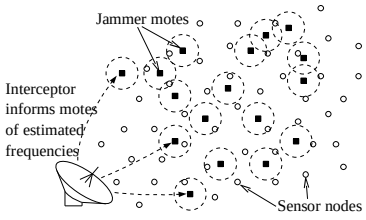


Figure 8.1: Distributed jamming.

We now describe a concrete set of circumstances under which our attack scenario is

applicable.

- When there are no fixed sink nodes to attack, e.g. in directed diffusion [120] where ID-less interests propagate through the network, there is no way for a node to tell where the real sinks are,
- or when it is infeasible for the attacker to deploy its nodes strategically, except perhaps to air-drop them on top of the target network,
- or when the attacker can only estimate where and not how the target network is or would be deployed,
- or when the target network is protected with a link-layer authentication (and optionally encryption) scheme like TinySec [138],

the attacker would find it appealing to distribute its jammer motes among the target WSN and apply our jamming attacks. We are in fact not alone in suggesting this attack scenario [204]. One note of caution though, is that if the target network employs frequency-hopping spread spectrum [9], the attacker would find it necessary to deploy an interceptor to discover the hopping frequencies first before its network of jammer motes can start jamming (Figure 8.1).

Naturally, our attacks are affected by the choice of values assigned to the protocol parameters. Throughout the paper, we pick values for the protocol parameters that are either as realistic or as faithful to what the creators of the protocols recommend as possible, in the absence of a universal consensus and a scientifically rigorous way of deriving these values.

8.3 MAC Protocols for WSNs

Before we describe our attacks, a brief overview of MAC protocols for WSNs is in order. Despite the number of protocols proposed so far, there is still no clear indication of the mechanisms the proposals are converging to [158]. However, since different applications require optimizing different parameters, there will most likely not be a single solution that fits all types of applications. According to Langendoen et al.'s survey [158], WSN MAC protocols can be classified according to (1) the number of channels used, (2) how the intended receiver of a message is notified, and (3) how medium accesses are temporally organized.

In terms of channels, most protocols use only a single channel, e.g. S-MAC [309, 310], LMAC [287] and B-MAC [226].

In terms of message notification, in some protocols, a scheduling algorithm determines when a node listens for messages to minimize energy consumed by idle listening. These type of protocols are typically either more resource-demanding or require architectural support [155, 162, 237]. In other protocols, a node has to determine on its own when to listen for messages. To reduce energy spent on idle listening, they typically employ some form of sleep-listen schedule. Since these protocols are more lightweight and hence more viable for current WSNs, we concentrate on this type of protocols, of which S-MAC, LMAC and B-MAC are, again, well-known examples.

In terms of temporal organization of medium accesses, S-MAC, LMAC and B-MAC belong to different categories. S-MAC divides time into slots, and nodes contend for slots to send packets. LMAC divides time into frames, and each node is allocated a slot in the frame to send packets. B-MAC uses random accesses to the communication medium, i.e. no slots and no frames, to send packets.

Based on the above analysis, S-MAC, LMAC and B-MAC are representative of current WSN MAC protocols, and our jamming analysis in the following will hence be targeted at these protocols. What follows is a brief introduction to each of the protocols.

8.3.1 S-MAC

The design of S-MAC revolves around a periodic sleep-listen schedule. A period is divided into a listen interval and a sleep interval. The listen interval in turn consists of a SYNC interval and a CTRL interval. The sleep interval allows the nodes to sleep in order to reduce the amount of energy spent on idle listening. The ratio of the length of the listen interval to the length of the period is the duty cycle. Lowering the duty cycle based on a fixed period reduces energy usage.

We now describe the operation of S-MAC. When a node A first joins a network, it listens for a whole period. If the channel is clear, A broadcasts its schedule in a SYNC packet, telling its neighbors that it will sleep t_s seconds later, marking the end of A's listen interval. If a node B receives A's schedule before choosing its own, B will adopt A's schedule and after a random delay of t_d seconds, broadcast to tell A and other neighbors that it will sleep at $t_s - t_d$ seconds later. Should B receive A's schedule after broadcasting its own, it adopts both schedules [310]. In this way, A, B and their neighbors are able to synchronize their schedules.

Data packets are mainly sent in the CTRL interval, and may extend into the sleep interval. When broadcast, data packets are sent without the exchange of RTS/CTS packets (RTS = Request-To-Send, CTS = Clear-To-Send). When unicast, RTS/CTS packets are exchanged. Collision avoidance depends solely on carrier sense and the use of network allocation vectors [310].

8.3.2 LMAC

LMAC is a TDMA protocol. In LMAC, time is divided into frames, which are further divided into time slots. In each frame, a sensor node takes control of one time slot (or more, for instance in a variant of LMAC called AI-LMAC [54]). A time slot is further divided into two parts of unequal length: (1) a control part for transmitting a control packet, and (2) a data part for transmitting a data packet. In the time slot it controls, a node always starts by sending out a control packet even if it does not have any data to send. Besides addressing other nodes, the control packet is also necessary for maintaining synchronization. When the neighbors of the node discover by listening to the control packet that they are not the intended receivers, or that the node simply has no data to send, they immediately turn off their receivers and sleep until the next slot. The neighbor addressed by the control packet stays listening. The data packet is transmitted right after the control packet. The absence of RTS/CTS signalling makes LMAC a particularly energy-efficient protocol, however tight time synchronization is required.

8.3.3 B-MAC

The central feature of B-MAC is its preamble sampling scheme, called low power listening (LPL), which is a continuation of a tradition set by El-Hoiydi [79], and Hill and Culler [109]. As one of the main sources of energy wastage in WSNs is idle listening, a simple solution is to listen and sleep periodically according to some duty cycle. The requirement for monitoring-type applications [169] to reduce the duty cycle to 1% (i.e. listen for 1% of an entire cycle) means that the sensor nodes should listen for the briefest time possible. Preamble sampling achieves this by delegating to the transmitter the responsibility of making sure the receiver

receives the packet, in that the transmitter must transmit a preamble that is long enough to be sensed by the receiver which only wakes up for the briefest moment and sleeps most of the time (Figure 8.2). Note that this is different from S-MAC in that the sender and receiver are not synchronized.

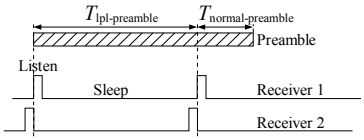


Figure 8.2: Preambles should be long enough such that a receiver listening at the very beginning of the cycle, or at the very end of the cycle would be able to detect the preamble.

Unlike S-MAC or LMAC, B-MAC is only a link protocol, in that it does not stipulate how the communication medium is shared between nodes. However it is reasonable to assume RTS/CTS signalling is used. When RTS/CTS signalling is used, the sender sends an RTS packet with an LPL preamble. Upon detecting this long preamble, the receiver snaps out of the LPL mode, and replies with a CTS packet that has a normal preamble, since the sender is already listening and waiting. The ensuing data packet and acknowledgement packet exhanged between the sender and receiver are all transmitted with a normal preamble. After sending the acknowledgement packet, the receiver returns to the LPL mode.

8.4 Description of Attacks

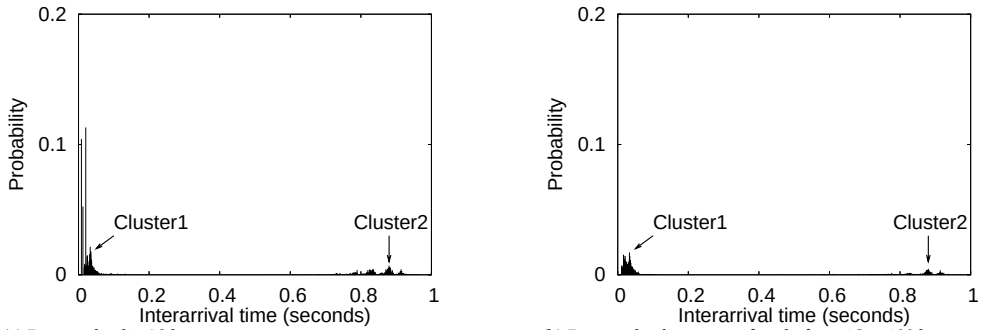
Imagine we are the attacker, the question now is how do we attack a protocol without knowing the content of the packets? Since the attacker is solely interested in jamming data packets, our first observation is that since data packets are longer than control packets, we can focus on jamming long packets. We can do this by sorting packets according to their length and predict when long packets would arrive. This strategy might not work however because (1) data packets might be generated spontaneously, rendering our prediction inaccurate, and (2) data packets are sparse, e.g. 1 packet every 5 minutes from each node [169]. Sparse packets require us to observe for a long time before we get a working prediction model, and offer us little opportunities to readjust our prediction.

A more promising approach is to look at the probability distribution of the interarrival times between packets, i.e. packets of all types. We look at S-MAC, LMAC and B-MAC in turn.

8.4.1 S-MAC

Figure 8.3 shows the probability distribution of the packet interarrival times observed by a node having 6 neighbors that send data to a sink multiple hops away every 5 seconds, in a static network using S-MAC with a period of 930 ms and a duty cycle of 10% (i.e. default values that come with the original S-MAC source code). There are 2 clusters in the graph. Let us call them Cluster1 and Cluster2. Although using different data packet lengths results in different shapes of the clusters (e.g. distinct spikes in Cluster1 in Figure 8.3(a) in contrast to Figure 8.3(b)), the clear separation between the clusters still stands. These two clusters can also be observed even if the nodes are mobile. In spite of suspicion, these clusters are *not* due to the periodic nature of data reporting by the nodes, in fact they are solely the result of the periodic nature of the protocol, in this case S-MAC itself. The explanation is as follows. In S-MAC, packets within a period are closely spaced in time, accounting for the interarrival times

in Cluster1. Two packets from two different periods are more widely spaced because of the sleep interval, and the interarrival time between these two packets falls in Cluster2. Unless the nodes insist on sending only 1 packet every period, which is improbable, it is only natural that Cluster1 has a larger weight, or higher probability, than Cluster2. Observe that Cluster2 has a larger variance than Cluster1. One way to understand this is to compare two Cluster2 interarrival times: the time separation between two SYNC packets of two consecutive periods is large, but the time separation between an acknowledgement packet and a SYNC packet of the *subsequent* period is small, and yet these two time separations belong to Cluster2. Actually, there are other clusters at the further end of the time axis, but their probability is negligible. They have to be filtered out in order for clustering to work. It is reasonable, for example, to expect S-MAC to have a *maximum* period of 1 s (otherwise the latency would be large), thus filtering all interarrival times larger than 1.5 s should eliminate these unwanted clusters.



(a) Data payload is 28 bytes.

(b) Data payload ranges uniformly from 16 to 100 bytes.

Figure 8.3: Probability distribution of packet interarrival times for S-MAC with a period of 930 ms and a duty cycle of 10%. Choices of packet lengths and reporting frequency are detailed in Section 8.5.

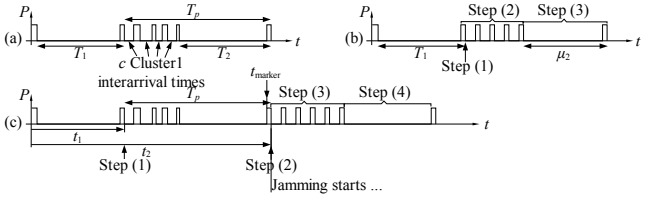


Figure 8.4: Jamming strategy for S-MAC: (a) actual timing, (b) naive version, (c) periodic clustering-based jamming (PCJ).

Our S-MAC attack strategy follows from the following deduction. If according to observation, for every Cluster2 interarrival time, there are c Cluster1 interarrival times, then right after observing a Cluster2 interarrival time, we should expect around c Cluster1 interarrival times (Figure 8.4(a)). Denote the means of Cluster1 and Cluster2 interarrival times as μ_1 and μ_2 respectively. A straightforward strategy is to first collect a reasonable number (e.g. 64) of consecutive interarrival times, and then perform clustering on them. The result of clustering gives us μ_1 and μ_2. The next steps are then to (Figure 8.4(b)):

1. Wait until a single Cluster2 interarrival time, T_1, is observed.
2. Jam with c packets, with a space of μ_1 seconds in between.
3. Sleep for μ_2 seconds.
4. Repeat the cycle starting from step 2.

This strategy however does not work in practice because μ_2 is often not a good approximation of T_2 due to the large variance of Cluster2. An improvement is to estimate T_p instead. Since T_p is a sum of several Cluster1 interarrival times and T_2, and since the variance of Cluster1 is smaller, T_p can be predicted more accurately. Our improved strategy is thus to (Figure 8.4(c)):

1. Wait until a single Cluster2 interarrival time is observed and record arrival time as t_1.
2. Wait for another Cluster2 interarrival time and record arrival time as t_2. Calculate the period $T_p = t_2 - t_1$. Record length of packet in time (not in bytes, as with all packet lengths that appear hereafter) just received as L_p.
3. Set t_{marker} = current time - L_p. Jam with $c - 1$ packets, with a space of μ_1 seconds in between. (Notice where this step starts in Figure 8.4(c).)
4. Sleep until $t_{\text{marker}} + T_p$.
5. Set t_{marker} = current time. Jam with c packets, with a space of μ_1 seconds in between.
6. Repeat the cycle starting from step 4.

Periodic re-estimation is done by repeating the cycle starting from step (1) instead of step (4). We call this approach *periodic clustering-based jamming* (PCJ), and depending on the context we also use PCJ to mean a periodic clustering-based *jammer*.

Additionally, we propose RPCJ, a reactive version of PCJ, by modifying step (3) of PCJ. In step (3) of RPCJ, the jammer records current time t_{marker} as before, but instead of jamming proactively, the jammer listens for a preamble by setting a timer that expires $\max(T_{\text{Cluster1}})$ seconds later, where $\max(T_{\text{Cluster1}})$ is the maximum interarrival time in Cluster1. If a preamble is detected, it jams the ensuing frame, otherwise if the timer expires, that is, if no preamble is detected, the jammer sleeps until $t_{\text{marker}} + T_p$ seconds, before waking up to listen for preambles again. Note that the jammer transmits only random packets, without any preamble.

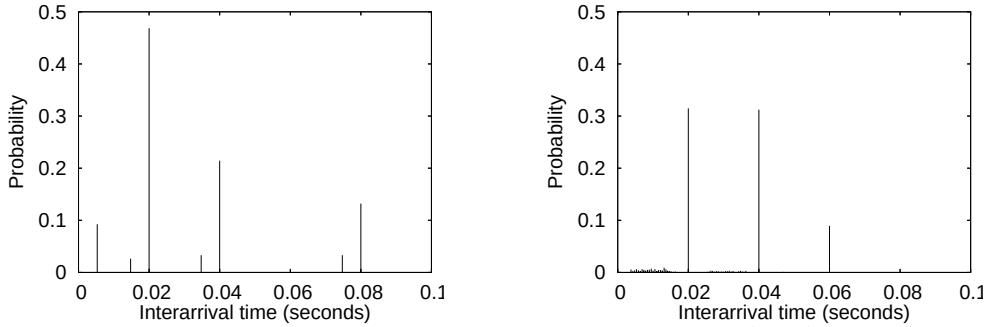
Clustering

The above attack relies on data clustering. A simple algorithm like K-means [179] is sufficient, because of the clear separation between the clusters, and the distinctness of each of the clusters. K-means involves only simple multiplications. A 32-bit floating-point hardware-accelerated multiplication (or division) consumes only 9 CPU cycles on a MSP430F149 [279]. A K-means iteration consists of an assignment step and an update step [179]. Denote the number of clusters as K and the number of data samples as N. The assignment step takes KN multiplications, whereas the update step takes KN multiplications and K divisions (having the same cost as multiplications). Assuming multiplication is the most expensive operation, then the computational complexity of a K-means iteration is $K(2N+1) \approx 4N$ multiplications, taking K as 2. According to simulations, only 2 iterations are usually required, 3 and above

are *rare*. So for example, if $N = 64$, the energy required for 2 iterations is at least 4.4 μJ, or 60% of the energy required to transmit one bit (7.4 μJ) on a CC1000 radio [293] (both MSP430F149 and CC1000 are common components in existing sensor nodes). From the jammer's perspective, the computational cost is likely justified, since the jammer has nothing else to do apart from jamming.

8.4.2 LMAC

Figure 8.5 shows clusters that are clearly spaced out at integral multiples of the slot size – typical of a TDMA protocol. However there might be one or more clusters before the cluster centered on the slot size, depending on the distribution of the data packet length. And unlike S-MAC, the clusters at the higher end of the time scale are not negligible, so the jamming algorithm suggested for S-MAC cannot be applied.



(a) Data payload is 28 bytes.

(b) Data payload ranges uniformly from 16 to 100 bytes.

Figure 8.5: Probability distribution of packet interarrival times for LMAC with a slot size of 20 ms and 20 slots in a frame.

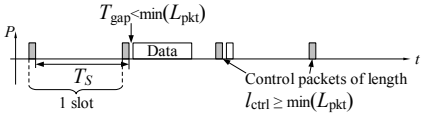


Figure 8.6: Interarrival times in LMAC.

The jammer's objective is to estimate the slot size by calculating the mean of T_S (Figure 8.6), μ_s, and to jam the beginning of every slot. The algorithm is based on two assumptions. **The first assumption** is that T_S can indeed be observed, i.e. the probability that at least two occupied slots are consecutive is at least larger than 0.5. Using elementary combinatorics, it can be shown (in the Appendix) that

$$\Pr\{\text{at least two occupied slots are consecutive}\} = \begin{cases} 0 & \text{if } 0 \leq n < 2 \\ 1 - \frac{\binom{s-n}{n}}{\binom{s-1}{n}} & \text{if } 2 \leq n \leq \frac{s-1}{2} \\ 1 - \frac{2\binom{s-n-1}{n-1}}{\binom{s}{n}} & \text{if } n = \frac{s}{2} \ (s \text{ is even}) \\ 1 & \text{if } \frac{s}{2} < n \leq s \end{cases} \tag{8.1}$$

In Equation 8.1, s ($s \geq 4$) is the total number of slots in a frame, and n ($0 \leq n \leq s$) is the

number of occupied slots in a frame. When s is even and $n = \frac{s}{2}$, the probability is always larger than 0.5. In other words, for a given s, a node is more likely than not to observe at least two consecutive occupied slots, if n, the number of occupied slots, at least satisfies Equation 8.2:

$$\frac{\binom{s-n}{n}}{\binom{s-1}{n}} = \prod_{j=1}^{n-1}\left(1 - \frac{n}{s-j}\right) < 0.5 \tag{8.2}$$

For example, if $s = 20$, the least n that satisfies Equation 8.2 is $n = 4$. Given that most practical WSNs are dense [12], this requirement is almost certainly satisfied.

The second assumption is that the shortest data packet *might* be shorter than a control packet, but the longest data packet *must* be longer than a control packet, i.e.

$$\min(L_{\text{pkt}}) \leq l_{\text{ctrl}} < \max(L_{\text{pkt}}) \tag{8.3}$$

where L_{pkt} is the random variable representing packet lengths (regardless of packet type), and l_{ctrl} is the fixed length of a control packet. For example, a control packet takes 15 bytes in an optimized implementation (or 23 bytes in our unoptimized implementation), a data packet header takes 7 bytes, a message authentication code takes 4 bytes, so if a data packet payload is less than $15 - 7 - 4 = 4$ bytes (or 12 bytes in our unoptimized implementation), the corresponding data packet would be shorter than a control packet. The control packets in LMAC are longer than those in S-MAC because they contain more information like the slot occupancy vector, the number of hops to the gateway etc. This assumption unfortunately bars us from estimating T_S by just measuring the interarrival time between two neighboring shortest packets, since we can no longer be sure if the two neighboring shortest packets are both control packets.

The algorithm itself consists of the following steps:

1. Suppose the observed packets are P_0, P_1, ... Denote the interarrival time between packet P_i and packet P_{i+1} as t_i, and the length of packet P_i as l_i ($i = 1, 2, \ldots$). Store $t_1, t_2, \ldots$ in the ordered set $\mathcal{T}$, and $l_1, l_2, \ldots$ in the ordered set $\mathcal{L}$. Had there only been data packets and no control packets, the jammer's job would have been easier, since the slot size is then simply

   $$\mu_S = t_i - l_{i+1} + l_i \tag{8.4}$$

   But there are control packets, so the jammer has to continue as follows.

2. This step is based on two observations. The first observation is that T_S has to be large enough to accommodate both a control packet and a data packet, i.e.

   $$T_S > L_{\text{ctrl}} + \max(L_{\text{pkt}}) \geq \min(L_{\text{pkt}}) + \max(L_{\text{pkt}}) \tag{8.5}$$

   The last inequality is the result of Equation 8.3. Equation 8.5 gives a lower bound for the slot size. The second observation is that a slot can accommodate at most 2 packets, so it is always smaller than the sum of 3 contiguous interarrival times, i.e.

   $$T_S < \min(t_i + t_{i+1} + t_{i+2}) \qquad \text{where } 0 \leq i \leq |\mathcal{T}| - 2 \tag{8.6}$$

   Equation 8.6 gives an upper bound for the slot size. Denote the lower bound and the upper bound respectively as a_0 and a_m. Set $a_0 = \min(\mathcal{L}) + \max(\mathcal{L})$ and $a_m = \min(t_i + t_{i+1} + t_{i+2})$.

3. Compute the probability mass function of the interarrival times in the interval $[a_0, a_m)$. For this purpose, we can quantize the time interval $[a_0, a_m)$ into m millisecond-strips, $[a_0, a_1)$, ..., $[a_{m-1}, a_m)$, as practical values for the slot size are in milliseconds, and count the number of interarrival times that fall within each strip. Denote the strip with the highest count, i.e. the highest probability mass, as $[a_i, a_{i+1})$ $(0 \leq i < m)$.

4. If a higher accuracy is desired, set $a_0 = a_i$ and $a_m = a_{i+1}$ and repeat the previous step with a finer time resolution. If not, set μ_S, the estimated slot size, as the mean of the interarrival times that lie in $[a_i, a_{i+1})$. If t_i is the time closest to μ_S, set μ_L, the estimated control packet length, as l_i. Cautionary note: this estimate might be wrong if the probability given by Equation 8.1 is not high enough. For example, if in Figure 8.5, the probability density at 40 ms (twice the slot size) is higher than the probability density at 20 ms (the slot size), i.e. two occupied slots are more likely to be separated by an unoccupied slot than to be consecutive, the estimate is double the real slot size. However this tends to happen only at the fringe of the network, where the network density is lower. Furthermore, if two occupied slots are indeed more likely to be separated than consecutive, the jammer would not miss much by jamming every two slots instead of every slot.

5. Listen for a packet that is of size μ_L. Once received, transmit a jamming packet, and sleep until time = current time - μ_L + μ_S. It is true that a packet of size μ_L might or might not be a control packet, in view of Equation 8.3. If the received packet is indeed a control packet, then the jammer is able to synchronize neatly with the LMAC schedule, allowing it to jam the control packet of every slot. If the received packet is not a control packet however, the jammer's schedule is offset by at least $L_{\text{ctrl}} + \mu_L$, but it is still able to jam the data packet, if there is any, of every slot.

6. Wake up, transmit a jamming packet, and sleep until μ_S seconds later. Repeat this step until when periodic re-estimation is required, in which case go to step (1).

We call this algorithm *periodic slot-based jamming* (PSJ). In the reactive version of the algorithm (RPSJ), the jammer listens for a preamble before jamming, instead of jamming proactively.

8.4.3 B-MAC

The probability distribution of packet interarrival times of B-MAC does show some clusters but they cannot be taken advantage of because B-MAC uses a periodic cycle only for *listening*, and not sending – we cannot periodically jam something that is not periodically sent. However it is exactly this periodic listening that the jammer can take advantage of to save energy. Since a B-MAC has to listen every, say 10 ms, for a valid preamble, the jammer can be sure that if it samples the RSSI every 10 ms, it would be able to hear whatever preamble is being sent. The jamming strategy is hence to find out the check interval the victim nodes are using. This can be achieved by finding out the length of the longest observed preamble. Assuming this observed LPL preamble is $T_{\text{lpl-preamble}}$ seconds long, and the length of the normal preamble is $T_{\text{normal-preamble}}$, according to Figure 8.2, the jammer should sample the channel every $(T_{\text{lpl-preamble}} - T_{\text{normal-preamble}})$ seconds. The jammer can either guess a value for $T_{\text{normal-preamble}}$, or listen to the channel for the shortest preamble used. If the jammer chooses to guess, to be on the safe side, the jammer should choose a value that is slightly larger than

the typical value, which is three to four bytes [244], so that the jammer would sample the channel slightly more frequently than the victim nodes do. We call this approach *LPL-based jamming* (LPLJ).

In the next two sections, we will explain how the attacks are simulated before presenting the results. Then we will explore some potential countermeasures.

8.5 Simulation and Evaluation Model

All protocols, attacks and countermeasures are simulated using the OMNeT++ framework (`www.omnetpp.org`). A simulation consists of a sink node, $(N_r - 1)$ router nodes, N_s source nodes and N_j jammer motes, all capable of a radio range of r, and located in a square $l \times l$ area. The sink is positioned in the middle of the area, whereas the router nodes are pseudorandomly placed at most r from the sink. Both the source nodes and the jammer motes are pseudorandomly placed more than r away – this is to avoid the jammer motes having direct effect on the sink, giving the jammer motes an unfair advantage and to allow us to investigate the effect of jamming on the routing of information from the sources to the sink. We require the *node density* to be uniform across the simulation area, i.e. $\frac{1+N_r-1}{\pi r^2} = \frac{N_r+N_s}{l^2}$, or $l = r\sqrt{\left(1+\frac{N_s}{N_r}\right)\pi}$, to avoid the peculiarities of any specific non-uniform topology having an effect on jamming. In the absense of jammer motes, the *network density* [316] is then $D = \frac{(1+N_r-1+N_S)\pi r^2}{l^2} = N_r$, i.e. equivalent to the number of sink and router nodes. To simulate attacks, the jammer motes are activated 10 seconds after the sensor network starts operating, to allow the sensor nodes to finish discovering their neighbors and settle down into a steady state before the jamming starts, thereby simulating the attack scenario described in Section 8.2. The total simulation time is T_{sim} virtual seconds. Every experiment is run I times, with different seeds each time. The network topologies are simulated as static. The values of the parameters are summarized in Table 8.1 and are chosen to satisfy the constraints of memory and time available for simulations (maximum 900 MB of RAM and 2 actual minutes per run).

Table 8.1: Simulation parameters and notations.

MAC	N_s	T_{sim}	I
S-MAC	40	600	10
LMAC	20	200	10
B-MAC	20	200	10

N_s = number of source sensor nodes
T_{sim} = simulated time in seconds
I = number of runs
N_j = number of jammer motes (set to $0.75N_s$ or N_s)
D = network density (set to 15 or 20)
N_r = number of sink and one-hop neighbors of the sink

On the application layer, the sink node broadcasts an interest once it found a neighbor but it only broadcasts the interest once throughout the simulation. The source nodes each broadcast a matching data every 5 seconds, as an approximation of a network with moderately fast-changing data [54]. The data packet payload ranges uniformly from 16 bytes to 100 bytes. The minimum payload corresponds to a TinyDiffision [196] payload of 2 attributes (the least number of attributes). The maximum payload is a popular choice [226, 285], and it corresponds to a TinyDiffusion payload of 23 attributes. While a uniform distribution of packet sizes is not realistic, it serves as a 'base case' that allows us to investigate the capability of jammers in reaction to a wide range of packet lengths.

On the network layer, TinyDiffusion [196], faithfully ported from TinyOS (`tinyos.sf.net`), is used for S-MAC and B-MAC. LMAC has a simple built-in routing algorithm, and so LMAC interfaces directly with the application layer.

On the data link layer, S-MAC is simulated with *adaptive listening* [310], with a period

of 930 ms and a duty cycle of 10%. The code is also faithfully ported from TinyOS. LMAC is simulated with a fixed slot size of 20 ms (a little more than enough to fit 100-byte payloads) and 20 time slots per frame (suitable for a network density of 20), using the same codebase from our previous work [287]. The B-MAC code is built on top of the LPL code from TU Delft's MAC simulator [158]. Following Polastre et al.'s choice [226], we use an RSSI sampling time of 350 μs and a check interval of 100 ms for B-MAC. We also implement RTS/CTS signalling on top of the core B-MAC protocol. Both S-MAC and B-MAC use a contention time of 41 ms (the default given by the original S-MAC source code).

On the physical layer, the radio characteristics follow those of RFM TR1001 [243]. The ratio between the power consumption in sleep, Rx and Tx mode is 1:960:2400. Switching times are taken into account. 8-to-12 bit data encoding scheme [241] is assumed, so an encoded data is 1.5 times the size of the original raw data. Denote T_{preamble} as the time required to transmit a preamble, T_{byte} the time required to transmit one byte and L_{pkt} the number of bytes in a packet on the data link layer. The total length of a frame (i.e. packet on the physical layer) in time is then given by

$$T_{\text{frame}} = T_{\text{preamble}} + (1 + 1.5L_{\text{pkt}})T_{\text{byte}} \tag{8.7}$$

The extra 1 byte in the parenthesis is to account for the start byte. In the simulation, $T_{\text{preamble}} = 5T_{\text{byte}}$ and $T_{\text{byte}} = 10/115200$. The extra 2 bits in T_{byte} is to account for 1 start bit and 1 stop bit at the beginning and at the end of a byte.

To simulate jamming, the jammer emits a random packet that is *at least* T_{preamble} seconds long if the jamming starts from the start of the preamble. If the jamming starts from the end of the preamble as is the case with all types of reactive jamming, the jamming packet is only T_{byte} seconds long. It is assumed that the integrity of a packet is protected by a message authentication code, and a corrupted bit is enough to nullify the validify of the packet, therefore corrupting one byte, or 8 bits, should be sufficient to corrupt the whole packet. The width of the jamming pulse has a large impact on the simulation outcomes.

Some aspects of the jammers are as follows. The random jammer is simulated to sleep, and jam for a time uniformly distributed between 1 ms and 500 ms. There are several tunable parameters for PCJ, RPCJ, PSJ and RPSJ. For example, all PCJ and RPCJ implementations start with a minimum sample size of 64 interarrival times, and readjust their estimation every 8 periods. Tuning these parameters allows the attacker to dynamically adjust its behavior, but the effect of such tunings are left to future investigation. The fact that jammer motes have limited buffer for storing interarrival times and packet lengths is realistically reflected in the implementations. The clustering operations are directly translatable to hardware implementations.

Among the things that are not simulated are (1) processing delays, (2) interference and (3) the gray area effect [318]. We will discuss processing delays in relation to our results later. As for interference, ideally the SNR model of Reijer et al. [240] should be used. However since the model takes into consideration all nodes in the network other than the sender for *every* transmission, it demands far greater computational resources than what is required of the conventional, circular model that only considers all nodes in the radio range of the sender. To see how we can implement the more accurate SNR model efficiently is our future work. As for the gray area effect, so far there is still no comprehensive theoretical model for simulating it. All in all, these are some of the many simplifications that are conventionally applied in simulations, as is the case that simulated radio ranges are circular/spherical by convention – a departure from actual measurements [240].

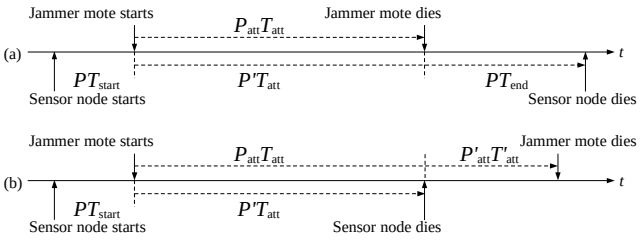
8.5.1 Metrics

Following our previous work [4], we evaluate how *effective* an attack is by measuring primarily the *censorship rate* R_c and secondarily the *attrition rate* R_a. R_c measures the fraction of messages blocked. Let M be the number of messages arriving at the sink in the absence of attacks, and M' be the number of messages arriving at the sink in the presence of attacks, then

$$R_c = (M - M')/M \tag{8.8}$$

R_a measures the fraction of additional energy the sensor network has to spend in the presence of attacks. Let E be the amount of energy spent when there is no attack, and E' be the amount of energy spent when there is attack, then

$$R_a = (E' - E)/E \tag{8.9}$$



P = power consumed by a sensor node when not under attack
P' = power consumed by a sensor node when under attack
P_{att} = power consumed by a jammer mote when attacking
P'_{att} = power consumed by a jammer mote when not attacking
T_{att} = duration of a jammer mote's attack
T'_{att} = remaining lifetime a jammer mote after finishing attack
T_{start} = time between when the sensor node starts operating and when jamming begins
T_{end} = time between when jamming ends and when sensor node dies

Figure 8.7: Calculation of lifetime advantage: (a) if the jammer mote dies before the sensor node, or (b) otherwise.

To evaluate how *energy-efficient* an attack is, we use the *effort ratio* R_e, defined as the ratio of the attacker's per-node energy expenditure to the sensor network's per-node energy expenditure when not under attack. We can compare the effort ratio of a random jammer and the effort ratio of a reactive jammer as follows. Given the same target protocol, if the power consumption ratio between the sleep, Rx and Tx mode is $1 : \rho : \tau$, a reactive jammer that jams j ($j < 1$) of the time has a *higher* effort ratio than a random jammer only when Equation 8.10 is satisfied:

$$j > \frac{1}{2}\left(1 + \frac{1-\rho}{\tau-\rho}\right) \tag{8.10}$$

Plugging in the values $\rho = 960$ and $\tau = 2400$ as given in Section 8.5, we get $j > 17\%$. This relation will be used later.

Using R_a and R_e, we can calculate the *lifetime advantage* R_l of a jammer mote over a sensor node, that is how long a jammer mote can live compared to a sensor node. To derive an expression for R_l, the following assumptions are used: (1) a jammer mote has the same total amount of energy as a sensor node, and (2) the power usage is constant. There are two

Table 8.2: Average censorship rates and lifetime advantages of jamming attacks against S-MAC, LMAC and B-MAC (results are rounded, standard deviations in parenthesis).

D	$\frac{N_j}{N_s}$	S-MAC						
		Rand.	React.	LIJ	CIJ	DPJ	PCJ	RPCJ
Censorship rate R_c (%)								
15	0.75	92 (15)	93 (22)	48 (101)	93 (5)	79 (17)	75 (17)	59 (17)
	1.00	99 (1)	100 (0)	75 (36)	94 (5)	87 (8)	83 (8)	71 (8)
20	0.75	99 (2)	95 (14)	64 (63)	91 (6)	83 (13)	83 (9)	69 (11)
	1.00	100 (0)	100 (0)	89 (22)	94 (5)	91 (6)	83 (7)	76 (10)
Lifetime advantage R_l (%)								
15	0.75	35 (2)	17 (1)	124 (45)	188 (8)	35 (3)	50 (3)	43 (4)
	1.00	37 (2)	17 (1)	116 (28)	196 (11)	35 (2)	47 (3)	43 (2)
20	0.75	35 (1)	17 (1)	124 (31)	169 (11)	35 (4)	46 (3)	42 (3)
	1.00	37 (1)	17 (1)	113 (27)	173 (11)	34 (2)	44 (3)	43 (3)

D	$\frac{N_j}{N_s}$	LMAC				B-MAC		
		Rand.	React.	PSJ	RPSJ	Rand.	React.	LPLJ
Censorship rate R_c (%)								
15	0.75	78 (13)	94 (5)	96 (3)	80 (9)	85 (19)	93 (9)	93 (9)
	1.00	86 (10)	96 (5)	95 (4)	88 (8)	97 (9)	99 (2)	99 (2)
20	0.75	70 (18)	97 (4)	94 (5)	82 (15)	93 (11)	97 (5)	97 (5)
	1.00	91 (7)	97 (5)	97 (1)	91 (9)	97 (5)	99 (3)	99 (3)
Lifetime advantage R_l (%)								
15	0.75	14 (1)	17 (1)	32 (6)	41 (7)	24 (5)	22 (4)	161 (50)
	1.00	14 (1)	17 (1)	31 (4)	35 (3)	24 (5)	22 (3)	139 (31)
20	0.75	14 (1)	16 (1)	32 (5)	37 (4)	20 (3)	20 (3)	134 (37)
	1.00	14 (1)	15 (1)	29 (5)	34 (4)	20 (3)	19 (3)	126 (29)

scenarios. Equation 8.11 is for the scenario where the jammer mote dies first (Figure 8.7(a)).

$$R_l = \frac{T_{\text{att}}}{T_{\text{att}} + T_{\text{start}} + T_{\text{end}}} = \frac{P}{P - P' + P_{\text{att}}} = \frac{1}{R_e - R_a} \tag{8.11}$$

Equation 8.12 is for the scenario where the sensor node dies first (Figure 8.7(b)).

$$\begin{aligned} R_l &= \frac{T_{\text{att}} + T'_{\text{att}}}{T_{\text{start}} + T_{\text{att}}} = \frac{P}{P'_{\text{att}}} + \left(1 + \frac{P' - P_{\text{att}} - P}{P'_{\text{att}}}\right) \frac{T_{\text{att}}}{T_{\text{start}} + T_{\text{att}}} \\ &\approx 1 + \frac{P' - P_{\text{att}}}{P'_{\text{att}}} \geq 1 + \frac{P' - P_{\text{att}}}{P_{\text{att}}} = \frac{1 + R_a}{R_e} \end{aligned} \tag{8.12}$$

In Equation 8.12, the $\approx$ relation is because $T_{\text{start}} \lll T_{\text{att}}$; the $\geq$ relation is because $P'_{\text{att}} \leq P_{\text{att}}$ for a jammer mote that has nothing more to jam, consistent with our assumptions of the attack model. Equation 8.12 gives only the lower bound but in cases where the jammer mote outlives the sensor node, knowing the lower bound is good enough.

8.6 Results

The censorship rates and lifetime advantages of the various attacks are given in Table 8.2. For readability's sake, the figures for attrition rate and effort ratio are omitted. We start by making some general observations of the results. First, the censorship rates, both the means and the standard deviations, improve with N_j/N_s and D (defined in Table 8.1). This is inline with intuition: more jammer motes have greater jamming effect, and the more sensor nodes a jammer mote has as neighbors, the sooner the jammer mote can synchronize with the S-MAC/LMAC schedule, or determine the preamble length in B-MAC's case. Note that in all

simulated cases, $N_j < N_s + N_r = N_s + D$. Therefore further increase in censorhip rate can be expected if we increase the number of jammer motes, N_j, to the total number of sensor nodes, $N_s + N_r$.

The second general observation is that our jamming algorithms outperform random jamming and reactive jamming in lifetime advantage as expected since random jamming is not a targeted effort, and reactive jamming consumes energy constantly in listening.

Thirdly, although it appears in Table 8.2 that random jamming is not as effective against LMAC and B-MAC as it is against S-MAC, the results are to be interpreted with caution. In LMAC's case for example, the jammer's random sleep interval, uniformly distributed between 1 ms and 500 ms (25 times the slot size), is most of the time wide enough to allow many data packets to pass through. By reducing the jammer's random sleep interval, we can get the censorhip rates equally close to 100% for all S-MAC, LMAC and B-MAC. We do not customize the random sleep interval for each of the protocol in our simulations however, because as the random sleep interval gets smaller, the jammer consumes more energy switching between the sleep mode and the transmission mode – this switching energy becomes a dominant component of the overall energy consumption, increasing the jammer's effort ratio. This is to say that although we know higher censorship rates can be achieved by customizing the random jam/sleep interval according to the target protocol, we do not, because the result only makes random jamming more effective than it already is now and not any more efficient. Finishing our general observations, we now analyze the results in more detail below.

8.6.1 S-MAC

We compare random jamming, reactive jamming with the following energy-efficient attacks: listen interval jamming (LIJ), control interval jamming (CIJ), data packet jamming (DPJ), periodic cluster-based jamming (PCJ) and reactive periodic cluster-based jamming (RPCJ). LIJ, CIJ and DPJ are three jamming algorithms we introduce in our previous work [4] that work on unencrypted packets. As their names imply, LIJ jams the listen interval of a S-MAC schedule; CIJ jams the control interval; DPJ waits for a CTS packet and jams the ensuing data packet. We classify LIJ, CIJ and DPJ as detailed knowledge attacks, PCJ and RPCJ as minimal knowledge attacks. It should be interesting to compare the latest minimal knowledge attacks with existing detailed knowledge attacks.

Censorship rate Random jamming and reactive jamming have the highest censorhip rates. Among the energy-efficient attacks, CIJ has the highest censorship rate. LIJ has large standard deviations because in LIJ, the SYNC packets are jammed – jammer motes that have too few sensor nodes or too many fellow jammer motes as neighbors would take a long time, if at all, to get hold of two SYNC packets to synchronize with the S-MAC schedule – making the censorship rate of LIJ highly topology-dependent. The censorship rate of DPJ is lower than CIJ's because of the adaptive listening mechanism in S-MAC [4]. PCJ and RPCJ are not as effective as the detailed knowledge attacks (LIJ etc.), but their censorship rates are still substantial, at 83% and 76% respectively when $N_j/N_s = 1$ and $D = 20$. RPCJ is worse than PCJ because in the proactive approach of PCJ, the victim nodes go to sleep when they fail to access the jammed medium, whereas RPCJ misses more packets as a result of misalignment with the S-MAC schedule.

Lifetime advantage Table 8.2 agrees with the result of our previous work [4] that among the detailed knowledge attacks, CIJ has the highest lifetime advantage, followed by LIJ. The minimal knowledge attacks, PCJ and RPCJ, have, not surprisingly, lower lifetime advantages. DPJ, Random jamming and reactive jamming have the lowest. This means, in

the absence of detailed knowledge, PCJ and RPCJ are genuine threats.

At high network density, the lifetime advantages of PCJ and RPCJ are comparable. The lifetime advantage of PCJ however decreases with network density, because as a jammer gets more neighbors, it fakes more transmissions, incurring a higher effort ratio. The lifetime advantage of RPCJ on the other hand hardly changes with network density, because an RPCJ jammer spends more time listening than transmitting, but the time spent on listening is roughly the duration of the listen interval, which does not change with network density.

A random jammer transmits for half of the time, and sleeps for half of the time. A reactive jammer transmits *at most* 10% of the time, and listens for at least 90% of the time, due to the 10% duty cycle. Based on these values alone, comparing the the energy consumptions using Equation 8.10 tells us that reactive jamming has a lower effort ratio. However, random jamming has a far higher attrition rate, because it makes some victim nodes stay in the backoff state in the listening mode throughout the sleep interval, and therefore has a higher lifetime advantage than reactive jamming does.

To summarize, the high censorhip rate and high lifetime advantage of CIJ indicates the importance of link-layer encryption. However encryption alone is insufficient as the temporal arrangement of the packets induced by the nature of the protocol still allows PCJ and RPCJ to be effective and energy-efficient.

8.6.2 LMAC

We compare random jamming, reactive jamming with PSJ and RPSJ.

Censorship rate The censorship rates of PSJ are impressively close to those of random and reactive jamming. RPSJ is worse than PSJ, but both PSJ and RPSJ are more effective against LMAC than PCJ and RPCJ are against S-MAC, because the slot size can be estimated more accurately than the period of an S-MAC schedule.

Lifetime advantage Both PSJ and RPSJ have twice the lifetime advantages of random jamming and reactive jamming. Looking more carefully, RPSJ has a higher lifetime advantage than PSJ. This is because not all slots in a frame are necessarily occupied, and when a slot is unoccupied, the energy RPSJ spends on listening is lower than the energy PSJ spends on transmitting.

As the network becomes denser, more slots in a frame are occupied, the effort ratios of both PSJ and RPSJ become higher, hence their lifetime advantages decrease with network density. This trend should stop when all the slots in a frame are occupied.

Random jamming does not make LMAC nodes monitor the channel more than it usually does, unlike the case with S-MAC – the sensor nodes only listen for at most a small fraction of the slot size, and as packets are jammed, less energy is used on propagating the packets to the sink, resulting in a negative attrition rate. One note of caution though: had the jamming been allowed to start *before* the LMAC nodes synchronize with each other, the results would have been different, because then the LMAC nodes would have had to constantly listen for broadcast schedules, and this would have resulted in a large positive attrition rate, and hence lifetime advantage. Since reactive jamming transmits at most 2 bytes (one byte to jam a control packet, another byte to jam a data packet), or 174 μs per 20 ms slot, it does not satisfy Equation 8.10 and hence has a lower effort ratio than random jamming. In fact, reactive jamming will only have a higher effort ratio when the slot size is 1 ms according to Equation 8.10 which is unrealistic. With their attrition effect being similar, reactive jamming has a higher lifetime advantage than random jamming.

To summarize, PSJ and RPSJ have high censorship rates and lifetime advantages. Coupled with ease of implementation, they are genuine threats to LMAC even when the packets are encrypted.

8.6.3 B-MAC

We compare random jamming, reactive jamming with LPLJ. LPLJ is by design reactive jamming with optimized listening, so it is only intuitive that LPLJ has similar censorship rates as reactive jamming's, but far higher lifetime advantages than reactive jamming's. The lifetime advantage of LPLJ however decreases with network density due to the following reason. As the network gets denser, a jammer mote gets not only more sensor nodes but also more jammer motes as its neighbors. More neighboring sensor nodes means more packets to jam. More neighboring jammer motes means staying awake more often because whenever a signal is detected on the channel, a jammer mote always stays awake for a while to listen for a valid preamble. Consequently, a jammer mote has a higher effort ratio in a denser network, and according to Equation 8.11 and Equation 8.12, the lifetime advantage becomes lower. The lifetime advantages of random jamming are comparable to reactive jamming's, i.e. significantly lower than those of LPLJ.

To summarize, since LPLJ is trivial to implement and yet allows the jammer motes to live as long as, or longer than the victim nodes, it is a devastating threat to B-MAC even when the packets are encrypted.

8.7 Implications to Other Protocols

We now look at the implications of the above findings to other protocols. As explained in Section 8.3, we only concentrate on the MAC protocols that (1) use a single channel, and (2) listen instead of follow some schedule to receive messages. Among these protocols, there are protocols that use (1) slots, (2) frames or (3) random access to organize medium accesses, using Langendoen et al.'s taxonomy [158]. Examples that use slots, frames and random access are S-MAC, LMAC and B-MAC respectively, which we have just investigated. We now look at other protocols that belong to each of the three categories, starting with slot-based protocols. We pick these protocols from Langendoen et al.'s extensive survey [158].

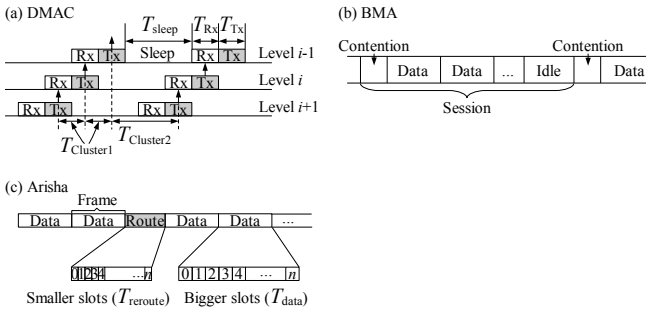


Figure 8.8: Medium access organization in (a) DMAC, (b) BMA, and (c) Arisha.

8.7.1 Slot-Based Protocols

Slot-based protocols include T-MAC [285] and DMAC [166]. Since T-MAC is derived from S-MAC, PCJ and RPCJ are applicable to T-MAC. The fact that T-MAC has a dynamic duty

cycle offers some relief because even though PCJ and RPCJ are able to adapt to the dynamic duty cycle through periodic readjustments, they would be less effective and less efficient against T-MAC than against S-MAC due to the need for more frequent readjustments.

Like T-MAC, DMAC is a slotted protocol that uses a dynamic duty cycle, but unlike T-MAC, it offsets the schedule of a node depending on the number of hops the node is away from the sink, in order to minimize latency (Figure 8.8(a)). For example, if the node is i hops away, its schedule is offset by $+T_{\text{Rx}}$ compared with the schedule of a node $i+1$ hops away. The following cluster pattern should emerge on the probability distribution of the packet interarrival times: Cluster1 has a mean of T_{Tx} and Cluster2 has a mean of T_{sleep}. Based on these clusters, PCJ and RPCJ can then be applied.

8.7.2 Frame-Based Protocols

Frame-based protocols include PACT [211], Arisha [18], TRAMA [237], BMA [162] and SS-TDMA [155]. These are TDMA protocols. The primary means of jamming these protocols is by way of estimating the slot size. To estimate the slot size, the jammer needs to be able to observe two consecutive slots, that is, the number of slots and the number of occupied slots in a frame need to satisfy Equation 8.2. This requirement is typically satisfied as explained in Section 8.4.2 and will not be mentioned again in the discussion below.

We start with SS-TDMA. SS-TDMA relies on the nodes being arranged in a rectangular or hexagonal rid. Due to the lack of control packets, SS-TDMA is easier to attack than LMAC since Equation 8.4 alone allows us to estimate the slot size.

Since both PACT and BMA use clustering to distribute TDMA schedules and both use similar frame structures, we only discuss BMA and extend our findings for BMA to PACT. In BMA, the network is partitioned into clusters. Every network starts with the cluster setup phase, where every node decides whether to become a clusterhead. At the end of this phase, clusters are formed and the network enters the steady state phase. By this time, every clusterhead already knows the number of members in its cluster. The steady state phase is a sequence of sessions or frames. A session in turn consists of a contention period, a data transmission period and an idle period, and each of these periods are slotted (Figure 8.8(b)). The so-called contention period allows the nodes to tell the clusterhead, in their assigned slot, their transmission schedules. Since the clusterhead already knows the number of nodes in the cluster, n, the number of slots in the contention period is exactly n. The data transmission period and idle period have a variable number of slots, but each session has a fixed length so that when there are more data to send, the number of data transmission slots is increased while the number of idle slots is decreased accordingly. Since the control packets and the data packets occupy a separate slot of their own, Equation 8.4 can be applied to estimate $T_{\text{contention}}$ and T_{data}. In the contention period, interarrival times are shorter than those in the data transmission period. This is how the jammer can tell whether it is in the contention period or the data transmission period. PSJ and RPSJ can now be applied to jam the contention period and the data transmission period, each with a different estimated slot size.

Arisha partitions the network into clusters, and in each cluster there is a gateway that arbitrates medium access among sensors and sets routes for sensor data. Arisha divides time into phases, the most important of which are the data transfer phase and reroute phase (Figure 8.8(c)). A phase is in turn made of frames. A data transfer frame has a bigger slot size than a reroute frame, i.e. $T_{\text{data}} > T_{\text{reroute}}$. A similar approach to jamming BMA, as explained before, can be applied to jamming Arisha.

TRAMA divides time into alternating periods of random access and scheduled access.

Both periods are slotted. A slot in the random access period is called a signaling slot, while a slot in the scheduled access period is called a transmission slot. Denote the size of a signaling slot as $T_{\text{signaling}}$, and the size of a transmission slot as T_{Tx}, then for ease of synchronization, T_{Tx} is typically a multiple $T_{\text{signaling}}$, e.g. $T_{\text{Tx}} = 7T_{\text{signaling}}$ [237]. Again, a similar approach to jamming BMA can be applied to jamming TRAMA.

8.7.3 Random Access-Based Protocols

Random access-based protocols include low power listening [109], PCM [130], Sift [124] and WiseMAC [80].

Low power listening is for all our intents and purposes equivalent to B-MAC, so we will not discuss it any further. PCM is an improvement to IEEE 802.11 by exercising power control on a per-packet basis. Sift is a contention window-based MAC protocol. Since they do not specify any duty cycle, they offer no obvious exploits to PCJ, RPCJ, PSJ, RPSJ or LPLJ. But since they do not have any duty cycle, it remains to be seen how suitable they are, or how they can be adapted for WSNs.

WiseMAC uses the same preamble sampling scheme as B-MAC, with the difference being if the sender knows the schedule of the receiver, it waits until when the receiver is about to wake up and sends its packet with a normal, shorter preamble, instead of an LPL preamble. However for broadcasting packets, the sender often has to stretch the preamble to the full length of the LPL preamble [158]. Therefore given enough broadcast traffic, the jammer is still able to figure out the check interval and apply LPLJ.

8.7.4 Discussion

Summing up, all protocols from Langendoen et al.'s authoritative survey [158] that we have discussed have weaknesses due to their organization of medium accesses. Among these protocols, frame-based protocols have better resistance to energy-efficient jamming because they spread out transmissions in time. Fortunately, there are more frame-based protocols than any other type of protocols that have been proposed. In the next section, we explore some countermeasures.

8.8 Countermeasures

Since our attacks against S-MAC are based on clustering, a countermeasure would naturally be to prevent clustering-based analysis from being feasible. This can be done by narrowing the distance between Cluster1 and Cluster2. Assuming no packet is transmitted in the sleep interval, the biggest possible Cluster1 interarrival time is the length of the listen interval, whereas the smallest possible Cluster2 interarrival time is the length of the sleep interval. We can 'stick' Cluster1 and Cluster2 together by equating the biggest Cluster1 interarrival time to the smallest Cluster2 interarrival time, which is tantamount to setting the duty cycle to 50%. According to simulations with $N_j/N_s = 1$ and $D = 20$, this defense is modestly effective against PCJ if K-means is used as the clustering algorithm. The resultant censorship rate is reduced to 38% (with standard error 17%). However if expectation maximization (EM) [74] is used as the clustering algorithm, the censorship rate can only be reduced to 75% (with standard error 4%). Relief can be found in the fact that a jammer using EM would considerably deteriorate its own lifetime advantage because EM is a computationally expensive, and hence energy-consuming algorithm that involves multiple exponentiations. On the other hand, a duty cycle of 50% may not be energy-efficient enough for most WSNs

that are characterized by low data rate [226]. All in all, using a high duty cycle is a partial countermeasure to energy-efficient link-layer jamming.

In the case of LMAC, we mentioned that it is advantageous to spread transmissions out in time, but as long as we transmit at fixed slot sizes, spikes would manifest on the probability distribution graph of the packet interarrival times. The strategy of flattening the spikes to increase the difficulty in estimating the slot size can be served by changing the slot size pseudorandomly as a function of time and a hidden seed. However due to the adaptibility of the jamming algorithm, this countermeasure is less than satisfactory. For example, if the sensor nodes change their slot size every second, by pseudorandomly picking a value from the range [20 ms, 30 ms], then according to simulations, a jammer using PSJ can still achieve a censorship rate of 80% (with standard error 12%) and a lifetime advantage of 47% (with standard error 5%) when $N_j/N_s = 1$ and $D = 20$. However we may take comfort in knowing that against LMAC, the most effective energy-efficient attack, PSJ, has a lifetime advantage that is considerably lower than that of the most effective energy-efficient attack against S-MAC, PCJ.

For B-MAC, the situation cannot be helped, because B-MAC relies on the preamble being long enough for the receivers to detect. Shortening the preamble any further than what we have simulated, 10 ms, which is the minimum considered by Polastre et al. [226], defeats the purpose of B-MAC.

From the above discussion, it appears for now that an effective countermeasure is lacking, but by comparing the censorship rate and lifetime advantage of the best attack on the respective protocols, LMAC emerges as a better choice than S-MAC and B-MAC in terms of resistance against link-layer jamming. Generalizing this observation, TDMA protocols are potentially a better choice than other types of protocols.

8.9 Related Work

Wood et al. wrote in 2002 that no effective defense was yet known against link-layer jamming [300]. Both Ståhlberg [273] and Wood et al. [300] quote how an attacker might keep sending RTS packets to elicit CTS packets from its victims.

Negi and Perrig [200] investigate the attack of a jammer that detects and jams RTS packets, as well as sends RTS packets to reserve the largest time interval possible, using the Poisson arrival model. This type of jammer needs to be an insider of the network, in order to know the content of the packets, and also send RTS packets that can pass the integrity check by the normal sensor nodes. Our attackers are not bound by such an assumption. Moreover, if the RTS packet is short enough, the jammer may not have enough time to respond [303]. Jamming the *ensuing* CTS or data packets might be a more successful attack [4].

Konorski [152] proposes a scheduling policy that addresses the selfish behavior of using small or no contention time (also called random backoff time) in contention-based protocols, in the context of *single-hop* networks. We are only interested in multi-hop networks. Kyasanur et al. [156] propose that instead of letting the sender set the contention time, the receiver sets and sends the time in the CTS and ACK packets to the sender. The sender uses this assigned contention time in the subsequent transmission to the receiver. The receiver can then tell if the sender is being selfish, i.e. using a value smaller than the assigned value, by observing the number of idle slots between consecutive transmissions from the sender. The problem with this approach is that if the receiver misbehaves, the sender is penalized. Cárdenas et al. [49] propose using Blum's coin flipping protocol [34] to ensure that the sender

and the receiver choose a random backoff time, such that if *either* the sender or the receiver deviates from the random backoff, the other party would know. Čagalj et al. [290] investigate the effect of a group of selfish cheaters from a game-theoretic viewpoint. We focus on DoS attacks, in which the purpose of the misbehaving party lies in disruption instead of getting more bandwidth. Different countermeasures are thus required.

Xu et al. [304] propose two evasion strategies against constant jammers: (1) channel surfing and (2) spatial retreat. Channel surfing is essentially an adaptive form of frequency hopping. Instead of continuously hopping from frequency to frequency, a node only switches to a different frequency when it discovers the current frequency is being jammed. Spatial retreat is an algorithm according to which two nodes move in Manhattan distances to escape from a jammed region. The algorithm is more suitable for wireless sensor and actor networks (WSANs) [11] than conventional WSNs. The algorithms are effective against constant jammers but we are motivated to look at jammers who are more intelligent than constant jammers.

We mentioned the 4 generic jammer models by Xu et al. [303] earlier. Based on the models, Xu et al. show that either received signal strength indication (RSSI) or carrier sensing time alone is not sufficient in detecting all the 4 types of jammers. Instead, attacks can be detected by measuring (1) both the packet delivery ratio and the signal strength, or (2) both the packet delivery ratio and the location. In our previous work [4], we look at potential attacks on one particular MAC protocol, which is S-MAC, and provide a countermeasure. Instead of looking at the packet delivery ratio of individual nodes, we look at the effect of distributed jammer motes on the WSN as a whole. In our opinion, apart from jamming effectiveness, a jammer cares about jamming efficiency. For this reason, while Xu et al. provide metrics for measuring the quality of service of individual nodes, we provide metrics for measuring the jamming effectiveness and efficiency of the jammers. Xu et al.'s and our work are complementary in the sense that Xu et al. target jammers at the physical layer, while we target jammers that aim to gain more edge by exploiting the data link layer. This work extends our previous work [4] to more protocols and in a more general setting (the latest attacks work even on encrypted traffic, while the earlier attacks do not).

8.10 Conclusion

In earlier work we investigate link-layer jamming based on *detailed* knowledge of the MAC protocols. In the present paper we propose new link-layer jamming algorithms that are based on *minimal* knowledge of the target protocols. The effectiveness and energy-efficiency of the minimal knowledge attacks is almost as good as the effectiveness and energy-efficiency of the detailed knowledge attacks. In addition, the minimal knowledge attacks are effective when data packets are encrypted. We propose new metrics (censorship rate, attrition rate, effort ratio and lifetime advantage) and a method to quantify the effect of link-layer jamming. With typical WSN systems in use today no effective measures against link-layer jamming are possible. For WSNs that require high security against link-layer jamming we recommend (1) encrypting link-layer packets to ensure a high entry barrier for jammers, (2) the use of spread spectrum hardware, and (3) the use of a TDMA protocol. It is our future work to establish analytically the desirable characteristics of MAC protocols that are secure against link-layer jamming attacks.

Acknowledgements

This work is partially sponsored by the Smart Surroundings project. The authors would like to thank Jerry den Hartog, Gertjan Halkes, Koen Langendoen, Shea Ming Oon, Haohui Liao, Tjerk Hofmeijer, Stefan Dulman, and Ferdy Hanssen for the help they have provided, and the anonymous reviewers for their inspiring comments.

Appendix

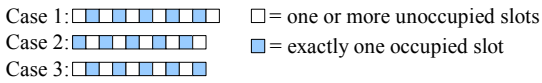


Figure 8.9: Three cases of slot distribution with no consecutive occupied slots.

Proof of Equation 8.1: Given s ($s \geq 4$) total number of slots and n ($0 \leq n \leq s$) occupied slots in a frame, we want to find the probability that at least two occupied slots are consecutive (denoted as event A), which is equivalent to 1 minus the probability that there is at least one unoccupied slot between any two occupied slots (denoted as event B). When $0 \leq n < 2$, $\Pr\{A\} = 0$, since there is either no occupied slot, or there is always one unoccupied slot between two occupied slots. When $\frac{s}{2} < n \leq s$, $\Pr\{A\} = 1$, since more than half of the slots are occupied and at least two of them are consecutive. For the other values of n, we refer to Figure 8.9 where one white box represents one or more unoccupied slots, and one shaded box represents exactly one occupied slot. Denote x_i ($x_i \geq 1$) as the number of unoccupied slots in the i-th white box. There are 3 cases to consider:

Case 1: $x_1 + \ldots + x_{n+1} = s - n$, $s \geq 2n + 1$, and the number of positive integer solutions for $x_1, \ldots, x_{n+1}$ is $\binom{s-n-1}{n}$ [288].

Case 2: $x_1 + \ldots + x_n = s - n$, $s \geq 2n$, and the number of solutions for $x_1, \ldots, x_n$ is $\binom{s-n-1}{n-1}$.

Case 3: Similar to Case 2, the number of solutions for $x_1, \ldots, x_n$ is $\binom{s-n-1}{n-1}$.

Therefore, when $2 \leq n \leq \frac{s-1}{2}$,

$$\Pr\{A\} = 1 - \Pr\{B\} = 1 - \frac{\binom{s-n-1}{n} + 2\binom{s-n-1}{n-1}}{\binom{s}{n}} = 1 - \frac{\binom{s-n}{n}}{\binom{s-1}{n}}$$

And when $\frac{s-1}{2} < n \leq \frac{s}{2}$, i.e. when $n = \frac{s}{2}$ and s is even,

$$\Pr\{A\} = 1 - \Pr\{B\} = 1 - \frac{2\binom{s-n-1}{n-1}}{\binom{s}{n}}$$

□

CHAPTER IX

Conclusions and Future Work

WSN technology is developing at full steam at the time of writing, but the corresponding progress in security research is lagging behind. This is partly due to the prevalent lack of common understanding about the nature of operation of WSNs, and partly due to the habitual lack of commercial motivation in the business circle, that cannot be summed up more clearly than in the saying: "don't fix it if it isn't broken". As the previous chapters have shown, the security problems with WSNs are open-ended. It is no use designing a protocol only to discover it is vulnerable to all sorts of security attacks, and scrambling to patch it. As security experts have all these while urged [214], security needs to be integrated into the system. Lessons should be learnt from the naïve assumption of the Internet protcols that all network nodes are trusted, and now that we are given the opportunity to start from a clean sheet of paper, full-hearted effort should be made in designing WSNs with *integrated security*. IEEE 802.15.4/ZigBee^{TM}Security for example is a step in the right direction.

To recapitulate, the contributions of this thesis are:

1. **Design framework**: We propose the concept of *system profile* and *critical system parameters*, allowing a system to be profiled, and a security architecture to be applied to it according to its profile.

2. **Evaluation of block ciphers**: We present a *detailed* benchmark for a selection of block ciphers, one of the most important cryptographic primitives for WSNs. The result should serve as a practical guideline on the suitable block cipher to use according to the constraint of available memory and the required level of security. This is the first systematic evaluation of block ciphers for WSNs in the literature and the first that focuses explicitly on energy.

3. **Key management**: We present ESA1, a lightweight, cluster-based, decentralized key management architecture for WSNs. The novelty of this work lies in the introduction of the security realms and the fact that the protocol suite is the first in WSNs to be formally verified.

4. **Secure multicast**: We present LKHW, a secure multicast scheme that is optimized for directed diffusion, and designed to facilitate secure data aggregation, an important operation primitive in WSNs. This is the first work that aims to harvest the synergy between directed diffusion and LKH.

5. **Link-layer jamming with detailed knowledge**: In the case where the packets are not encrypted, we present three new energy-efficient jamming attacks against S-MAC, thus demonstrating the need to prepare a countermeasure to link-layer jamming attacks in general. We provide a countermeasure to one of the attacks.

6. **Link-layer jamming with minimal knowledge**: In the case where the packets are encrypted, we present several jamming attacks against three representative protocols:

S-MAC, LMAC and B-MAC. We reason that the attacks are extensible to other protocols belong to the respective categories of S-MAC, LMAC and B-MAC. With typical WSN systems in use today no effective measures against link-layer jamming are possible. For WSNs that require high security against link-layer jamming we recommend (1) encrypting link-layer packets to ensure a high entry barrier for jammers, (2) the use of spread spectrum hardware, and (3) the use of a TDMA protocol.

Based on these contributions, we now review what we have accomplished in broader terms. To start with, we have systematized the design and development of WSN security. Our system profile-based assessment framework helps security designers of WSNs decide what to focus on in their design projects. In fact, our framework points them to only four aspects, greatly reducing the complexity of their design work. As soon as integrated security architectures start appearing, we will see system profiles doing their work – different architectures will be assigned to different system profiles, and hence different applications in a systematic manner.

Our results about the suitable block ciphers to use is the sole reference in the literature that helps developers determine which cipher to implement on their WSNs.

Our key management scheme ESA1 is applicable to the system profile that best describes the class of networks in which all nodes are not tamper-resistant and cannot use public-key cryptography, but in which some nodes are more resource-rich than the others. As this system profile encompasses a lot of real-life WSNs, ESA1 is widely applicable. The ease of implementation, the low resource requirement of ESA1, combined with the level of security it provides, should be appealing to any WSN developer that is concerned by the complexity and resource requirement of many current schemes. In order words, ESA1 lowers the entry barrier to secure WSNs. Our secure multicast scheme LKHW is based on direct diffusion, open source implementations of which are already available on the the website of TinyOS (the premiere operating system for WSNs). The practical values of LKHW are hence immense.

Our study in link-layer jamming is the first of its kind. For the first time, it reveals in a formal and systematic manner how the radio communication of WSNs can be energy-efficiently jammed on the data link layer. It should open avenues for research into new ways of protecting WSNs from energy-efficient jamming.

As future work, we propose extending our current work in key management methodologies to deal with more dynamic topologies. An integrated security framework that caters for both point-to-point and group communications is also a highly anticipated feature. Our current work in link-layer anti-jamming security could be extended by devising a comprehensive set of guidelines for designing secure MAC protocols. Aside from the areas the thesis discourses on, we propose more investigation on the provable security of WSN routing protocols as suggested by Buttyán et al. [47], and application-level security as suggested by Wagner [291].

Bibliography

Publications

[1] Y.W. Law, R. Corin, S. Etalle, and P.H. Hartel. A formally verified decentralized key management architecture for wireless sensor networks. In M. Conti, S. Giordano, E. Gregori, and S. Olariu, editors, *4th IFIP TC6/WG6.8 International Conference on Personal Wireless Communications (PWC 2003)*, volume 2775 of *LNCS*, pages 27–39. Springer-Verlag, September 2003. ISBN 3-540-20123-8. URL http://www.springerlink.com/index/TP55MU4LTW9WKVY7.pdf.

[2] Y.W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2006. To appear.

[3] Y.W. Law, S. Etalle, and P.H. Hartel. Assessing security in energy-efficient sensor networks. In D. Gritzalis, S. De Capitani di Vimercati, P. Samarati, and S.K. Katsikas, editors, *18th IFIP TC11 International Conference on Information Security, Security and Privacy in the Age of Uncertainty (SEC 2003)*, pages 459–463. Kluwer Academic Publishers, May 2003. ISBN 1-4020-7449-2. URL http://www.eyes.eu.org/publications/law03assessing.pdf.

[4] Y.W. Law, P. Hartel, J. den Hartog, and P. Havinga. Link-layer jamming attacks on S-MAC. In *2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 217–225. IEEE, 2005. ISBN 0-7803-8801-1. URL http://ieeexplore.ieee.org/iel5/9875/31391/01462013.pdf.

[5] Y.W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy-Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols. In *The Third ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005)*. ACM Press, 2005. To appear.

[6] R. Di Pietro, L.V. Mancini, Y.W. Law, S. Etalle, and P. Havinga. LKHW: A directed diffusion-based secure multicast scheme for wireless sensor networks. In C.-H. Huang and J. Ramanujam, editors, *32rd International Conference on Parallel Processing Workshops (ICPPW '03)*, pages 397–406. IEEE Computer Society Press, October 2003. ISBN 0-7695-2018-9. URL http://ieeexplore.ieee.org/iel5/8781/27812/01240395.pdf. Full paper at http://www.ub.utwente.nl/webdocs/ctit/1/000000cc.pdf.

References

[7] 3GPP. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2: KASUMI Specification. ETSI/SAGE Specification Version: 1.0, Dec 1999. URL http://downloads.securityfocus.com/library/3GTS35.202.pdf.

[8] M. Abdalla and M. Bellare. Increasing the lifetime of a key: A comparitive analysis of the security of rekeying techniques. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 546–565. Springer-Verlag, 2000. URL http://www.cs.ucsd.edu/users/mihir/papers/rekey.html.

[9] David L. Adamy and David Adamy. *EW 102: A Second Course in Electronic Warfare*. Artech House Publishers, 2004. ISBN 1580536867.

[10] Jonathan R. Agrea, Loren P. Clarea, Gregory J. Pottieb, and Nikolai P. Romanova. Development platform for self-organizing wireless sensor networks. In *Proc. SPIE, Unattended Ground Sensor Technologies and Applications*, volume 3713, pages 258–268, 1999.

[11] I.F. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Elsevier Ad Hoc Networks*, 2(4): 351–367, October 2004.

[12] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38 (4):393–422, 2002. ISSN 1389-1286.

[13] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., 2001. ISBN 0-471-38922-6.

[14] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *The 2nd USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11, 1996. ISBN 1-880446-83-9. URL http://www.cl.cam.ac.uk/~mgk25/tamper.html.

[15] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. http://www.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf, 1998.

[16] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms. In D.R. Stinson and S.E. Tavares, editors, *Proc. Selected Areas in Cryptography (SAC'00)*, number 2012 in LNCS, pages 39–56. Springer-Verlag, 2001. URL http://info.isl.ntt.co.jp/camellia.

[17] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Specification of Camellia – A 128-Bit Block Cipher. Specification Version 2.0, Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation, 2001.

[18] K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA based MAC for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications Computing and Networking (IMPACCT 2002)*, May 2002.

Bibliography

[19] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In *Proceedings of the 1998 European Solid State Circuits Conference*, 1998. URL `http://www.janet.ucla.edu/awairs/download_pubs/esscirc98.pdf`.

[20] Associated Press. Japan's 'smart' transport systems taking the slow road, December 2004. URL `http://www.boston.com/business/technology/articles/2004/12/27/japans_smart_transport_systems_taking_the_slow_road/`.

[21] G. Ateniese, M. Steiner, and G. Tsudik. New multiparty authentication services and key agreement protocols. *IEEE Journal on Selected Areas in Communications*, 18(4), 2000. URL `http://citeseer.nj.nec.com/article/ateniese00new.html`.

[22] S. Babbage and L. Frisch. On MISTY1 Higher Order Differential Cryptanalysis. In *3rd International Conference on Information Security and Cryptology, ICISC 2000*, volume 2015 of *LNCS*, pages 22–36. Springer-Verlag, 2001.

[23] E. Barkan and E. Biham. In How Many Ways Can You Write Rijndael. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on Theory and Application of Cryptology and Information Security*, volume 2501 of *LNCS*, pages 160–175. Springer-Verlag, 2002.

[24] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure pebblenets. In *Proc. 2001 ACM Int. Symp. on Mobile Ad Hoc Networking and Computing*, pages 156–163. ACM Press, October 2001. ISBN 1-58113-428-2.

[25] M. Bellare and B. Yee. Forward-security in private-key cryptography. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, volume 2612 of *LNCS*, pages 1–18. Springer-Verlag, 2003.

[26] Celeste Biever. Virtual fences to herd Wi-Fi cattle. NewScientist.com, June 2004. URL `http://www.newscientist.com/news/news.jsp?id=ns99995079`.

[27] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Advances in Cryptology - EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *LNCS*, pages 12–23. Springer-Verlag, 1999.

[28] Eli Biham and Vladimir Furman. Improved Impossible Differentials on Twofish. In *Progress in Cryptology - INDOCRYPT 2000: First International Conference in Cryptology in India*, volume 1977 of *LNCS*, pages 80–92. Springer-Verlag, 2000.

[29] A. Biryukov. *Methods of Cryptanalysis*. PhD thesis, Technion, 1999.

[30] A. Biryukov and E. Kushilevitz. Improved Cryptanalysis of RC5. In *Advances in Cryptology – EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *LNCS*, pages 85–99. Springer-Verlag, 1998.

[31] A. Biryukov and D. Wagner. Advanced slide attacks. In *Advances in Cryptology - EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 589–606. Springer-Verlag, 2000.

[32] L. Blažević, L. Buttyán, S. Čapkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self-organization in mobile ad hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 39(6):164–174, June 2001.

[33] G. E. Blomgren. Current status of lithium ion and lithium polymer secondary batteries. In *IEEE Fifteenth Annual Battery Conference on Applications and Advances*, pages 97–100, 2000.

[34] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, 1983. ISSN 0163-5700.

[35] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 146(1):1–23, 1995. URL `http://citeseer.nj.nec.com/blundo95perfectlysecure.html`.

[36] Dan Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory (ANTS-III)*, volume 1423 of *LNCS*, pages 48–63. Springer-Verlag, 1998.

[37] Dan Boneh and Ramarathnam Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes (Extended Abstract). In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *LNCS*, pages 129–142. Springer-Verlag, 1996.

[38] Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, and Michael Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. In *Proceedings of the 14th USENIX Security Symposium*, pages 1–16. USENIX Association, 2005. URL `http://www.rfidanalysis.org/DSTbreak.pdf`.

[39] G. Boone. Reality mining: Browsing reality with sensor networks. *Sensors*, 21(9), September 2004. URL `http://sensorsmag.com/articles/0904/14/main.shtml`.

[40] G. Borriello and R. Want. Embedded computation meets the world wide web. *Communications of the ACM*, 43(5):59–66, 2000.

[41] J. Borst, B. Preneel, and J. Vandewalle. Linear Cryptanalysis of RC5 and RC6. In L.R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99*, volume 1636 of *LNCS*, pages 16–30. Springer-Verlag, March 1999. URL `http://www.esat.kuleuven.ac.be/~borst/RC6.html`.

[42] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31. ACM Press, 2002. ISBN 1-58113-589-0.

[43] M. Brown, D. Cheung, D. Hankerson, J. L. Hernandez, M. Kirkup, and A. Menezes. PGP in Constrained Wireless Devices. In *9th USENIX Security Symposium*, pages 247–261. USENIX Association, August 2000. URL `http://www.usenix.org/events/sec2000/brown.html`.

[44] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A.D. Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *LNCS*, pages 275–286. Springer-Verlag, 1995.

[45] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, David Safford, and Nevenko Zunic. MARS - a candidate cipher for AES. `http://researchweb.watson.ibm.com/security/mars.pdf`, September 1999.

[46] L. Buttyán and J.-P. Hubaux. Nuglets: A Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical Report DSC/2001/001, Department of Communication Systems, Swiss Federal Institute of Technology, 2001. URL `http://ntrg.cs.tcd.ie/htewari/papers/tr01_001.pdf`.

[47] Levente Buttyán and István Vajda. Towards provable security for ad hoc routing protocols. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 94–105, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-972-1.

[48] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and efficient constructions. In *INFOCOM '99*, volume 2, pages 708–716, March 1999. URL `http://www.bell-labs.com/user/garay/multicast.ps`.

[49] Alvaro A. Cárdenas, Svetlana Radosavac, and John S. Baras. Detection and prevention of MAC layer misbehavior in ad hoc networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 17–22, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-972-1.

[50] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and approaches for distributed sensor network security. Technical Report #00-010, NAI Labs, 2000. URL `http://download.nai.com/products/media/nai/zip/nailabs-report-00-010-final.zip`.

[51] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[52] Wayne Catlin, Lee Eccles, and Larry Malchodi. Smart sensor project takes flight – boeing 'pressure belt' to measure airplane wing stress. *InTech*, May 2002. URL `http://www.isa.org/Content/ContentGroups/InTech2/Features/20023/May6/20020531.pdf`.

[53] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2003.

[54] S. Chatterjea, L. van Hoesel, and P. Havinga. AI-LMAC: An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks. In *Proceedings of the DEST International Workshop on Signal Processing for Sensor Networks*. IEEE Computer Society Press, 2004. ISBN 0-7803-8894-1.

[55] J.H. Cheon, M.J. Kim, K. Kim, and S. W. Kang J.-Y. Lee. Improved Impossible Differential Cryptanalysis of Rijndael and Crypton. In K. Kim, editor, *4th International Conference on Information Security and Cryptology, ICISC 2001*, volume 2288 of *LNCS*, pages 39–49. Springer-Verlag, 2002.

[56] P.C. Chien and V. Wen. CS199 – StrongARM Energy Measurement Report. Online slides: `http://www.cs.berkeley.edu/~vwen/strongarm/slides/cs199.ppt`, 1998.

[57] *SmartRF® CC1000 Datasheet (rev. 2.2)*. Chipcon AS, April 2004. URL `http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf`.

[58] *SmartRF® CC2420 Datasheet (rev. 1.2)*. Chipcon AS, June 2004. URL `http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf`.

[59] David Cohn. New outlets for viruses, spam. Wired News Report, February 2005. URL `http://www.wired.com/news/business/0,1367,66549,00.html`.

[60] D. Cook and S. Das, editors. *Smart Environments: Technology, Protocols and Applications*. John Wiley and Sons, Ltd., December 2004. ISBN 0-471-54448-5.

[61] D. Coppersmith. Re: Impact of Courtois and Pieprzyk results. Forum message at `http://aes.nist.gov/aes/`, September 2002.

[62] R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In M. Hermenegildo and G. Puebla, editors, *9th Int. Static Analysis Symp. (SAS)*, volume 2477, pages 326–341, Madrid, Spain, Sep 2002. Springer-Verlag. ISBN 3-540-44235-9.

[63] N. Courtois, L. Goubin, W. Meier, and J.-D. Tacier. Solving underdefined systems of multivariate quadratic equations. In *PKC 2002*, volume 2274 of *LNCS*, pages 211–227. Springer-Verlag, 2002.

[64] N.T. Courtois, R.T. Johnson, P. Junod, T. Pornin, and M. Scott. Did Filiol Break AES? Cryptology ePrint Archive: Report 2003/022, February 2003. URL `http://eprint.iacr.org/2003/022/`.

[65] N.T. Courtois and J. Patarin. About the XL Algorithm over GF(2). In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, volume 2612 of *LNCS*, pages 141–157. Springer-Verlag, 2003.

[66] N.T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on Theory and Application of Cryptology and Information Security*, volume 2501 of *LNCS*, pages 267–287. Springer-Verlag, 2002.

[67] N.T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive: Report 2002/044, November 2002. URL http://eprint.iacr.org/2002/044/.

[68] S.A. Crosby and D.S. Wallach. Denial of service via algorithmic complexity attacks. In *12th USENIX Security Symposium*, pages 29–44. USENIX Association, 2003. URL http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003/index.html.

[69] *MPR/MIB User's Manual, Rev. A, August 2004, Document 7430-0021-06*. Crossbow Technology, Inc., August 2004.

[70] CRYPTREC. Analysis of RC6. 暗号アルゴリズム及び関連技術の評価報告 (trans.: Evaluation report of cryptographic algorithms and related technologies) no. 1086, January 2001. URL http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/cryptrec20030424_outrep01.html.

[71] CRYPTREC. 電子政府推奨暗語の仕様書 (trans.: Specification of e-government-recommended ciphers). http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/cryptrec20030425_spec01.html, December 2003.

[72] J. Daemen, L. Knudsen, and V. Rijmen. The Block Cipher SQUARE. In E. Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer-Verlag, January 1997.

[73] J. Daemen and V. Rijmen. AES Proposal: Rijndael, 1999. URL http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip.

[74] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[75] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Bassard, editor, *Advances in Cryptology – Crypto '89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, August 1989.

[76] Terry Devitt. Harnessing microbes, one by one, to build a better nanoworld. Press release, March 2005. URL http://www.news.wisc.edu/releases/print.php?id=10831.

[77] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[78] O. Dunkelman. Comparing MISTY1 and KASUMI. NESSIE Public Report NES/DOC/TEC/WP5/029/a, Computer Science Department, Technion, December 2002. URL https://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase2/misty-kasumi.pdf.

[79] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *IEEE International Conference on Communications*. IEEE Press, April 2000.

[80] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux. Poster abstract: WiseMAC, an ultra low power MAC protocol for the WiseNET wireless sensor network. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 302–303, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-707-9.

[81] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. 9th ACM conference on Computer and communications security*, pages 41–47. ACM Press, 2002. ISBN 1-58113-612-9.

[82] M. Esler, J. Hightower, T. Anderson, and G. Borriello. Next century challenges: data-centric networking for invisible computing: the portolano project at the university of washington. In *5th Annual ACM/IEEE International Conference on Mobile computing and networking*, pages 256–262. ACM Press, 1999. ISBN 1-58113-142-9. URL http://doi.acm.org/10.1145/313451.313553.

[83] B.J. Feder. Wireless Sensor Networks Spread to New Territory. The New York Times, July 2004. URL http://www.nytimes.com/2004/07/26/business/26sensor.html.

[84] L. Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T99/07, Swedish Institute of Computer Science, October 1999.

[85] L. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous networking: An application-oriented approach to ad hoc networking. *IEEE Communications*, 39(6):176–181, June 2001.

[86] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved Cryptanalysis of Rijndael. In B. Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000*, volume 1978 of *LNCS*, pages 213–230. Springer-Verlag, 2001.

[87] N. Ferguson, R. Schroeppel, and D. Whiting:. A Simple Algebraic Representation of Rijndael. In *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001*, volume 2259 of *LNCS*, pages 103–111. Springer-Verlag, 2001.

[88] E. Filiol. Plaintext-dependant Repetition Codes Cryptanalysis of Block Ciphers - The AES Case. Cryptology ePrint Archive: Report 2003/003, January 2003. URL http://eprint.iacr.org/2003/003/.

[89] Christy Fujio. Millennial Net and Ferro Solutions Join Forces to Deliver Battery-Free Wireless Sensor Networks. Press release, October 2003. URL http://www.ferrosi.com/pr1.htm.

[90] J. Fuller and W. Millan. On Linear Redundancy in the AES S-Box. Cryptology ePrint Archive: Report 2002/111, August 2002.

[91] Andrea Gabrielli, Luigi Mancini, Sanjeev Setia, and Sushil Jajodia. Securing topology maintenance protocols for sensor networks: Attacks & countermeasures. In *Proc. of First IEEE/Createnet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005)*, September 2005.

[92] D. Ganesan, A. Cerpa, Y. Yu, and D. Estrin. Networking issues in wireless sensor networks. *Journal of Parallel and Distributed Computing (JPDC), Special issue on Frontiers in Distributed Sensor Networks*, To appear. URL `http://lecs.cs.ucla.edu/~deepak/PAPERS/jpdc.pdf`.

[93] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (MC2R)*, 1(2), 2002.

[94] H. Gilbert, H. Handschuh, A. Joux, and S. Vaudenay. A Statistical Attack on RC6. In *Fast Software Encryption, 7th International Workshop, FSE 2000*, volume 1978 of *LNCS*, pages 64–74. Springer-Verlag, 2000.

[95] H. Gilbert and M. Minier. A collision attack on 7 rounds of Rijndael. In *Proc. 3rd AES Conference (AES3)*, April 2000. URL `http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/papers/11-hgilbert.pdf`.

[96] Steven D. Glaser. Some real-world applications of wireless sensor nodes/ nde 2004. In *SPIE Symposium on Smart Structures & Materials*, March 2004. URL `http://firebug.sourceforge.net/glaser_spie_2004_paper.pdf`.

[97] N. Goh. Wireless Sensor Networks – Ubiquitous Watchdogs of the Future? INTRO Newsletter, August 2003. URL `http://www.nus.edu.sg/intro/newsletter0311.shtml`.

[98] Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and correcting malicious data in VANETs. In *Proceedings of the first ACM workshop on Vehicular ad hoc networks*, pages 29–37. ACM Press, 2004. ISBN 1-58113-922-5.

[99] R. Govindan, T. Faber, J. Heidemann, and D. Estrin. Ad-hoc smart environments. In *Proceedings of the DARPA/NIST Workshop on Smart Environments*, June 1999. URL `http://www.isi.edu/scadds/papers/smartenv.ps.gz`.

[100] Charles Graeber. The lock busters. *Wired Magazine*, 13.02, February 2005. URL `http://www.wired.com/wired/archive/13.02/lockbusters.html`.

[101] Mark G. Graff and Kenneth R. Van Wyk. *Secure Coding: Principles and Practices*. O'Reilly, 2003. ISBN 0596002424.

[102] G. Hachez, F. Koeune, and J.-J. Quisquater. cAESar results: Implementation of Four AES Candidates on Two Smart Cards. In *2nd AES Candidate Conference (AES2)*, October 1999. URL `http://csrc.nist.gov/encryption/aes/round1/conf2/papers/hachez.pdf`.

[103] H. Handschuh and H. Heys. A Timing Attack on RC5. In S.E. Tavares and H. Meijer, editors, *Selected Areas in Cryptography '98, SAC'98*, volume 1556 of *LNCS*, pages 306–318. Springer-Verlag, 1998.

[104] H. Handschuh and D. Naccache. SHACAL. In *Proc. First Open NESSIE Workshop*, 2000. URL `https://www.cosic.esat.kuleuven.ac.be/nessie/tweaks.html`.

[105] Vlado Handziski, Joseph Polastre, Jan-Hinrich Hauer, and Cory Sharp. Flexible hardware abstraction of the ti msp430 microcontroller in tinyos. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 277–278. ACM Press, 2004. ISBN 1-58113-879-2.

[106] Y. Hatano, H. Sekine, and T. Kaneko. Higher Order Differential Attack of *Camellia*(II). In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography. 9th Annual Int. Workshop, SAC 2002*, volume 2595 of *LNCS*, pages 129–146. Springer-Verlag, 2002.

[107] Y. He and S. Qing. Square Attack on Reduced Camellia Cipher. In S. Qing, T. Okamoto, and J. Zhou, editors, *Information and Communications Security: Third International Conference, ICICS 2001*, volume 2229 of *LNCS*, pages 238–245. Springer-Verlag, 2001.

[108] J.S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, pages 146–159, 2001. URL `citeseer.nj.nec.com/heidemann01building.html`.

[109] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November 2002.

[110] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K.S.J. Pister. System architecture directions for networked sensors. *SIGOPS Oper. Syst. Rev.*, 34(5):93–104, 2000. ISSN 0163-5980.

[111] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen, and P.J.M. Havinga. AmbientRT - real time system software support for data centric sensor networks. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 61–66. IEEE, 2004.

[112] Y.-C. Hu, A. Perrig, and D.B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *Proc. 8th Annual Int. Conf. on Mobile Computing and Networking*, pages 12–23. ACM Press, 2002. ISBN 1-58113-486-X.

[113] Q. Huang, C. Lu, and G.-C. Roman. Spatiotemporal multicast in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 205–217. ACM Press, 2003. ISBN 1-58113-707-9.

[114] J.-P. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proc. of the ACM Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC '01)*, pages 146–155. ACM Press, October 2001. ISBN 1-58113-428-2.

[115] IEEE. IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and

Physical Layer (PHY) Specifications, November 1999. URL http://standards.ieee.org/getieee802/802.11.html.

[116] IEEE. IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), October 2003. URL http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

[117] *Wireless Components: ASK/FSK 868MHz Wireless Transceiver TDA 5250 D2 Version 1.6*. Infineon Technologies AG, 07.02 edition, July 2002. URL http://www.infineon.com/cmc_upload/documents/046/676/TDA5250_V1.6.pdf.

[118] I. Ingemarsson, D.T. Tang, and C.K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5):714–720, 1982. URL http://www.cbcis.wustl.edu/~adpol/courses/cs502/Notes/conf.pdf.

[119] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *6th Annual Int. Conf. on Mobile Computing and Networking (MobiCOM '00)*, pages 56–67, Boston, Massachusetts, United States, 2000. ACM Press. ISBN 1-58113-197-6.

[120] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003. ISSN 1063-6692.

[121] Intel Corporation. Intel Research – Digital Home Technologies for Aging in Place. http://www.intel.com/research/exploratory/digital_home.htm, .

[122] Intel Corporation. Intel Research – Exploratory Research – Deep Networking. http://www.intel.com/research/exploratory/heterogeneous.htm, .

[123] *Intel Architecture Software Developer's Manual Volume 2: Instruction Set Reference*. Intel Corporation, 1997.

[124] K. Jamieson, H. Balakrishnan, and Y.C. Tay. Sift: A MAC protocol for event-driven wireless sensor networks. Technical Report MIT-LCS-TR-894, Massachusetts Institute of Technology, May 2003.

[125] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[126] R.C. Johnson. Sandia enlists MEMS for anti-terror systems. *EE Times*, March 2002. URL http://www.eet.com/at/news/OEG20020514S0033.

[127] R.C. Johnson. Silicon-based gunpowder may propel MEMS devices. *EE Times*, January 2002. URL http://www.eetimes.com/semi/news/OEG20020122S0070.

[128] S. Johnson. Stopping loose nukes. *Wired Magazine*, 10.11, November 2002.

[129] Jakob Jonsson. On the Security of CTR + CBC-MAC. In *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 76–93. Springer-Verlag, 2003. ISBN 3-540-00622-2.

[130] Eun-Sun Jung and Nitin H. Vaidya. A power control MAC protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 36–47, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-486-X.

[131] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *LNCS*, pages 36–49. Springer-Verlag, 1996.

[132] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Mobile networking for smart dust. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, United States*, pages 271–278. ACM Press, 1999.

[133] B.S. Kaliski and Y.L. Yin. On the Security of the RC5 Encryption Algorithm. Technical Report TR-602, RSA Laboratories, September 1998. URL ftp://ftp.rsasecurity.com/pub/rsalabs/rc5/rc5-report.pdf.

[134] J.-S. Kang, S.-U. Shin, D. Hong, and O. Yi. Provable Security of KASUMI and 3GPP Encryption Mode $f8$. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *LNCS*, pages 255–271. Springer-Verlag, 2001.

[135] J.-S. Kang, O. Yi, D. Hong, and H. Cho. Pseudorandomness of MISTY-Type Transformations and the Block Cipher KASUMI. In V. Varadharajan and Y. Mu, editors, *Proceedings of the 6th Australasian Conference on Information Security and Privacy, ACISP 2001*, volume 2119 of *LNCS*, pages 60–73. Springer-Verlag, 2001.

[136] Holger Karl and Andreas Willig. A short survey of wireless sensor networks. Technical Report TKN-03-018, Technical University Berlin, October 2003. URL http://www.tkn.tu-berlin.de/tkn/publications/papers/TechReport_03_018.pdf.

[137] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, 2003. URL http://www.cs.berkeley.edu/~daw/papers/.

[138] Chris Karlof, Naveen Sastry, and David 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 Press. ISBN 1-58113-879-2.

[139] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM Press, 2000. ISBN 1-58113-197-6.

[140] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure multicast groups on ad hoc networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 94–102, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-783-4.

[141] Geoffrey Keating. Performance Analysis of AES candidates on the 6805 CPU core. In *2nd AES Candidate Conference (AES2)*, February 1999. URL `http://csrc.nist.gov/CryptoToolkit/aes/round1/conf2/papers/keating.pdf`.

[142] J. Kelsey. Key Separation in Twofish. Technical Report #7, Counterpane Internet Security, Inc., April 2000.

[143] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side channel cryptanalysis of product ciphers. In *Computer Security (ESORICS'98)*, volume 1485 of *LNCS*, pages 97–110. Springer-Verlag, 1998. URL `http://www.counterpane.com/side-channel2.pdf`.

[144] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Network Working Group, November 1998.

[145] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search. In *Advances in Cryptology - CRYPTO '96: 16th Annual International Cryptology Conference*, number 1109 in LNCS. Springer-Verlag, 1996.

[146] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 235–244, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-203-4.

[147] R. Kling. Intel mote: An Enhanced Sensor Network Node. In *International Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures*, 2003. URL `http://www.mita.sd.keio.ac.jp/news/workshop/proceedings/Kling.pdf`.

[148] J. Kloeppel. Smart bricks could monitor buildings, save lives. News Bureau, University of Illinois at Urbana-Champaign, June 2003. URL `http://www.news.uiuc.edu/scitips/03/0612smartbricks.html`.

[149] Terry Knott. Smart surrogates. *Frontiers*, 9, April 2004. URL `http://www.bp.com/liveassets/bp_internet/globalbp/STAGING/global_assets/images/fr/downloads/Frontiers_magazine_issue_09_smart_surrogates.pdf`.

[150] L.R. Knudsen and W. Meier. Correlations in RC6 with a Reduced Number of Rounds. In *Fast Software Encryption, 7th International Workshop, FSE 2000*, volume 1978 of *LNCS*, pages 94–108. Springer-Verlag, 2000.

[151] L.R. Knudsen and D. Wagner. Integral cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer-Verlag, 2002.

[152] J. Konorski. Multiple Access in Ad-Hoc Wireless LANs with Noncooperative Stations. In *NETWORKING 2002: Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, volume 2345 of *LNCS*, pages 1141–1146. Springer-Verlag, 2002.

[153] U. Kühn. Cryptanalysis of Reduced-Round MISTY. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 325–339. Springer-Verlag, 2001.

[154] U. Kühn. Improved Cryptanalysis of MISTY1. In *Fast Software Encryption, 9th International Workshop, FSE 2002*, volume 2365 of *LNCS*, pages 61–75. Springer-Verlag, 2002.

[155] S. S. Kulkarni and U. Arumugam. TDMA Service for Sensor Networks. *Proceedings of the Third International Workshop on Assurance in Distributed Systems and Networks (ADSN)*, March 2004.

[156] P. Kyasanur and N. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. In *Int. Conf. on Dependable Systems and Networks (DSN'03)*, pages 173–182. IEEE Computer Society Press, 2003.

[157] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981. ISSN 0001-0782.

[158] Koen Langendoen and Gertjan Halkes. *Embedded Systems Handbook*, chapter Energy-Efficient Medium Access Control. CRC Press, 2005. URL `http://www.isa.ewi.tudelft.nl/~koen/papers/MAC-chapter.pdf`. To appear.

[159] Loukas Lazos and Radha Poovendran. SeRLoc: secure range-independent localization for wireless sensor networks. In *Proceedings of the 2004 ACM workshop on Wireless security*, pages 21–30. ACM Press, 2004. ISBN 1-58113-925-X.

[160] S. Lee, S. Hong, S. Lee, J. Lim, and S. Yoon. Truncated Differential Cryptanalysis of Camellia. In K. Kim, editor, *4th International Conference on Information Security and Cryptology, ICISC 2001*, volume 2288 of *LNCS*, pages 32–38. Springer-Verlag, 2002.

[161] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[162] Jing Li and Georgios Y. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 55–60, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-846-6.

[163] Tieyan Li, Hongjun Wu, Xinkai Wang, and Feng Bao. SenSec Design. Technical Report TR-I2R-v1.1, InfoComm Security Department,Institute for Infocomm Research, February 2005.

[164] G. Lin and G. Noubir. Low Power DOS Attacks in Data Wireless LANs and Countermeasures. Technical report, Northeastern University, 2002. URL `http://www.ccs.neu.edu/home/noubir/publications/LN02a.pdf`.

[165] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005. ISSN 1094-9224.

[166] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. *18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 12*, 13(13), 2004.

[167] Stefan Lucks. The Saturation Attack – A Bait for Twofish. In *Fast Software Encryption, 8th International Workshop, FSE 2001*, volume 2355 of *LNCS*, pages 1–15. Springer-Verlag, 2002.

[168] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *7th IEEE Symp. on Computers and Communications*, pages 567–574, 2002. URL `http://citeseer.nj.nec.com/507663.html`.

[169] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM Press, 2002. ISBN 1-58113-589-0.

[170] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004. URL `http://www.eecs.harvard.edu/~mdw/papers/codeblue-bsn04.pdf`.

[171] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. 6th Annual Int. Conf. on Mobile Computing and Networking*, pages 255–265. ACM Press, 2000. URL `http://citeseer.nj.nec.com/marti00mitigating.html`.

[172] M. Matsui. New Block Encryption Algorithm MISTY. In E. Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97*, volume 1267 of *LNCS*, pages 54–68. Springer-Verlag, 1997.

[173] M. Matsui and T. Tokita. MISTY, KASUMI and Camellia Cipher Algorithm. *Mitsubishi Electric ADVANCE (Cryptography Edition)*, 100:2–8, Dec 2000. URL `http://global.mitsubishielectric.com/pdf/advance/vol100/vol100_complete.pdf`.

[174] Mitsuru Matsui. Linear Cryptanalysis of DES. In *Advances in Cryptology - EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *LNCS*, pages 386–397. Springer-Verlag, 1993.

[175] William Maurer. *The Scientist and Engineer's Guide to TinyOS Programming*. The TinyOS Documentation Project, June 2003. URL `http://ttdp.org/guides.html`.

[176] MaxStream. XStream^{TM}OEM RF Module. Product Manaul v4.2A, September 2004. URL `http://www.maxstream.net/products/xstream/module/product-manual_XStream_OEM-RF-Module.pdf`.

[177] K. Mayer. Instrumenting cattle – real time health monitoring of cattle using wireless technologies. Poster for Sir Mark Oliphant Conference 2004 "Converging Technologies for Agriculture and Environment", August 2004. URL `http://mobile.act.cmis.csiro.au/kevin/smartsensors2004.pdf`.

[178] D.A. McGrew and A.T. Sherman. Key establishment in large dynamic groups using one-way function trees. Technical Report 0755, TIS Labs, Network Associates, Inc., 1998. URL `http://download.nai.com/products/media/nai/misc/oft052098.ps`.

[179] D.J.C. McKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. URL `http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html`.

[180] Megan McRainey. Mini generator has enough power to run electronics. Georgia Tech News Release, November 2004. URL `http://www.gatech.edu/news-room/release.php?id=490`.

[181] A.J. Menezes, S.A. Vanstone, and P.C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996. ISBN 0849385237.

[182] MeshNetworks. MeshNetworks Enabled Architecture (MEA^{TM}) Solutions for Law Enforcement Agencies. White paper. URL `http://www.meshnetworks.com/pdf/wp_mea_law_enforcement.pdf`.

[183] Jonathan Min. *Analysis and Design Of A Frequency-Hopped Spread-Spectrum Transceiver For Wireless Personal Communications*. PhD thesis, University of California, Los Angeles, 1995. URL `http://www.icsl.ucla.edu/aagroup/PDF_files/tcvr-arch.pdf`.

[184] R. Min and A. Chandrakasan. Mobicom poster: top five myths about the energy consumption of wireless communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(1):65–67, 2003.

[185] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Amit Sinha, Eugene Shih, Alice Wang, and Anantha Chandrakasan. An architecture for a power-aware distributed microsensor node. In *IEEE Workshop on Signal Processing Systems (SiPS '00)*. IEEE Computer Society Press, October 2000. URL `http://www-mtl.mit.edu/research/icsystems/uamps/pubs/files/rmin-sips00-paper.pdf`.

[186] Fauzan Mirza and Sean Murphy. An Observation on the Key Schedule of Twofish. In *Proc. 2nd AES Conference (AES2)*, 1999.

[187] Mitsubishi Electric Corp. `http://info.isl.ntt.co.jp/crypt/camellia/dl/camellia.c`, 2001.

[188] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 277–288. ACM Press, 1997.

ISBN 0-89791-905-X.

[189] A. Miyaji, M. Nonaka, and Y. Takii. Known plaintext correlation attack against RC5. In B. Preneel, editor, *Topics in Cryptology – CT-RSA 2002, The Cryptographers' Track at the RSA Conference 2002*, volume 2271 of *LNCS*, pages 131–148. Springer-Verlag, 2002.

[190] T. Moh. On the Courtois-Pieprzyk's Attack on Rijndael . Web page: `http://www.usdsi.com/aes.html`, September 2002.

[191] Moteiv Corporation. Telos (rev a). Datasheet, August 2004. URL `http://www.moteiv.com/products/docs/telos-reva-datasheet-r.pdf`.

[192] S. Murphy and M.J.B. Robshaw. Comments on the Security of the AES and the XSL Technique. `http://www.isg.rhul.ac.uk/~mrobshaw/rijndael/xslnote.pdf`, September 2002.

[193] S. Murphy and M.J.B. Robshaw. Essential Algebraic Structure within the AES. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *LNCS*, pages 1–16. Springer-Verlag, 2002.

[194] S. Murphy and M.J.B. Robshaw. Key-dependent s-boxes and differential cryptanalysis. *Des. Codes Cryptography*, 27(3): 229–255, 2002. ISSN 0925-1022.

[195] Sean Murphy. The Key Separation of Twofish. In *Proc. 3rd AES Conference (AES3)*, 2000.

[196] M. Mysore, M. Golan, E. Osterweil, D. Estrin, and M. Rahimi. TinyDiffusion in the Extensible Sensing System at the James Reserve. `http://www.cens.ucla.edu/~mmysore/Design/OPP`, May 2003.

[197] *LMX3162 Single-Chip Radio Transceiver*. National Semicondutor, March 2000. URL `http://cache.national.com/ds/LM/LMX3162.pdf`.

[198] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback. Report on the Development of the Advanced Encryption Standard (AES). Technical report, NIST, 2000. URL `http://csrc.nist.gov/encryption/aes/round2/r2report.pdf`.

[199] George C. Necula. Proof-carrying code (abstract): design, implementation and applications. In *PPDP '00: Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 175–177, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-265-4.

[200] R. Negi and A. Perrig. Jamming analysis of MAC protocols. Carnegie Mellon Technical Memo, 2003. URL `http://www.ece.cmu.edu/~negi/publications/preprints/csma_jam.ps`.

[201] *Portfolio of recommended cryptographic primitives.* NESSIE Consortium, February 2003.

[202] *Skipjack and KEA Algorithm Specifications Version 2.0*. NIST, May 1998. URL `http://csrc.nist.gov/CryptoToolkit/skipjack/`.

[203] G. Noubir and G. Lin. Low-power DoS attacks in data wireless LANs and countermeasures. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):29–30, 2003.

[204] Geuvara Noubir. On connectivity in ad hoc networks under jamming using directional antennas and mobility. In *Wired/Wireless Internet Communications*, volume 2957 of *LNCS*, pages 186–200. Springer-Verlag, 2004.

[205] N. Noury, P. Barralon, P. Couturier, F. Favre-Reguillon, R. Guillemaud, C. Mestais, Y. Caritu, D. David, S. Moine, A. Franco, F. Guiraud-By, M. Berenguer, and H. Provost. A MEMS based microsystem for the monitoring of the activity of frail elderly in their daily life: the ACTIDOM project. In *6th International Workshop on Enterprise Networking and Computing in Healthcare Industry (Healthcom 2004)*, pages 105–109. IEEE Computer Society Press, June 2004.

[206] K. Nyberg. Linear approximations of block ciphers. In *Advances in Cryptology – EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *LNCS*, pages 439–444. Springer-Verlag, 1995.

[207] H. Ohta and M. Matsui. A Description of the MISTY1 Encryption Algorithm. RFC 2994, Network Working Group, IETF, November 2000. URL `http://www.faqs.org/rfcs/rfc2994.html`.

[208] Oki Electric Industry Co., Ltd. Oki Electric Develops 2.4GHz LSI Compliant to IEEE802.15.4/ZigBee® – Forms an Alliance with CompXs in the Next Generation Low Rate Wireless Network IC Development –. Press release, May 2004. URL `http://www.oki.com/en/press/2004/z04017e.html`.

[209] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, August 2003. ISBN 0195128958.

[210] Nathan Ota. Design and analysis of wireless sensor and control networks with applications for smart buildings. Master's thesis, Berkeley Manufacturing Institute, University of California, Berkeley, 2003. URL `http://bwrc.eecs.berkeley.edu/Publications/2003/index.htm`.

[211] G. Pei and C. Chien. Low power TDMA in large wireless sensor networks. In *Military Communications Conference (MILCOM 2001)*, volume 1, pages 347—351. IEEE, 2001. URL `http://ieeexplore.ieee.org/iel5/7739/21247/00985817.pdf`.

[212] C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100. IEEE Computer Society Press, 1999.

[213] A. Perrig, D. Song, and D. Tygar. ELK, a new protocol for efficient large-group key distribution. In *Proc. 2001 IEEE Symposium on Security and Privacy*, pages 247–262. IEEE Computer Society Press, 2001. URL `http://paris.cs.berkeley.`

edu/~perrig/projects.html.

[214] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004. ISSN 0001-0782.

[215] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of the 7th Ann. Int. Conf. on Mobile Computing and Networking*, pages 189–199. ACM Press, 2001. ISBN 1-58113-422-3.

[216] Roger L. Peterson, Rodger E. Ziemer, and David E. Borth. *Introduction to spread spectrum communications*. Prentice Hall, Inc., 1995. ISBN 0-02-431623-7.

[217] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of spread spectrum communications – a tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.

[218] R. Di Pietro, L.V. Mancini, and S. Jajodia. Efficient and secure keys management for wireless mobile communications. In *Proc. 2nd ACM Int. Workshop on Principles of Mobile Computing*, pages 66–73. ACM Press, 2002. ISBN 1-58113-511-4.

[219] R. Di Pietro, L.V. Mancini, and S. Jajodia. Secure selective exclusion in ad hoc wireless network. In M.A. Ghonaimy, M.T. El-Hadidi, and H.K. Aslan, editors, *Security in the Information Society: Visions and Perspectives*, pages 423–434. Kluwer Academic Publishers, 2002.

[220] R. Di Pietro, L.V. Mancini, and A. Mei. Random key assignments for secure wireless sensor networks. In *1st ACM Workshop on Security of Ad-hoc and Sensor Networks*, pages 62–71. ACM Press, October 2003. ISBN 1-58113-783-4.

[221] R. Di Pietro, L.V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan. Connectivity properties of secure wireless sensor networks. In *2nd ACM workshop on Security of ad hoc and sensor networks*, pages 53–58. ACM Press, 2004. ISBN 1-58113-972-1.

[222] L. A. Pinnaduwage, V. Boiadjiev, J. E. Hawk, and T. Thundat. Sensitive detection of plastic explosives with self-assembled monolayer-coated microcantilevers. *Applied Physics Letters, Vol. 83, No. 7, pp. 1471–1473, 18 August 2003*, 83(7): 1471–1473, August 2003.

[223] K. Pister. UCB/MLB 29 Palms UAV-Dropped Sensor Network Demo. http://robotics.eecs.berkeley.edu/~pister/29Palms0103, 2001.

[224] Richard A. Poisel. *Modern Communications Jamming Principles and Techniques*. The Artech House Information Warfare Library. Artech House Publishers, 2003. ISBN 158053743X.

[225] Joe Polastre. Private communication, 2005.

[226] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM Press, 2004. ISBN 1-58113-879-2.

[227] Sofie Pollin, Bruno Bougard, Rahul Mangharam, Liesbet Van der Perre, Francky Catthoor, Raghunathan Rajkumar, and Ingrid Moerman. Optimizing transmission and shutdown for energy-efficient packet scheduling in sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 290–301. IEEE, February 2005.

[228] G.J. Pottie and W.J. Kaiser. Embedding the Internet: Wireless Integrated Sensor Networks. *Communications of the ACM*, 43 (5):51–58, May 2000.

[229] Gregory J. Pottie and Loren P. Clare. Wireless integrated network sensors: toward low-cost and robust self-organizing security networks. In *Sensors, C3I, Information, and Training Technologies for Law Enforcement*, volume 3577 of *SPIE Proceedings*, pages 86–95, 1999. ISBN 0-8194-3043-9. URL http://wins.rsc.rockwell.com/publications/spie3577-12.pdf.

[230] B. Preneel. Cryptographic primitives for information authentication - state of the art. In B. Preneel and V. Rijmen, editors, *State of the Art in Appplied Cryptography*, volume 1528 of *LNCS*, pages 50–105. Springer-Verlag, 1998.

[231] B. Preneel, A. Biryukov, E. Oswald, B. Van Rompay, L. Granboulan, E. Dottax, S. Murphy, A. Dent, J. White, M. Dichtl, S. Pyka, M. Schafheutle, P. Serf, E. Biham, E. Barkan, O. Dunkelman, J.-J. Quisquater, M. Ciet, F. Sica, L. Knudsen, M. Parker, and H. Raddum. NESSIE Security Report. Deliverable D20, NESSIE Consortium, February 2003.

[232] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM Press, 2003. ISBN 1-58113-707-9.

[233] J. Rabaey, E. Arens, C. Federspiel, A. Gadgil, D. Messerschmitt, W. Nazaroff, K. Pister, S. Oren, and P. Varaiya. Smart energy distribution and consumption: Information technology as an enabling force. White paper, CITRIS, 2001. URL http://citris.berkeley.edu/SmartEnergy/SmartEnergy.pdf.

[234] Jan M. Rabaey, M. Josie Ammer, Julio L. da Silva, Danny Patel, and Shad Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *Computer*, 33(7):42–48, 2000. ISSN 0018-9162.

[235] Rachel Cardell-Oliver and Keith Smettem and Mark Kranz and Kevin Mayer. Field testing a wireless sensor network for reactive environmental monitoring. In *International Conference on Intelligent Sensors, Sensor Networks & Information*. IEEE, December 2004.

[236] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: an efficient protocol for group key management. In Jon Crowcroft and Marcus Hofmann, editors, *Proceedings of the Third International COST264 Workshop (NGC 2001)*, volume 2233 of *LNCS*, pages 159–171. Springer-Verlag, 2001. URL http://citeseer.nj.nec.com/448864.html.

[237] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 181–192. ACM Press, 2003. ISBN 1-58113-707-9.

[238] T.S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2nd edition, 1996. ISBN 0130422320.

[239] Ben Reichardt and David Wagner. Markov Truncated Differential Cryptanalysis of Skipjack. In *Selected Areas in Cryptography: 9th Annual International Workshop (SAC 2002)*, volume 2595 of *LNCS*, pages 110–128. Springer-Verlag, 2002.

[240] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *1st IEEE Int. Conf. on Mobile Ad hoc and Sensor Systems (MASS '04)*. IEEE Computer Society Press, 2004.

[241] *ASH Transceiver Software Designer's Guide*. RF Monolithics, Inc., 2002. URL `http://www.rfm.com/products/tr_swg05.pdf`.

[242] RF Monolithics, Inc. TR1000: 916.50 MHz Hybrid Transceiver. Datasheet, 2002. URL `http://www.rfm.com/products/data/tr1000.pdf`.

[243] RF Monolithics, Inc. TR1001: 868.35 MHz Hybrid Transceiver. Datasheet, 2002. URL `http://www.rfm.com/products/data/tr1001.pdf`.

[244] *ASH Transceiver Designer's Guide*. RF Monolithics, Inc., 2004. URL `http://www.rfm.com/products/tr_des24.pdf`.

[245] R.L. Rivest. The RC5 Encryption Algorithm. In *Proc. 1994 Leuven Workshop on Fast Software Encryption*, pages 86–96. Springer-Verlag, 1995. URL `http://theory.lcs.mit.edu/~rivest/Rivest-rc5rev.pdf`.

[246] R.L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin. The RC6^{TM} Block Cipher. Specification version 1.1, August 1998. URL `http://www.rsasecurity.com/rsalabs/rc6/`.

[247] S. Roundy, D. Steingart, L. Fréchette, P. K. Wright, and J. Rabaey. Power sources for wireless networks. In *1st European Workshop on Wireless Sensor Networks (EWSN '04)*, January 2004. URL `http://www.eureka.gme.usherb.ca/memslab/docs/PowerReview-2.pdf`.

[248] David Ruppe. Nations to discuss using nuclear test sensors as tsunami warning system. Global Security Newswire, January 2005. URL `http://www.nti.org/d_newswire/issues/print.asp?story_id=5FDDA53C-7385-41B0-A0D5-47652595F5CE`.

[249] F. Sano, M. Koike, S. Kawamura, and M. Shiba. Performance evaluation of aes finalists on the high-end smart card. In *Proc. 3rd AES Conference (AES3)*, 2001.

[250] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 78–89. IEEE Computer Society, 2002. ISBN 0-7695-1856-7.

[251] Naveen Sastry and David Wagner. Security considerations for IEEE 802.15.4 networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 32–42, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-925-X.

[252] D. Curtis Schleher. *Electronic Warfare in the Information Age*. Artech House, July 1999. ISBN 0890065268.

[253] B. Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*, LNCS, pages 191–204. Springer-Verlag, 1994.

[254] B. Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, Inc., 2nd edition, 1996. ISBN 0-471-12845-7.

[255] B. Schneier. AES News. Crypto-gram newsletter, Counterpane Internet Security, Inc., September 2002. URL `http://www.counterpane.com/crypto-gram-0209.html`.

[256] B. Schneier. More on AES Cryptanalysis. Crypto-gram newsletter, Counterpane Internet Security, Inc., October 2002. URL `http://www.counterpane.com/crypto-gram-0210.html`.

[257] B. Schneier and D. Whiting. A performance comparison of the five AES finalists. Proc. 3rd AES Conference (AES3), 2001. URL `http://csrc.nist.gov/encryption/aes/round2/conf3/papers/17-bschneier.pdf`.

[258] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-Bit Block Cipher. `http://www.schneier.com/paper-twofish-paper.pdf`, June 1998.

[259] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. On the Twofish Key Schedule. In S.E. Tavares and H. Meijer, editors, *Selected Areas in Cryptography '98, SAC'98*, volume 1556 of *LNCS*, pages 27–42. Springer-Verlag, 1999.

[260] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*. Wiley, 1999. ISBN 0471353817.

[261] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6), December 2004.

[262] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 272–287. ACM Press, 2001. ISBN 1-58113-422-3.

[263] T. Shimoyama, M. Takenaka, and T. Koshiba. Multiple linear cryptanalysis of a reduced round RC6. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002*, volume 2365, pages 76–88. Springer-

Verlag, 2002.

[264] T. Shimoyama, K. Takeuchi, and J. Hayakawa. Correlation Attack to the Block Cipher RC5 and the Simplified Variants of RC6. In *Proc. 3rd AES Conference (AES3)*, 2000.

[265] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM Press, 2004. ISBN 1-58113-879-2.

[266] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, M.B. Srivastava, and M. Potkonjak. On communication security in wireless ad-hoc sensor networks. In *11th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 139–144, June 2002. URL `http://ieeexplore.ieee.org/iel5/8006/22131/01030000.pdf`.

[267] Joel H. Spencer. *The Strange Logic of Random Graphs*. Springer-Verlag, 2001. ISBN 3540416544.

[268] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of The 7th International Workshop on Security Protocols*, volume 1796 of *LNCS*, pages 172–194. Springer-Verlag, 2000. URL `http://www.cl.cam.ac.uk/~fms27/papers/duckling.pdf`.

[269] W. Stallings. *Network and Internetwork Security : Principles and Practice*. Prentice Hall, 2nd edition, 1995. ISBN 0024154830.

[270] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *Proc. 18th Int. Conf. on Distributed Computing Systems*, pages 380–387, 1998.

[271] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, 2000. URL `http://citeseer.nj.nec.com/steiner00key.html`.

[272] D.R. Stinson. Some observations on the theory of cryptographic hash functions. Cryptology ePrint Archive, Report 2001/020, 2001.

[273] M. Ståhlberg. Radio jamming attacks against two popular mobile networks. In H. Lipmaa and H. Pehu-Lehtonen, editors, *Proceedings of the Helsinki University of Technology. Seminar on Network Security. Mobile Security*. Helsinki University of Technology, Fall 2000. URL `http://www.tml.hut.fi/Opinnot/Tik-110.501/2000/papers/stahlberg.pdf`.

[274] M. Sugita, K. Kobara, and H. Imai. Security of Reduced Version of the Block Cipher Camellia against Truncated and Impossible Differential Cryptanalysis. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *LNCS*, pages 193–207. Springer-Verlag, 2001.

[275] Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. Lessons from a sensor network expedition. In *Proc. 1st European Workshop Wireless Sensor Networks (EWSN 04)*, volume 2920 of *LNCS*, pages 307–322. Springer-Verlag, 2004.

[276] M. Takenaka, T. Shimoyama, and T. Koshiba. Theoretical Analysis of "Correlations in RC6". Cryptology ePrint Archive: Report 2002/176, 2002. URL `http://eprint.iacr.org/2002/176/`.

[277] M. Takenaka, T. Shimoyama, and T. Koshiba. Theoretical Analysis of χ^2 Attack on RC6. In *Proc. 8th Australasian Conference on Information Security and Privacy (ACISP2003)*, volume 2727 of *LNCS*, pages 142–153. Springer-Verlag, July 2003.

[278] H. Tanaka, C. Ishii, and T. Kaneko. On the strength of KASUMI without FL functions against Higher Order Differential Attack. In *3rd International Conference on Information Security and Cryptology, ICISC 2000*, volume 2015 of *LNCS*, pages 14–21. Springer-Verlag, 2001.

[279] Texas Instruments, Inc. MSP430x13x, MSP430x14x Mixed Signal Microcontroller. Datasheet, 2001. URL `http://www-s.ti.com/sc/ds/msp430f149.pdf`.

[280] *MSP430x1xx Family User's Guide*. Texas Instruments, Inc., December 2003. URL `http://www-s.ti.com/sc/psheets/slau049d/slau049d.pdf`.

[281] Don J. Torrieri. *Principles of Secure Communication Systems*. Artech House, Inc., 0-89006-139-4 1985.

[282] Don J. Torrieri. Fundamental limitations on repeater jamming of frequency-hopping communications. *IEEE Journal on Selected Areas in Communications*, 7(4):569–575, May 1989.

[283] K.H. Tovmark. *Chipcon Application Note AN014: Frequency Hopping Systems (Rev. 1.0)*. Chipcon AS, March 2002. URL `http://www.chipcon.com/files/AN_014_Frequency_Hopping_Systems_1_0.pdf`.

[284] Tri Van Le. Novel Cyclic and Algebraic Properties of AES. Cryptology ePrint Archive: Report 2003/108, May 2003. URL `http://eprint.iacr.org/2003/108/`.

[285] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 171–180. ACM Press, 2003. ISBN 1-58113-707-9.

[286] L.F.W. van Hoesel, S.O. Dulman, P.J.M. Havinga, and H.J. Kip. Design of a low-power testbed for wireless sensor networks and verification. Technical Report TR-CTIT-03-45, Centre for Telematics and Information Technology, University of Twente, The Netherlands, September 2003.

[287] L.F.W. van Hoesel and P.J.M. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. In *INSS*, June 2004.

[288] J.H. van Lint and R.M. Wilson. *A course in combinatorics*. Cambridge University Press, 2nd edition, 2001. ISBN 0-521-00601-5.

[289] K.L. Vantran. WolfPack Proves Strength in Numbers. DefenseLINK News, August 2003. URL http://www.defenselink.mil/news/Aug2003/n08142003_200308147.html.

[290] M. Čagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On Cheating in CSMA/CA Ad Hoc Networks. Technical Report IC/2004/27, EPFL-IC, 2004.

[291] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM Press, 2004. ISBN 1-58113-972-1.

[292] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. RFC 2627, IETF, June 1999.

[293] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*, pages 324–328. IEEE Computer Society Press, 2005.

[294] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 28–39, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-707-9.

[295] M. Weiser. The computer for the 21st century. *Scientific American*, pages 94–104, September 1991. URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[296] M. Weiser. The computer for the 21st century. *Pervasive Computing*, 1(1):19–25, 2002.

[297] Geoffrey Werner-Allen, Matt Welsh, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Monitoring volcanic eruptions with a wireless sensor network. Technical Report 27-04, Harvard University, 2004. URL http://www.eecs.harvard.edu/~werner/projects/volcano/.

[298] Doug Whiting. http://www.schneier.com/code/twofish-optimized-c.zip, 1998.

[299] C.K. Wong, M. Gouda, and S.S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking (TON)*, 8(1):16–30, 2000. ISSN 1063-6692. URL http://citeseer.nj.nec.com/wong98secure.html.

[300] A.D. Wood and J.A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.

[301] J. Worley, B. Worley, T. Christian, and C. Worley. AES Finalists on PA-RISC and IA-64: Implementations & Performance. In *Proc. 3rd AES Conference (AES3)*, 2001.

[302] Christopher R. Wren and Srinivasa G. Rao. Self-configuring, lightweight sensor networks for ubiquitous computing. Technical Report TR2003-24, MITSUBISHI ELECTRIC RESEARCH LABORATORIES, October 2003. URL http://www.merl.com/reports/docs/TR2003-24.pdf.

[303] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *ACM MobiHoc '05*, page To appear. ACM Press, 2005.

[304] Wenyuan Xu, Timothy Wood, Wade Trappe, and Yanyong Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 80–89, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-925-X.

[305] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84. ACM Press, 2001. ISBN 1-58113-422-3.

[306] Qi Xue and Aura Ganz. Runtime Security Composition for Sensor Networks (SecureSense). In *IEEE Vehicular Technology Conference (VTC Fall 2003)*, 2003.

[307] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: challenges and solutions. *IEEE Wireless Communications*, 11(1):38–47, February 2004.

[308] S. Yang. Redwoods go high tech: Researchers use wireless sensors to study California's state tree. Press release, University of California, Berkeley, July 2003. URL http://www.berkeley.edu/news/media/releases/2003/07/28_redwood.shtml.

[309] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In *Proc. IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE. URL http://www.isi.edu/~johnh/PAPERS/Ye02a.html.

[310] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2003. URL http://ieeexplore.ieee.org/iel5/90/29000/01306496.pdf.

[311] Y. Yeom, S. Park, and I. Kim. On the Security of CAMELLIA against the Square Attack. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002*, volume 2365 of *LNCS*, pages 128–142. Springer-Verlag, 2002.

[312] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 299–302. ACM Press, 2001. ISBN 1-58113-428-2. URL http://dx.doi.org/10.1145/501449.501464.

[313] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proc. IEEE INFOCOM*, volume 2, pages 1312–1321, April 2003.

[314] A.M. Youssef and S.E. Tavares. On Some Algebraic Structures in the AES Round Function. Cryptology ePrint Archive: Report 2002/144, September 2002. URL `http://eprint.iacr.org/2002/144/`.

[315] M.G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM workshop on Wireless security*, pages 1–10. ACM Press, 2002. ISBN 1-58113-585-8.

[316] H. Zhang and J. Hou. On deriving the upper bound of α-lifetime for large sensor networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 121–132. ACM Press, 2004. ISBN 1-58113-849-0.

[317] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in ZebraNet. In *2nd International Conference on Embedded Networked Sensor Systems*, pages 227–238. ACM Press, 2004. ISBN 1-58113-879-2.

[318] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-707-9.

[319] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[320] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *10th ACM Conference on Computer and Communications Security (CCS '03)*, pages 62–72. ACM Press, 2003. ISBN 1-58113-738-9.

Titles in the IPA Dissertation Series

J.O. Blanco. *The State Operator in Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1996-01

A.M. Geerling. *Transformational Development of Data-Parallel Algorithms*. Faculty of Mathematics and Computer Science, KUN. 1996-02

P.M. Achten. *Interactive Functional Programs: Models, Methods, and Implementation*. Faculty of Mathematics and Computer Science, KUN. 1996-03

M.G.A. Verhoeven. *Parallel Local Search*. Faculty of Mathematics and Computing Science, TUE. 1996-04

M.H.G.K. Kesseler. *The Implementation of Functional Languages on Parallel Machines with Distrib. Memory*. Faculty of Mathematics and Computer Science, KUN. 1996-05

D. Alstein. *Distributed Algorithms for Hard Real-Time Systems*. Faculty of Mathematics and Computing Science, TUE. 1996-06

J.H. Hoepman. *Communication, Synchronization, and Fault-Tolerance*. Faculty of Mathematics and Computer Science, UvA. 1996-07

H. Doornbos. *Reductivity Arguments and Program Construction*. Faculty of Mathematics and Computing Science, TUE. 1996-08

D. Turi. *Functorial Operational Semantics and its Denotational Dual*. Faculty of Mathematics and Computer Science, VUA. 1996-09

A.M.G. Peeters. *Single-Rail Handshake Circuits*. Faculty of Mathematics and Computing Science, TUE. 1996-10

N.W.A. Arends. *A Systems Engineering Specification Formalism*. Faculty of Mechanical Engineering, TUE. 1996-11

P. Severi de Santiago. *Normalisation in Lambda Calculus and its Relation to Type Inference*. Faculty of Mathematics and Computing Science, TUE. 1996-12

D.R. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. Faculty of Mathematics and Computing Science, TUE. 1996-13

M.M. Bonsangue. *Topological Dualities in Semantics*. Faculty of Mathematics and Computer Science, VUA. 1996-14

B.L.E. de Fluiter. *Algorithms for Graphs of Small Treewidth*. Faculty of Mathematics and Computer Science, UU. 1997-01

W.T.M. Kars. *Process-algebraic Transformations in Context*. Faculty of Computer Science, UT. 1997-02

P.F. Hoogendijk. *A Generic Theory of Data Types*. Faculty of Mathematics and Computing Science, TUE. 1997-03

T.D.L. Laan. *The Evolution of Type Theory in Logic and Mathematics*. Faculty of Mathematics and Computing Science, TUE. 1997-04

C.J. Bloo. *Preservation of Termination for Explicit Substitution*. Faculty of Mathematics and Computing Science, TUE. 1997-05

J.J. Vereijken. *Discrete-Time Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1997-06

F.A.M. van den Beuken. *A Functional Approach to Syntax and Typing*. Faculty of Mathematics and Informatics, KUN. 1997-07

A.W. Heerink. *Ins and Outs in Refusal Testing*. Faculty of Computer Science, UT. 1998-01

G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. Faculty of Mechanical Engineering, TUE. 1998-02

J. Verriet. *Scheduling with Communication for Multiprocessor Computation*. Faculty of Mathematics and Computer Science, UU. 1998-03

J.S.H. van Gageldonk. *An Asynchronous Low-Power 80C51 Microcontroller*. Faculty of Mathematics and Computing Science, TUE. 1998-04

A.A. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. Faculty of Mathematics and Computing Science, TUE. 1998-05

E. Voermans. *Inductive Datatypes with Laws and Subtyping – A Relational Model*. Faculty of Mathematics and Computing Science, TUE. 1999-01

H. ter Doest. *Towards Probabilistic Unification-based Parsing*. Faculty of Computer Science, UT. 1999-02

J.P.L. Segers. *Algorithms for the Simulation of Surface Processes*. Faculty of Mathematics and Computing Science, TUE. 1999-03

C.H.M. van Kemenade. *Recombinative Evolutionary Search*. Faculty of Mathematics and Natural Sciences, UL. 1999-04

E.I. Barakova. *Learning Reliability: a Study on Indecisiveness in Sample Selection*. Faculty of Mathematics and Natural Sciences, RUG. 1999-05

M.P. Bodlaender. *Scheduler Optimization in Real-Time Distributed Databases*. Faculty of Mathematics and Computing Science, TUE. 1999-06

M.A. Reniers. *Message Sequence Chart: Syntax and Semantics*. Faculty of Mathematics and Computing Science, TUE. 1999-07

J.P. Warners. *Nonlinear approaches to satisfiability problems*. Faculty of Mathematics and Computing Science, TUE. 1999-08

J.M.T. Romijn. *Analysing Industrial Protocols with Formal Methods*. Faculty of Computer Science, UT. 1999-09

P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. Faculty of Computer Science, UT. 1999-10

G. Fábián. *A Language and Simulator for Hybrid Systems*. Faculty of Mechanical Engineering, TUE. 1999-11

J. Zwanenburg. *Object-Oriented Concepts and Proof Rules*. Faculty of Mathematics and Computing Science, TUE. 1999-12

R.S. Venema. *Aspects of an Integrated Neural Prediction System*. Faculty of Mathematics and Natural Sciences, RUG. 1999-13

J. Saraiva. *A Purely Functional Implementation of Attribute Grammars*. Faculty of Mathematics and Computer Science, UU. 1999-14

R. Schiefer. *Viper, A Visualisation Tool for Parallel Program Construction*. Faculty of Mathematics and Computing Science, TUE. 1999-15

K.M.M. de Leeuw. *Cryptology and Statecraft in the Dutch Republic*. Faculty of Mathematics and Computer Science, UvA. 2000-01

T.E.J. Vos. *UNITY in Diversity. A stratified approach to the verification of distributed algorithms*. Faculty of Mathematics and Computer Science, UU. 2000-02

W. Mallon. *Theories and Tools for the Design of Delay-Insensitive Communicating Processes*. Faculty of Mathematics and Natural Sciences, RUG. 2000-03

W.O.D. Griffioen. *Studies in Computer Aided Verification of Protocols*. Faculty of Science, KUN. 2000-04

P.H.F.M. Verhoeven. *The Design of the MathSpad Editor*. Faculty of Mathematics and Computing Science, TUE. 2000-05

J. Fey. *Design of a Fruit Juice Blending and Packaging Plant*. Faculty of Mechanical Engineering, TUE. 2000-06

M. Franssen. *Cocktail: A Tool for Deriving Correct Programs*. Faculty of Mathematics and Computing Science, TUE. 2000-07

P.A. Olivier. *A Framework for Debugging Heterogeneous Applications*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2000-08

E. Saaman. *Another Formal Specification Language*. Faculty of Mathematics and Natural Sciences, RUG. 2000-10

M. Jelasity. *The Shape of Evolutionary Search Discovering and Representing Search Space Structure*. Faculty of Mathematics and Natural Sciences, UL. 2001-01

R. Ahn. *Agents, Objects and Events a computational approach to knowledge, observation and communication*. Faculty of Mathematics and Computing Science, TU/e. 2001-02

M. Huisman. *Reasoning about Java programs in higher order logic using PVS and Isabelle*. Faculty of Science, KUN. 2001-03

I.M.M.J. Reymen. *Improving Design Processes through Structured Reflection*. Faculty of Mathematics and Computing Science, TU/e. 2001-04

S.C.C. Blom. *Term Graph Rewriting: syntax and semantics*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2001-05

R. van Liere. *Studies in Interactive Visualization*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2001-06

A.G. Engels. *Languages for Analysis and Testing of Event Sequences*. Faculty of Mathematics and Computing Science, TU/e. 2001-07

J. Hage. *Structural Aspects of Switching Classes*. Faculty of Mathematics and Natural Sciences, UL. 2001-08

M.H. Lamers. *Neural Networks for Analysis of Data in Environmental Epidemiology: A Case-study into Acute Effects of Air Pollution Episodes*. Faculty of Mathematics and Natural Sciences, UL. 2001-09

T.C. Ruys. *Towards Effective Model Checking*. Faculty of Computer Science, UT. 2001-10

D. Chkliaev. *Mechanical verification of concurrency control and recovery protocols*. Faculty of Mathematics and Computing Science, TU/e. 2001-11

M.D. Oostdijk. *Generation and presentation of formal mathematical documents*. Faculty of Mathematics and Computing Science, TU/e. 2001-12

A.T. Hofkamp. *Reactive machine control: A simulation approach using χ*. Faculty of Mechanical Engineering, TU/e. 2001-13

D. Bošnački. *Enhancing state space reduction techniques for model checking*. Faculty of Mathematics and Computing Science, TU/e. 2001-14

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects*. Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems*. Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity*. Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing*. Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. Faculty of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing*. Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms*. Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining*. Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL*. Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand*. Faculty of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks*. Faculty of Mathematics and Natural Sciences, UL. 2003-04

T.A.C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing*. Faculty of Mathematics and Computer Science, TU/e. 2003-05

S.V. Nedea. *Analysis and Simulations of Catalytic Reactions*. Faculty of Mathematics and Computer Science, TU/e. 2003-06

M.E.M. Lijding. *Real-time Scheduling of Tertiary Storage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-07

H.P. Benz. *Casual Multimedia Process Annotation – CoMPAs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-08

D. Distefano. *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-09

M.H. ter Beek. *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components*. Faculty of Mathematics and Natural Sciences, UL. 2003-10

D.J.P. Leijen. *The λ Abroad – A Functional Approach to Software Components*. Faculty of Mathematics and Computer Science, UU. 2003-11

W.P.A.J. Michiels. *Performance Ratios for the Differencing Method*. Faculty of Mathematics and Computer Science, TU/e. 2004-01

G.I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving*. Faculty of Mathematics and Computer Science, TU/e. 2004-02

P. Frisco. *Theory of Molecular Computing – Splicing and Membrane systems*. Faculty of Mathematics and Natural Sciences, UL. 2004-03

S. Maneth. *Models of Tree Translation*. Faculty of Mathematics and Natural Sciences, UL. 2004-04

Y. Qian. *Data Synchronization and Browsing for Home Environments*. Faculty of Mathematics and Computer Science and Faculty of Industrial Design, TU/e. 2004-05

F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-06

L. Cruz-Filipe. *Constructive Real Analysis: a Type-Theoretical Formalization and Applications*. Faculty of Science, Mathematics and Computer Science, KUN. 2004-07

E.H. Gerding. *Autonomous Agents in Bargaining Games: An Evolutionary Investigation of Fundamentals, Strategies, and Business Applications*. Faculty of Technology Management, TU/e. 2004-08

N. Goga. *Control and Selection Techniques for the Automated Testing of Reactive Systems*. Faculty of Mathematics and Computer Science, TU/e. 2004-09

M. Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs*. Faculty of Science, Mathematics and Computer Science, RU. 2004-10

A. Löh. *Exploring Generic Haskell*. Faculty of Mathematics and Computer Science, UU. 2004-11

I.C.M. Flinsenberg. *Route Planning Algorithms for Car Navigation*. Faculty of Mathematics and Computer Science, TU/e. 2004-12

R.J. Bril. *Real-time Scheduling for Media Processing Using Conditionally Guaranteed Budgets*. Faculty of Mathematics and Computer Science, TU/e. 2004-13

J. Pang. *Formal Verification of Distributed Systems*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-14

F. Alkemade. *Evolutionary Agent-Based Economics*. Faculty of Technology Management, TU/e. 2004-15

E.O. Dijk. *Indoor Ultrasonic Position Estimation Using a Single Base Station*. Faculty of Mathematics and Computer Science, TU/e. 2004-16

S.M. Orzan. *On Distributed Verification and Verified Distribution*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-17

M.M. Schrage. *Proxima - A Presentation-oriented Editor for Structured Documents*. Faculty of Mathematics and Computer Science, UU. 2004-18

E. Eskenazi and A. Fyukov. *Quantitative Prediction of Quality Attributes for Component-Based Software Architectures*. Faculty of Mathematics and Computer Science, TU/e. 2004-19

P.J.L. Cuijpers. *Hybrid Process Algebra*. Faculty of Mathematics and Computer Science, TU/e. 2004-20

N.J.M. van den Nieuwelaar. *Supervisory Machine Control by Predictive-Reactive Scheduling*. Faculty of Mechanical Engineering, TU/e. 2004-21

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support- .* Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication*. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22