Access Control in Wireless Sensor Networks

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Zinaida Benenson

aus Moskau

Mannheim, 2008

Dekan: Professor Dr. Ing. Felix Freiling, Universität Mannheim
Referent: Professor Dr. Ing. Felix Freiling, Universität Mannheim
Korreferent: Professor Dr. Adrian Perrig, Carnegie Mellon University

Tag der mündlichen Prüfung: 19. Dezember 2008

Abstract

Wireless sensor networks consist of a large amount of sensor nodes, small low-cost wireless computing devices equipped with different sensors. Sensor networks collect and process environmental data and can be used for habitat monitoring, precision agriculture, wildfire detection, structural health monitoring and many other applications.

Securing sensor networks calls for novel solutions, especially because of their unattended deployment and strong resource limitations. Moreover, developing security solutions without knowing precisely against what threats the system should be protected is impossible. Thus, the first task in securing sensor networks is to define a realistic adversary model. We systematically investigate vulnerabilities in sensor networks, specifically focusing on physical attacks on sensor node hardware. These are all attacks that require direct physical access to the sensor nodes. Most severe attacks of this kind are also known as node capture, or node compromise. Based on the vulnerability analysis, we present a novel general adversary model for sensor networks.

If the data collected within a sensor network is valuable or should be kept confidential then the data should be protected from unauthorized access. We determine security issues in the context of access control in sensor networks in presence of node capture attacks and develop protocols for broadcast authentication that constitute the core of our solutions for access control.

We develop broadcast authentication protocols for the case where the adversary can capture up to some threshold t sensor nodes. The developed protocols offer absolute protection while not more than t nodes are captured, but their security breaks completely otherwise. Moreover, security in this case comes at a high cost, as the resource requirements for the protocols grow rapidly with t.

One of the most popular ways to overcome impossibility or inefficiency of solutions in distributed systems is to make the protocol goals probabilistic. We therefore develop efficient probabilistic protocols for broadcast authentication. Security of these protocols degrades gracefully with the increasing number of captured nodes. We conclude that the perfect threshold security is less appropriate for sensor networks than the probabilistic approach. Gracefully degrading security offers better scalability and saves resources, and should be considered as a promising security paradigm for sensor networks.

Zusammenfassung

Drahtlose Sensornetze messen und verarbeiten Umgebungsparameter wie z.B. Temperatur, Helligkeit oder Luftdruck. Die erhobenen Daten können in verschiedenen Anwendungen benutzt werden, z.B. zur Beobachtung von wilden Tieren in ihrem natürlichen Lebensraum, zur Früherkennung von Waldbränden oder zur Gebäudeüberwachung.

Sicherheitsanforderungen an Sensornetze sind schwer zu erfüllen, insbesondere weil die Sensornetze unbeaufsichtigt funktionieren sollen, und die Sensorknoten sehr ressourcenbeschränkt sind. Außerdem gibt es keine bewährten Angreifermodelle für Sensornetze, da entsprechende Erfahrungen mit Anwendungen in der Praxis bisher fehlen. Folglich war das erste Ziel dieser Arbeit, realitätsnahe Angreifermodelle für Sensornetze zu definieren. Zu diesem Zweck wurden systematisch Angriffe auf Sensorknoten untersucht. Anhand der gewonnenen Erkentnisse wurde daraufhin ein neues, erweiterebares Rahmenwerk für Angreifermodelle in Sensornetzen entwickelt.

Ein weiteres wichtiges Thema in Sensornetzen ist Zugriffskontrolle, denn die gesammelten Daten sind oft wertvoll oder sensibel. Insbesondere schwierig gestaltet sich die Absicherung von Sensornetzen falls der Angreifer in der Lage ist, einige Sensorknoten vollständig unter seine Kontrolle zu bringen.

In dieser Arbeit wurden zuerst Protokolle für Zugriffskontrolle entwickelt, die sicher gegen einen Angreifer sind, der eine bestimmte Anzahl t von Sensorknoten übernimmt. Diese Protokolle leisten sehr hohe Sicherheit solange nicht mehr als t Knoten übernommen wurden, versagen aber komplett, falls der Angreifer mehr als t Knoten in seine Gewalt bringt. Leider verbrauchen diese Protokolle für eine realistische Anzahl übernommener Knoten (z.B., mehr als 10 Knoten) sehr viele Ressourcen und sind demnach nicht optimal für batteriebetriebene Sensornetze.

Randomisierung ist eine bekannte Methode zur Verbesserung der Effizienz von verteilten Algorithmen. Aus diesem Grund wurden ferner in dieser Arbeit probabilistische Protokolle zur Zugriffskontrolle in Sensornetzen entwickelt. Diese Protokolle sind tatsächlich viel effizienter als die oben genannten deterministischen Verfahren. Außerdem sind die entwickelten probabilistischen Protokolle graduell abstufbar, d.h. sie bieten hohe Sicherheit falls wenige Knoten übernommen wurden und bieten immer niederigere Sicherheit mit steigender Anzahl von kompromittierten Knoten. Graduell abstufbare Sicherheit hat in Sensornetzen einen großen Vorteil, da es sehr schwierig ist, eine genaue Grenze für die Anzahl der übernommenen Knoten zu definieren.

Acknowledgements

This thesis resulted from my interactions with a lot of people, both at professional and at private levels. Now the time has come to thank them for their kind help.

I'm deeply grateful to Felix Freiling, my advisor. His way to do research as well as his attitude towards life in general influenced me immensely, and I still feel there is a lot to learn from him. Among other things, I have always admired his ability to discuss research problems from scratch, to formalize rigorously, his scientific bravery in supporting ideas that sounded completely mental at first glance. Most important of all was probably his calm and constant belief in my ability to get a PhD!

As I started looking for an interesting topic for my PhD, I came across an increasing number of papers co-authored by Adrian Perrig. The ideas in these papers and his style of writing have always impressed me and influenced my own style of work. I therefore feel extremely grateful and honored that he accepted to be my second evaluator.

In the first half of 2006 Christian Rohner gave me a great opportunity to work as a project assistant at Uppsala University. I thank Christian very much for his supervision, for research we did together, and for his kind support in the everyday matters in Sweden. I'm also very grateful to Thiemo Voigt for initiating my first contact with Christian, and for his support and hospitality in Sweden.

Dogan Kesdogan helped me a lot as I started my PhD at the RWTH Aachen University. I thank him for his wisdom, for many fruitful discussions, and for co-authored papers.

I'm very grateful to my fellow PhD students Matthias Brust, Maximillian Dornseif, Ioannis Krontiris and Lexi Pimenidis for discussions, moral support, and fruitful cooperation over various papers.

I thank all members of the project ZeuS for interesting and helpful discussions on different aspects of sensor networks.

The members of the Laboratory for Dependable Distributed Systems

at the University of Mannheim created a friendly and supportive research atmosphere that I enjoyed very much. I'm very grateful to all PhD and master students at our chair for conversations and sympathy, to Jürgen Jaap for overall technical support, and to Sabine Braak, the secretary at our chair, for her support in everyday academic life, and also for her cheerfulness and optimism.

I thank the DFG graduate school 643 "Software for mobile communication systems", and the head of the graduate school Prof. Dr. Otto Spaniol in particular, for financial support during my PhD studies at the RWTH Aachen University. I also thank the project ZeuS of the Landesstiftung Baden-Württemberg and the Schlieben-Lange program of the European Social Fund and the Bundesland Baden-Württemberg for financial support during my PhD studies at the University of Mannheim.

Many thanks to all my co-authors for fruitful cooperation, and also for being so different from each other that I really learned a lot from interactions with them. All co-authored papers can be found in the "Publications" section on page ix.

I'm very grateful to master students and student helpers who worked with me on various pieces of this thesis and on other publications.

Some people influenced me profoundly during my undergraduate studies at the University of Trier. Without their support I would have never came up with the idea of starting a PhD. I'm very grateful to Martin Mundhenk for overall encouragement, to Elisabeth Kaiser for supervision in the Ada-Lovelace project, to Frauke Sprengel and to all other members of the MUFFIN21 project, and to my master thesis supervisor Birgit Pfitzmann.

I was very lucky to have received a lot of support in my private life. I thank Anna Chudnovsky, Angelika Glöckner and Susanne Preuschoff for their very valuable help.

I thank Werner Massonne for his love, moral support, patience and understanding, and especially for his humorous and inspiring account on "how I wrote up my PhD in three weeks".

I thank my son Simon for being so understanding and patient with me when I was writing up my PhD, for making me laugh in every possible and impossible situation, and for asking me a lot of impertinent questions such as: "Mummy, how many pages have you written today?"

Finally and most importantly, I thank my parents Mera and Mikhail. They constantly supported me during all these years, and gave me not only their love, but also their time, nerves and energy – in short, an awful lot of practical help. We really did this journey together. Thank you.

Publications

Chapter 2 is based on joint work with Alexander Becher and Maximillian Dornseif [11], and with Peter Cholewinski and Felix Freiling [19].

Chapter 3 is based on Benenson [14], on joint work with Felix Freiling and Dogan Kesdogan [24], and with Nils Gedicke and Ossi Raivio [25].

Chapter 5 is partly based on joint work with Markus de Brün that resulted in his master thesis [32].

Chapter 6 is partly based on joint work with Felix Freiling, Ernest Hammerschmidt, Stefan Lucks and Lexi Pimenidis [21], and also on up to now unpublished work with Christian Rohner and Dirk Stegemann [22].

Publications on the following topics are not incorporated into this thesis: a survey on lower bounds for algorithms for sensor and ad hoc networks [15], clustering in ad hoc and sensor networks (with Matthias Brust, Adrian Andronache and Steffen Rothkugel) [33], metrics for security of cryptographic algorithms (with Ulrich Kühn and Stefan Lucks) [26], secure multiparty computation (with Milan Fort, Felix Freiling, Lucia Draque Penso and Dogan Kesdogan) [58], agreement protocols (with Andreas Achtzehn, Felix Freiling, Birgit Pfitzmann, Christian Rohner and Michael Waidner) [1, 23], secure data storage in sensor networks (with Peter Cholewinski and Felix Freiling) [17, 18], energy-efficient query dissemination in sensor networks (with Markus Bestehorn, Erik Buchmann, Felix Freiling and Marek Jawurek) [16], intrusion detection systems for sensor networks (with Ioannis Krontiris, Felix Freiling, Tassos Dimitriou and Thanassis Giannetsos) [90].

Contents

Acknowledgments **vii**

Publications **ix**

1 Introduction **1**
1.1 Motivation 1
1.1.1 The Need for Access Control 2
1.1.2 The Need for Novel Security Solutions 2
1.1.3 The Need for Realistic Adversary Models 3
1.2 Contributions 4
1.2.1 Real-World Physical Attacks on Sensor Networks . . . 4
1.2.2 Adversary Models for Sensor Networks 5
1.2.3 Access Control to Sensor Network Data 5
1.3 Thesis Outline 6

2 Security Threats and Adversary Models **9**
2.1 Introduction 9
2.2 Related Work 10
2.2.1 Security Goals 10
2.2.2 Physical Attacks 10
2.2.3 Adversary Models 12
2.3 Security Goals in Sensor Networks 13
2.4 Real-World Physical Attacks on Sensor Networks 15
2.4.1 Background on Physical Attacks on Sensor Nodes . . 16
2.4.2 Examples of In-the-Field Attacks and Countermeasures 18
2.4.3 Summary 23
2.5 Adversary Models 25
2.5.1 Overview 25
2.5.2 Presence 27

2.5.3 Intervention 28
2.5.4 Adversary Model Lattice 29
2.5.5 Summary 31
2.6 Discussion of Protection Mechanisms 31
2.7 Conclusions 33

3 Access Control Issues in Wireless Sensor Networks 35
3.1 Introduction 35
3.2 System and Adversary Model 37
3.3 Access Control Problems in Sensor Networks 38
3.3.1 Access Control via Base Station 38
3.3.2 Decentralized Access Control 39
3.4 Two Phases of Access Control to Sensor Network Data 40
3.4.1 User Access Control 40
3.4.2 Authenticated Querying 42
3.4.3 Putting the Two Phases Together 43
3.4.4 Choice of the Public-Key Cryptosystem 45
3.5 Design Space for Access Control 45
3.5.1 Base Station Access Control 46
3.5.2 Decentralized Access Control 49
3.6 Conclusions 50

4 Background And Related Work on WSN Security 51
4.1 Introduction 51
4.2 Query Processing Systems 52
4.3 Query Dissemination 53
4.4 Access Control 54
4.5 Key Establishment 55
4.6 Broadcast Authentication 56
4.6.1 Desirable Properties for Broadcast Authentication . . 56
4.6.2 Broadcast Authentication using Time Synchronization 57
4.6.3 Broadcast Authentication using Public-Key Cryptography 58
4.6.4 Broadcast Authentication using Symmetric-Key Cryptography 59
4.7 Secure Code Update 60
4.8 Authenticated Reports 61
4.9 Authenticated Aggregation 62
4.10 Conclusions 63

5 t-Robust Access Control in WSNs 65
5.1 Introduction . . . 65
5.2 Preliminaries . . . 66
5.2.1 Polynomial-based Key Pre-distribution . . . 67
5.2.2 Interleaved Hop-by-Hop Authentication . . . 68
5.3 System and Adversary Model . . . 71
5.4 Interleaved Authenticated Querying . . . 71
5.4.1 Introduction . . . 71
5.4.2 Basic Interleaved Authenticated Querying . . . 72
5.4.3 Analysis of the Basic Interleaved Scheme . . . 78
5.4.4 Tree-based Interleaved Authenticated Querying . . . 82
5.4.5 Analysis of the Tree-Based Scheme . . . 86
5.4.6 Optimality of the Tree-Based Scheme. . . . 90
5.4.7 Comparison to Related Work . . . 91
5.4.8 Summary . . . 92
5.5 t-Robust Access Control . . . 93
5.5.1 t-Robust Access Control via Base Station . . . 93
5.5.2 t-Robust Decentralized Access Control . . . 93
5.6 Conclusions . . . 95

6 Gracefully Degrading Access Control in WSNs 97
6.1 Introduction . . . 97
6.2 Preliminaries . . . 98
6.2.1 ID-Based Random Key Predistribution . . . 98
6.2.2 1-bit MACs . . . 99
6.3 System and Adversary Model . . . 99
6.4 Basic Authenticated Query Flooding . . . 100
6.4.1 Protocol bAQF . . . 100
6.4.2 Analysis . . . 101
6.4.3 Choosing Parameters . . . 105
6.4.4 Simulation Results . . . 107
6.4.5 Improving Efficiency . . . 112
6.4.6 Attacks and Countermeasures . . . 112
6.4.7 Comparison to Related Work . . . 116
6.4.8 Summary . . . 116
6.5 Simple Authenticated Query Flooding . . . 117
6.5.1 Protocol `sAQF` . . . 117
6.5.2 Analysis . . . 118
6.5.3 Choosing Parameters . . . 120
6.5.4 Simulation Results . . . 122

6.5.5 Summary 124
6.6 Conclusions 124

7 Conclusions and Future Work **127**

Bibliography **133**

List of Figures

2.1 General schematic view of sensor node hardware. 16
2.2 Current sensor node hardware: Mica 2 by Crossbow, Berkeley [103]; Tmote sky by moteiv, Berkeley [111]; and Embedded Sensor Board by ScatterWeb, Berlin [27] 17
2.3 Design space for physical attacks on sensor nodes. 18
2.4 Adversary presence parameter. 28
2.5 Adversary intervention parameter. 29
2.6 Adversary Model Lattice 30

3.1 Design space for access control (AC) in sensor networks. Entities that generate the authenticator for the query, as well as the connections (sometimes, multihop) between the network and the user, are depicted with thick lines. 47

4.1 μTESLA example. In this example the key is revealed two periods after the message was sent. 58

5.1 Illustration of the *interleaved hop-by-hop scheme* for $t = 1$. The full arrows represent sent messages whereas the dashed arrows indicate for which nodes MACs are generated. 70
5.2 Discovery of additional nodes for the node B. 74
5.3 Structure of an authenticator. 75
5.4 Impact of the compromised path $(2, 4, 10)$ on the network, $t = 2$. 79
5.5 Nodes 1 and 2 are captured, $t = 2$. The faked query can reach non-compromised paths $(6, 8, 11)$, $(7, 9, 12)$ and $(7, 10, 12)$ if the adversary tampers with the authenticators generated by non-compromised nodes on the path $(3, 4, 5)$. 80
5.6 Diamond grid topology. 82

5.7 Area for which the authentication information is included in a message, the base station is located in the middle of the grid. 83
5.8 Example for the redundant calculation of MACs in the basic scheme. Here $t = 1$, node A generates a MAC for C, and C generates a MAC for A. 83
5.9 Spanning tree of a network with the base station as root. . . 84
5.10 Example of a spanning tree for the diamond grid. 88
5.11 The number of bytes in the authenticator for the basic vs. tree-based interleaved authenticated querying according to Formulas 5.1 and 5.4 depending on the maximal number of captured nodes t for the diamond grid topology (Figure 5.6). 91
5.12 Transformation of the original spanning tree into a tree with another root node. 94

6.1 Query propagation in a branching-process-like network (to the left), and in a wireless sensor network. 104
6.2 Necessary authenticator size for node density $d = 7$, depending on the ratio of the key ring size k to the size of the key pool l, and the number of compromised nodes $\tilde{n}$. 106
6.3 Minimum authenticator size m_{min} for node density 7 depending on the number of compromised nodes. 108
6.4 Necessary authenticator size, depending on node density, key ring size, and the number of compromised nodes. The size of key pool is $l = 10000$. 109
6.5 Number of nodes reached by the fake query depending on authenticator size, with node density 15 and key ring size 75. 110
6.6 Number of nodes reached by the fake query depending on authenticator size, with node density 7 and key ring size 150. 111
6.7 Probability of forwarding the fake query for the authenticator size $m \in \{200, 400\}$ and the number of compromised nodes $\tilde{n} \in \{20, 50\}$. 121
6.8 Number of nodes that accepted the fake query for the authenticator size 200 and key ring sizes 4 and 8. 123

List of Tables

3.1 Design space for realizing access control (AC) in sensor networks. "BS: yes" means that the base station must be accessed before the query processing can be started by the network. "Multihop: yes" means that multihop communication is needed before the query processing can be started by the network. 46

6.1 Variables used in the analysis of `bAQF` 102

Chapter 1

Introduction

1.1 Motivation

Wireless sensor networks (WSN) consist of a large number of sensor nodes which are small low-cost, low-performance, battery-powered wireless computing devices equipped with various sensors. The sensor nodes jointly collect and store environmental data such as temperature, humidity, light conditions, seismic activities, images of the environment. This data can be used to detect certain events and to trigger activities in such applications as habitat monitoring [119], wildfire detection [52], precision agriculture [34], structural health monitoring [148], military perimeter protection [8], medical emergency response [99] and many others. Comprehensive overviews on WSN applications can be found in Zhao and Guibas [154], Karl and Willig [83].

In principle, a sensor network can be regarded as a large and massively distributed database that continuously acquires and stores new data and makes it accessible to users in some meaningful way. For example, a user might wish to know in which areas covered by the sensor network the temperature is above some threshold values of θ degrees.

This thesis considers the problem of protecting sensor network data from illegitimate access. In the following, we explain why this data needs protection in the first place (Section 1.1.1), why the traditional security mechanisms cannot be used in sensor networks (Section 1.1.2), and finally, why determining against which attacks the data should be protected is in itself a new challenge in sensor networks (Section 1.1.3).

1.1.1 The Need for Access Control

If the data collected within a sensor network is valuable or should be kept confidential, then security measures should protect the access to this data.

There are two main reasons why this is necessary.

The first reason is concerned with the value of data. With the increasing ubiquity of WSNs, environmental data will be available on demand almost everywhere in our environment from a surrounding WSN. Of course, accessing this data will in general not be for free since deployment of WSNs induces some costs. This means that the deployment agencies of these services will make them available only to certain people, usually people who pay for receiving the service. In this case, a WSN must be able to distinguish legitimate users from illegitimate ones. The incentives to gain illegitimate access to these valuable data can be expected to be quite high.

Moreover, if the sensor network data is used for gaining critical information, such as intrusion detection for building or area security, the injection of false data into the network (for example, in order to mask an intrusion) becomes a likely event.

The second reason concerns the sensitivity of data. For example, if the sensor network is used to monitor a building, then the temperature and humidity data from different rooms would allow the burglars to infer the presence of humans in the different parts of the building.

1.1.2 The Need for Novel Security Solutions

Securing sensor networks is a difficult problem due to a number of reasons [117, 126]. The extreme *resource scarceness* of the sensor nodes, the *large network size*, and the possibility of *node capture attacks* are usually mentioned as the most severe difficulties.

A typical sensor node contains an 8 to 16 bit microcontroller, with the amount of RAM varying between 2 kB and 10 kB and flash instruction memory ranging from 48 kB to 128 kB. The speed of radio communications is in the order of 100 $^{\text{kbit}}/_{\text{s}}$. The sensor networks are supposed to operate for months and even years using a few small (e.g., AA) batteries. This can only be possible if all algorithms running on the sensor nodes are designed with energy saving in mind.

This means that computation-intensive tasks that are typical for public-key cryptography are very expensive in terms of energy, memory and time consumption, and should be used sparingly, if at all. This is unfortunate since public-key cryptography is the main instrument to achieve access con-

trol in traditional networks.

For example, the most efficient so far implementation of elliptic curve cryptography (ECC) on sensor nodes [70] still needs 0.5 seconds for the basic ECC operation of point multiplication. For comparison, basic symmetric-key operations can be performed in a fraction of a millisecond [84].

Additionally, especially the radio consumes a lot of energy. Sending one bit can be as expensive as executing 1000 instructions [74]. A sensor node which operates with its radio always on (for example, waiting for messages), may exhaust its battery after only one week, even if it does not compute anything. Thus, security measures in sensor networks should not demand much communication.

Restricted communication puts constraints on the use of symmetric-key cryptography in large sensor networks. Although symmetric-key algorithms are lightweight enough to run on sensor nodes, the management of shared keys in a large network requires much communication.

Furthermore, management of shared keys becomes especially difficult in presence of node capture attacks. Node capture means gaining full control over a sensor node through a physical attack, e.g., by opening the node's cover and reading out its memory and changing its program. This attack is very likely to go unnoticed in a large sensor network where the physical access to the nodes is not restricted. Thus, security solutions for sensor networks should be resilient to node capture and therefore cannot completely rely on secrets which are stored inside single nodes.

The above features of sensor networks make traditional methods for achieving security in general, and access control in particular, inapplicable and call for novel security paradigms and solutions.

1.1.3 The Need for Realistic Adversary Models

When designing a secure system, a fundamental question should be answered first: *what* should be protected *against which threats*? The question of *how* to protect cannot be answered otherwise.

The adversaries in sensor networks differ from the adversaries in traditional computer security with respect to the amount of physical access they can get to the nodes. In traditional computer security the adversary has only remote access to the nodes. In ad hoc networks, although the adversary may gain physical access to a restricted number of devices (e.g., by stealing them), most of the devices are supposed to be under the control of legitimate users.

In sensor networks, on the other hand, direct physical access to arbitrary

nodes may become possible, if the network operates unattended. This means that the adversary models used in traditional computer security do not fit the realities of sensor networks.

Numerous papers on different aspects of security in sensor networks have been published in the last 10 years. However, most of these papers have been astonishingly lax about the description of their adversaries. Although the adversary is usually mentioned and some of its properties are described, the descriptions are often ambiguous, such that comparison of different security solutions with the goal of choosing the most appropriate one becomes very difficult.

This may be due to the fact that the problem of security in sensor networks is relatively new, and the possible attacks are not well understood. Moreover, only very few real-world deployments are known so far, and to our knowledge, none of them has ever been attacked. On the other hand, adversary types in traditional security are very well known, not at least because of numerous real attacks on real computer systems. Moreover, traditional security has been developing for many decades, giving the researches many opportunities to define and refine adversary models. Thus, before designing security solutions for sensor networks, possible attacks should be analyzed and classified.

For example, how realistic are the above mentioned node capture attacks? Which resources do they require? Most current security mechanisms for WSNs take node capture into account. It is usually assumed that node capture is "easy". Thus, some security mechanisms are verified with respect to being able to resist capture of 100 and more sensor nodes out of 10,000 [37, 79]. However, to the best of our knowledge, nobody ever tried to determine the actual cost and effort needed to attack currently available sensor nodes.

1.2 Contributions

1.2.1 Real-World Physical Attacks on Sensor Networks

Many sensor network applications demand that the sensor nodes be placed in unattended environments, such as forests, fields, large buildings, where an attacker has the opportunity to gain physical access to several nodes. This type of physical attacks distinguish sensor networks from all other types of networks, where the nodes are usually associated with some human (the user, or the network operator) who takes care of restricting the access to the device.

This thesis presents, to our knowledge, the first systematical study of physical attacks on sensor nodes. We developed a design space for physical attacks, experimentally determined the effort needed for various kinds of attacks and proposed possible countermeasures. This systematical investigation also proved very valuable for developing and refining the adversary models for sensor networks.

1.2.2 Adversary Models for Sensor Networks

Is it possible to define realistic adversary models for sensor networks, considering the young age of the field and the absence of attacks on real systems? This thesis provides a set of *generic* adversary models for sensor networks. Our flexible and extendable framework comprises a broad range of possible adversaries which are structured into several orthogonal dimensions. Using our approach, the system designers can select the most appropriate adversary models for their sensor networks. The adversaries are placed into a partially ordered lattice. This is especially useful as the partial order gives insights into what can happen in case the adversary is stronger (or weaker) than the anticipated one.

Our adversary model is to our knowledge the first attempt to systematically investigate possible adversaries in sensor networks. It can be easily extended to new classes of adversaries, and makes security solutions for sensor networks comparable to each other.

1.2.3 Access Control to Sensor Network Data

Resource constraints and new types of attacks in sensor networks make security in this domain particularly challenging. As is well known in the world of fault-tolerant computing and service replication, withstanding capture of even a small amount of nodes in a network requires a severe amount of computation and communication [73, 121].

In this thesis, we consider the problem of access control in presence of node capture attacks. Firstly, we develop a general model for access control to the data in sensor networks. According to our model, access control comprises two subproblems: user access control and authenticated querying. We then consider solutions to these problems in the face of node capture attacks.

In particular, authenticated querying is a variant of a very prominent security problem called *broadcast authentication*. Numerous solutions to broadcast authentication in sensor networks have emerged in the last years.

Luk et al. [100] identify the most desirable properties for broadcast authentication in sensor networks. None of currently published protocols provide all these properties. In search of solutions to authenticated querying, we come up with two novel broadcast authentication protocols.

We firstly investigate the concept of sensor networks which are able to withstand capture of at most t sensor nodes. We call such networks *t-robust*. We consider security primitives for t-robust networks and develop corresponding protocols, especially a t-robust broadcast authentication protocol. The developed protocol is secure an efficient for small values of t.

On the other hand, we show that the resource requirements for t-robust sensor networks grow very fast with t, such that values of $t > 10$ are prohibitive. Moreover, according to the design goal, t-robust protocols provide secure solutions only as long as no more than t nodes are captured, and break completely in case of the capture of $t + 1$ nodes. This feature is undesirable, as it is difficult to determine the appropriate t. If the adversary was able to capture $t = 5$ nodes, the effort of capturing a sixth node seems to be marginal.

In the face of these difficulties, the concept of *gracefully degrading* and *probabilistic* security seems to be more appropriate for sensor networks. This concept means that the security properties of the protocol are satisfied with certain large probability and degrade with the number of captured nodes.

Although probabilistic security has been previously considered in sensor networks for key management [37, 55] and for data delivery [151], our *authenticated query flooding (AQF)* protocol is the first one to provide probabilistic, gracefully degrading security for access control in sensor networks. AQF is the first protocol to provide *all* desirable broadcast authentication properties.

1.3 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2 we discuss security goals in sensor networks, and present our experiments with physical attacks on sensor nodes. Based on this experience, we then present a flexible and extendable generic adversary model, and discuss possible protection mechanisms for different adversary classes.

In Chapter 3 we consider the problem of access control to sensor network data, subdivide it into two subproblems: user access control and authenticated querying, and discuss the design space for possible solutions in presence of different adversary classes.

Chapter 5 considers the concept of t-robust sensor networks. We develop and analyze protocols for t-robust access control, and especially for t-robust authenticated querying.

In Chapter 6 we develop a solution for probabilistic authenticated querying. The security of this solution degrades gracefully with the number of captured nodes. We conclude that gracefully degrading probabilistic security is more appropriate for sensor networks than t-robustness.

Finally, Chapter 7 summarizes this thesis and gives directions for future work.

Chapter 2

Security Threats and Adversary Models

2.1 Introduction

The following quotation by Gligor [65] gives probably the best (and shortest) motivation for the work presented in this chapter: *A system without an adversary definition cannot be insecure. It can only be astonishing.*

This statement emphasizes the impossibility of designing a secure system without answering a fundamental question first: *what* should be protected *against which threats*?

Considering the Internet as an example, it is extremely difficult to add security to systems which were originally designed without security in mind. The goal of security is to "protect right things in a right way" [3]. Thus, careful analysis is needed concerning which things to protect against which threats and how to protect them. Of course, this analysis is only possible in context of a particular class of applications. However, it makes much sense to provide a set of abstract security requirements and a set of generic attacker models, i.e., a *framework for security analysis in wireless sensor networks*, which can be refined for particular applications.

In this chapter, we present such a framework. It provides concepts to clarify two important aspects of the security analysis in wireless sensor networks:

1. What should be protected? Here we offer a set of generic classes of requirements which can be used to structure and refine a set of concrete security requirements. We highlight the main differences between security requirements in classical systems and security requirements in

wireless sensor networks.

2. Against what are we protecting the system? Here we offer a set of generic attacker models which can be used to choose and refine particular attacker models for individual systems.

Overall, attacker models in conjunction with security requirements determine the means to achieve security.

In practice it is very important to formulate *realistic* security requirements and *realistic* attacker models. Such choices guarantee that precious resources of wireless sensor networks are invested efficiently. It is therefore useful to evaluate the practicality of certain attacker models. One metric to measure practicality is to evaluate the *effort* an attacker has to invest to perform certain attacks. We contribute to this area by reporting on a number of experiments in which we attacked *real* sensor node hardware.

This chapter is structured as follows: We first present related work in Section 2.2. We give an overview of security goals in sensor networks, i.e., we approach the question "what to protect" (Section 2.3). We then report on experiments in attacking wireless sensor networks in Section 2.4. Building on these experiences we develop a generic set of attacker models in Section 2.5, i.e., we approach the question "against which threats to protect". Finally, we discuss protection mechanisms in Section 2.6, i.e., we approach the question "how to protect". We outline open problems and summarize in Section 2.7.

2.2 Related Work

2.2.1 Security Goals

The most prominent taxonomy of security goals, which is also used in this thesis, is the CIA taxonomy [82, 141] which defines security as the combination of three attributes: confidentiality, integrity and availability.

There have been many discussions on whether or not the CIA spectrum really covers all relevant security properties. For example, *accountability* is sometimes treated as a property aside from CIA [66], whereas others [2, 122] argue that it falls into the domain of integrity.

2.2.2 Physical Attacks

Physical attacks on embedded systems, that is, on microcontrollers and smart cards, has been intensively studied before [3, 5, 127, 128]. Skorobogatov describes in depth tampering attacks on microcontrollers, and classifies

them in the three categories of *invasive*, *semi-invasive*, and *non-invasive* attacks [127]. Invasive attacks are those which require access to a chip's internals, and they typically need expensive equipment used in semiconductor manufacturing and testing, as well as a preparation of the chip before the attack can begin. Semi-invasive attacks require much cheaper equipment and less time than the invasive attacks, while non-invasive attacks are the easiest.

All these attacks, including the so-called low-cost attacks, if applied to sensor nodes, would require that they be removed from the deployment area and taken to a laboratory. Even if in some cases, the laboratory could be moved into the deployment area in a vehicle, all attacks would require at least disruption of the regular node operation. Most of the invasive and many of the semi-invasive attacks also require disassembly or physical destruction of the sensor nodes.

Skorobogatov also lists several possible classification schemes, including U. S. government standards, both for attackers, according to their capabilities, and for defenses, according to their abilities to resist attacks from a certain adversary class.

The existing literature on physical attacks usually assumes that an attacker can gain unsupervised access to the system to be attacked for an extended period of time. This is a sensible assumption for systems such as pay-per-view TV cards, pre-paid electricity tokens, or GSM SIM cards. Attacks which may take days or even weeks to complete present a real threat to the security of these systems.

In wireless sensor networks, however, regular communication with neighboring nodes is usually part of normal network operation. Continuous absence of a node can therefore be considered an unusual condition that can be noticed by its neighbors. This makes time a very important factor in evaluating attacks against sensor nodes, as the system might be able to detect such attacks while they are in progress and respond to them in real-time. One of our aims has been to determine the exact amount of time needed to carry out various attacks. Based on these figures, the frequency with which neighbors should be checked can be adapted to the desired level of security and the anticipated threat model.

Finally, the focus of previous work has been mostly on attacking the components themselves as opposed to the entire products. Attacks on the circuit-board level have been deliberately excluded from many works, although they are recognized to be security-relevant in some cases. We did not exclude such attacks from our investigation since our focus was on the security of the entire node and not only of its individual components.

Independently and concurrently to our work, Hartung et al. [72] also investigated sensor node compromise. They showed how to extract data from the MICA2 nodes using debugging tools, and suggested countermeasures which are similar to our recommendations. However, our work is more systematical, as we develop and investigate a design space for physical attacks on the sensor node hardware. We also looked at multiple types of sensor nodes, identified some additional attack types, and suggested some additional countermeasures.

Probably most worrying type of physical attacks on sensor nodes would be non-invasive side-channel attacks, such as timing and power analysis [87]. To the best of our knowledge, these attacks have not been applied to sensor networks yet and constitute an interesting direction for future research.

2.2.3 Adversary Models

Adversary models should be defined (and usually are defined) in every paper on WSN security. However, these definitions are informal, and, more importantly, ambiguous. This makes it impossible to compare different solutions to each other, and to choose a solution which is more appropriate for a particular situation. To our knowledge, there is no systematic treatment of adversary models in WSNs in the literature. On the other hand, adversary models is a well-established topic in the area of traditional computer security and safety.

One of the first well-established formal models in computer security is the Dolev-Yao model [49]. This model considers an outside adversary who has the full control over the communication channel. The Dolev-Yao adversary can eavesdrop, delay, delete, inject, replay and modify messages, but it cannot compromise nodes.

Node compromise is considered in the area of *secure multi-party computation (SMC)* [104]. Here, the nodes need to compute a function cooperatively, and the outcome of the computation should be kept secret from the adversary, even if it compromised some nodes, usually t out of n, although the *general adversary structures* [57] also consider other subsets of compromised nodes.

The SMC-adversaries can be classified according to multiple (usually, binary) parameters that can be combined arbitrarily. A *passive* adversary knows the program and the memory of the compromised nodes, but does not influence their protocol behavior, whereas an *active* adversary can make the compromised nodes execute arbitrary programs. A *computational* adversary cannot break public-key cryptography, whereas an *information-theoretical*

adversary possesses unlimited computational resources. If the communication between the nodes is *synchronous*, then the adversary cannot delay or delete messages of the honest participants, whereas in *asynchronous* communication model the adversary has full control over the communication channel. An *adaptive* adversary can compromise nodes at any time during the protocol run, whereas the *static* adversary has to decide beforehand which nodes to compromise.

The first to our knowledge attacker model for ad hoc networks [77] defines the $n - m$ attacker model, where the attacker is a legitimate owner of m nodes, and additionally compromises n nodes belonging to other users. This attacker model is not quite appropriate for sensor networks considered in this work, as we assume that all legitimate nodes have the same owner (e.g., the company that deploys the WSN).

In the area of fault-tolerance no intelligent adversaries are usually considered, and the adversary models are called *failure models*. Accordingly, the compromised, or faulty, nodes are susceptible to the corresponding failures. *Crash* failure means that the faulty nodes can crash at some point in the protocol execution. In the *Byzantine* failure model, the faulty nodes may exhibit arbitrary behavior. This means that in the worst case, the behavior of Byzantine nodes may be the same as that of an intelligent adversary. However, some care is required in using the Byzantine fault model in security [62]. In the *omission* model, the nodes can crash or commit send and receive omissions. That is, the faulty nodes may lose outcoming or incoming messages without noticing this. In the *crash-recovery* model, the crashed nodes can recover from the crash to a predefined state.

In our adversary model for sensor networks we try to merge both models for security and for fault-tolerance, taking into account the specific features of sensor networks.

2.3 Security Goals in Sensor Networks

A sensor network can be considered as a highly distributed database. Security goals for distributed databases are very well studied: The data should be accessible only to authorized users (Confidentiality), the data should be genuine (Integrity), and the data should be always available on the request of an authorized user (Availability). All these requirements also apply to sensor networks and their users. Here, the distributed database, as well as the sensor network, are considered as a single entity from the user's point of view. Therefore, we call these security issues *outside security*. To out-

side security belong, e.g., query processing [76, 120, 142], user access control (considered in this thesis) and large-scale anti-jamming services [147].

The internal organization of a distributed database and of a sensor network are quite different. Outside security, as well as all other types of interactions between the user and the corresponding system, is based on the interactions between the internal system components (servers or sensor nodes, respectively). We call security issues for such interactions *inside security*. In sensor networks, inside security realizes robust, confidential and authenticated communication between individual nodes [84, 139]. This also includes authenticated querying (considered in this thesis), in-network processing [47, 156], data aggregation [29, 159], routing [44, 85] and in-network data storage [17, 64].

Aside from necessitating the distinction between inside and outside security, sensor networks differ from conventional distributed databases in other obvious ways: A distributed database consists of a small number of powerful servers, which are well protected from physical capture and from network attacks. The servers use resource-demanding data replication and cryptographic protocols for inside and outside security. In contrast, a sensor network consists of a large number of resource-constrained, physically unprotected sensor nodes which operate unattended. Therefore, security measures for distributed databases cannot be directly applied to sensor networks. So even if a sensor network can be considered as a distributed system (e.g., as an ad hoc network), sensor networks have some additional constraints which make security mechanisms for distributed systems inapplicable. Apart from the obvious resource constraints, single sensor nodes are relatively unimportant for the properties of the whole system – at least, if the inherent redundancy of sensor networks is utilized in their design.

To summarize, security goals in sensor networks are similar to security goals in distributed databases (outside security) and distributed systems (inside security). So these can be taken as an orientation. While requirements are similar, many standard mechanisms to *implement* security (e.g., public key infrastructures or agreement protocols) are not applicable because they require too many resources or do not scale to hundreds or thousands of nodes. This is the dilemma of sensor networks and forces security mechanisms in wireless sensor networks to spend the "right" amount of effort in the "right" places by exploiting the natural features of sensor networks: inherent redundancy and broadcast communication.

In the remainder of this chapter, we will take a first step towards developing realistic security mechanisms. We evaluate real attacks in practice and from this evaluation derive a fine-grained set of abstract attacker as-

sumptions.

2.4 Real-World Physical Attacks on Sensor Networks

In this section we take the viewpoint of an adversary who wishes to violate one of the security requirements of a WSN which were described in Section 2.3. From this viewpoint, a WSN is a very interesting target because if offers a large attack surface and an interesting playground for creative attack ideas.

Of course, the many possibilities to attack WSNs include all the techniques known from classic system security. An adversary can eavesdrop on the communication, perform traffic analysis of the observed network behavior, replay old messages or inject false messages into the network. Possible are also other types of attacks that aim at violating Availability (denial-of-service attacks) like jamming the wireless channel. In WSNs these attacks can be particularly important as they can cause rapid battery drainage and effectively disable individual sensor nodes or entire parts of a WSN.

While there are many techniques known from other areas of security, the ability of an attacker to access (and eventually change) the internal state of a sensor node seems particularly characteristic for sensor networks. This type of attack is called *node capture* in the literature [117]. Depending on the WSN architecture, node capture attacks can have significant impact. For example, in the TinySec mechanism [84] that enables secure and authenticated communication between the sensor nodes by means of a network-wide shared master key, capture of a single sensor node suffices to give the adversary unrestricted access to the WSN. Furthermore, most existing routing schemes for WSNs can be substantially influenced even through capture of a minute portion of the network [85].

In this section, we determine the actual cost and effort needed to attack currently available sensor nodes. We especially concentrate on *physical attacks* which require direct physical access to the sensor node hardware. As sensor nodes operate unattended and cannot be made tamper proof because they should be as cheap as possible, this scenario is more likely than in most other computing environments. Physical attacks of some form are usually considered a prerequisite to perform node capture.

Another possibility to gain access to the internal state of a sensor node is to exploit some bugs in software running on the sensor nodes or on the base stations. These attacks directly correspond to well-known techniques from

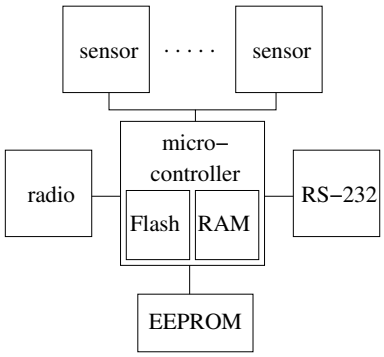


Figure 2.1: General schematic view of sensor node hardware.

classical software security. If a software vulnerability is identified by the attacker, an attack can be easily automated and can be mounted on a very large number of nodes in a very short amount of time. Such attacks are relatively well-understood in software security [75]: Possible countermeasures include a heterogenous network design and standard methods from software engineering [68, 105, 116, 138]. Software attacks on WSNs have only just recently came into research focus [59, 69]. Although they constitute a very interesting research direction, they are out of scope of this thesis.

In the following, we first give some background on physical attacks on sensor node hardware. Then we report on the effort needed to attack some current sensor nodes. Where appropriate we also discuss countermeasures that increase the effort to mount a successful attack. Overall, this section gives insight into *practical* attacks on WSNs which motivate the abstract attacker models which follow in Section 2.5.

2.4.1 Background on Physical Attacks on Sensor Nodes

2.4.1.1 Current Sensor Node Hardware

Currently available sensor nodes typically consist of embedded hardware with low power consumption, and low computing power. A typical sensor node contains some sensors (light, temperature, acceleration etc.), a radio chipset for wireless communication, an EEPROM chip for logging sensor data, a node-to-host communication interface (typically a serial port), and a microcontroller which contains some amount of flash memory for program storage and RAM for program execution. Power is provided by batteries.

Typical choices for the microcontroller are the 8 bit Atmel ATmega 128

Figure 2.2: Current sensor node hardware: Mica 2 by Crossbow, Berkeley [103]; Tmote sky by moteiv, Berkeley [111]; and Embedded Sensor Board by ScatterWeb, Berlin [27]

or the 16 bit Texas Instruments MSP430 family, with the amount of RAM varying between 2 kB and 10 kB and flash memory ranging from 48 kB to 128 kB. External EEPROM memory can be as small as 8 kB or as large as 1 MB. The speed of radio communications is in the order of 100 kbit/s.

The most interesting part for an attacker will be the microcontroller, as control over this component means complete control over the operation of the node. However, the other parts might be of interest as well in certain attack scenario.

Figure 2.1 shows a general schematic view of the hardware of current sensor nodes, while Figure 2.2 shows photographs of some concrete models available today. The Crossbow Mica2 nodes [42, 43] (Fig. 2.2, left) use the 8 bit Atmel ATmega 128 microcontroller [7] with 4 kB RAM and 128 kB integrated flash memory, the Atmel AT45DB041B 4 Mbit flash memory chip [6], and the Chipcon CC1000 radio communications chipset [40] with a maximum data rate of 76.8 kbit/s. Programming is done via the Atmel's serial programming interface by placing the node in a special interface board and connecting it to an RS-232 serial port on the host.

The Telos motes [110, 111] (Fig. 2.2, center) by Moteiv utilize the Texas Instruments MSP430 F1611 microcontroller [134, 135], providing 10 kB of RAM and 48 kB flash memory. The EEPROM used is the 8 Mbit ST Microelectronics M25P80 [130], and the radio chipset is the Chipcon CC2420 [41], whose maximum data rate is 250 kbit/s. Programming is performed by connecting to the USB interface and writing memory with the help of the MSP430 bootloader [132]. A JTAG interface is available as an alternative programming method and can also be used for debugging.

The Embedded Sensor Boards [27] (Fig. 2.2, right) from ScatterWeb GmbH are built around the Texas Instruments MSP430 F149 microcontroller [133, 135] featuring 2 kB of RAM and 60 kB flash memory, the Mi-

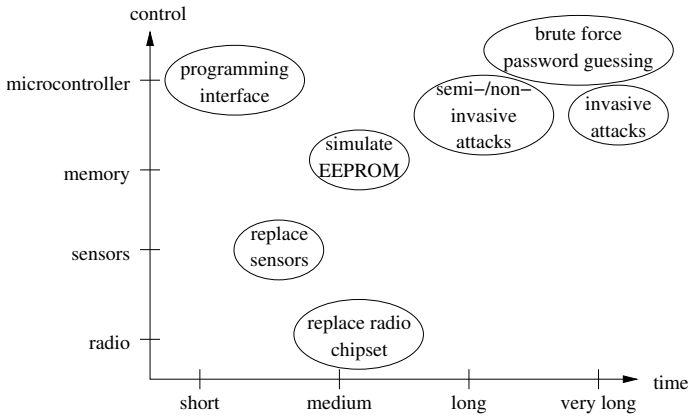


Figure 2.3: Design space for physical attacks on sensor nodes.

crochip 24LC64 [108] 64 kbit EEPROM, and the RFM TR1001 radio chipset [109] with a maximum data rate of 19.2 $^{\text{kbit}}/_{\text{s}}$. Programming is done either through a JTAG interface or over-the-air using a gateway.

2.4.1.2 Possibilities for Physical Attacks

Concrete physical attacks on sensor nodes can be classified in several ways. Our classification takes our previous considerations into account that sensor nodes are more or less in permanent contact with each other. This means that long interruptions of regular operation can be noticed and acted upon. Therefore, attacks which result in a long interruption (e. g. because the node has to be physically removed from the network and taken to a distant laboratory) are not as dangerous as those which can be carried out *in the field*. Therefore, in the following we concentrate on this type of attacks as well as on possible countermeasures to be employed by a sensor network.

The two main categories that we use for classifying physical attacks are (1) the degree of control over the sensor node the attacker gains; and (2) the time span during which regular operation of a node is interrupted [11]. Figure 2.3 illustrates this design space and classifies example attacks from the forthcoming Section 2.4.2 according to its criteria.

2.4.2 Examples of In-the-Field Attacks and Countermeasures

As explained above, we especially consider attacks which can be mounted without noticeable interruption of the regular sensor node operation. We now discuss some attacks in detail.

2.4.2.1 Attacks via JTAG

The IEEE 1149.1 JTAG standard is designed to assist electronics engineers in testing their equipment during the development phase. Among other things, it can be used in current equipment for on-chip debugging, including single-stepping through code, and for reading and writing memory.

A JTAG Test Access Port (TAP) is present on both the Atmel and Texas Instruments microcontrollers used on the sensor nodes described above. All sensor nodes examined by us have a JTAG connector on their circuit board allowing easy access to the microcontroller's TAP. While the capabilities offered by JTAG are convenient for the application developer, it is clear that an attacker must not be provided with the same possibilities. Therefore it is necessary to disable access to the microcontroller's internals via JTAG before fielding the finished product.

The MSP430 has a security fuse which can be irreversibly blown (as described in the data sheet) to disable the entire JTAG test circuitry in the microcontroller. Further access to the MSP430's memory is then only possible by using the Bootstrap Loader. The ATmega128 requires the programmer to set the appropriate fuses and lock bits, which effectively disable all memory access via JTAG or any other interface from the outside.

If JTAG access is left enabled, an attacker equipped with an appropriate adapter cable and a portable computer is capable of taking complete control over the sensor node. Even if there is no JTAG connector provided on the circuit board, attackers can still get access to the JTAG ports by directly connecting to the right pins on the microcontroller which can be looked up in the datasheet. Typical data rates for JTAG access are 1–2 kB/s, so reading or writing 64 kB of data takes between 30 and 70 s. However, there are specialized programming devices on the market which can attain much higher data rates.

2.4.2.2 Attacks via the Bootstrap Loader

On the Telos nodes, the canonical way of programming the microcontroller is by talking to the Texas Instruments specific bootstrap loader (BSL) through the USB interface. The bootstrap loader [132] is a piece of software contained in the ROM of the MSP430 series of microcontrollers that enables reading and writing the microcontroller's memory independently of both the JTAG access and the program currently stored on the microcontroller.

The BSL requires the user to transmit a password before carrying out any interesting operation. Without this password, the allowed operations are

essentially “transmit password” and “mass erase”, i.e. erasing all memory on the microcontroller.

The BSL password has a size of $16 \cdot 16\,\text{bit}$ and consists of the flash memory content at addresses 0xFFE0 to 0xFFFF. This means in particular that, immediately after a mass erase operation, the password consists of 32 bytes containing the value 0xFF. The memory area used for the password is the same that is used for the interrupt vector table, i.e. the BSL password is actually identical to the interrupt vector table. The interrupt vector table, however, is usually determined by the compiler and not by the user, although Texas Instruments documents describe the password as user-settable.

Finding out the password may be quite time-consuming for an attacker. However, such an investment of time may be justified if a network of nodes all having an identical password is to be attacked. Therefore, an evaluation of the possibility to guess the password follows.

Brute Force. As the password is composed of interrupt vectors, certain restrictions apply to the values of the individual bytes. This section examines the expected size of the key space and estimates the expected duration of a brute force attack on the password.

Initially, the key space has a size of $16 \cdot 16\,\text{bit} = 256\,\text{bit}$. Assuming a typical compiler (mspgcc 3.2 [112] was tested), the following restrictions apply:

- All code addresses must be aligned on a 16 bit word boundary, so the least significant bit of every interrupt vector is 0. This leaves us with a key space of $16 \cdot 15\,\text{bit} = 240\,\text{bit}$.
- The reset vector, which is one of the interrupt vectors, is fixed and points to the start of the flash memory, reducing the key space to $15 \cdot 15\,\text{bit} = 225\,\text{bit}$.
- Interrupt vectors which are not used by the program are initialized to the same fixed address, containing simply the instruction `reti` (return from interrupt). As a worst case assumption, even the most basic program will still use at least four interrupts, and therefore have a key space of at least $4 \cdot 15\,\text{bit} = 60\,\text{bit}$.
- Code is placed by the compiler in a contiguous area of memory starting at the lowest flash memory address. Under the assumption that the program is very small and uses only $2\,\text{kB} = 2^{11}\,\text{B}$ of memory, we are left with a key space of a mere $4 \cdot 10\,\text{bit} = 40\,\text{bit}$.

We conclude that the size of the key space for every BSL password is at least 40 bit.

A possible brute force attack can be performed by connecting a computer to the serial port (the USB port, in the case of Telos nodes) and consecutively trying passwords. This can be done by executing a modified version of the `msp430-bsl` [112] program that is normally used for communicating with the BSL.

The rate at which passwords can be guessed was measured to be approximately 12 passwords per second when the serial port was set to 9600 baud. However, the MSP430 F1611 used on the Telos nodes is capable of a line speed of 38,400 baud, and at this speed, approximately 31 passwords can be tried per second. Finally, the BSL program normally waits for an acknowledgment from the microcontroller after each message sent over the serial line. If this wait is not performed, the speed of password guessing rises to 83 passwords per second.

The maximum speed of password guessing in practice can therefore be assumed to be 2^7 passwords per second. This is quite close to the theoretical limit of $38,400\,{}^{\text{bit}}/_{\text{s}} \cdot (256\,{}^{\text{bit}}/_{\text{pw}})^{-1} = 150\,{}^{\text{pw}}/_{\text{s}}$.

Recalling that the key space has a size of at least 40 bit, we can now conclude that a brute force attack can be expected to succeed on the average after $2^{40-7-1}\,\text{s} = 2^{32}\,\text{s} \approx 128\,\text{a}$. As 128 years is well beyond the expected life time of current sensor nodes, a brute force attack can be assumed to be impractical.

Knowledge of the Program. One consequence of the fact that the password is equal to the interrupt vector table is that anyone in possession of an object file of the program stored on a sensor node also possesses the password. Worse, even someone who only has the source code of the program still can get the password if he has the same compiler as the developer, since he can use this compiler to produce an image from the source code identical to the one on the deployed nodes.

The secret key in the current TinySec implementation, for example, is contained in the image but does not influence the interrupt vector table. If TinySec were ported to Telos motes, the source code and the compiler used would be sufficient information for an attacker to extract the secret key material. The same holds for any kind of cryptographic mechanism where the key material does not influence the interrupt vectors.

Another way of exploiting the identity of the password and the interrupt vector table is to take one node away from the network and attack the

microcontroller on this node with classic invasive or semi-invasive methods. The absence of the node from the network will probably be noticed by the surrounding nodes and its key(s) will be revoked. However, once the attacker succeeds with his long-term attack and learns the BSL password of the one node, it is trivial for him to attack all of the other nodes in the field if they all have the same BSL password.

If an attacker knows the BSL password, reading or writing the whole flash memory takes only about 25 s. In order to avoid these forms of attack, *interrupt vector randomization* [11] can be used. This significanyly raises the bar for an attacker but is not yet part of standard software distributions.

2.4.2.3 Attacking the External Flash

Some applications might want to store valuable data on the external EEPROM. For example, the Deluge [78] implementation of network reprogramming in TinyOS stores program images received over the radio there. If these images contain secret key material, an attacker might be interested in reading or writing the external memory.

Probably the simplest form of attack is eavesdropping on the conductor wires connecting the external memory chip to the microcontroller. Using a suitable logic analyzer makes it easy for the attacker to read all data that are being transferred to and from the external EEPROM while she is listening. If a method were found to make the microcontroller read the entire external memory, the attacker would learn all memory contents. This kind of attack could be going on unnoticed for extended periods of time, as it does not influence normal operation of the sensor node.

A more sophisticated attack would connect a second microcontroller to the I/O pins of the flash chip. If the attacker is lucky, the mote microcontroller will not access the data bus while the attack is in progress, and the attack will be completely unnoticed. If the attacker is skillful, he can sever the direct connection between the mote microcontroller and the flash chip, and then connect the two to his own microcontroller. The attacker could then simulate the external memory to the mote, making everything appear as normal.

Of course, instead of using his own chip, the attacker could simply do a “mass erase” of the mote’s microcontroller and put his own program on it to read the external memory contents. This operation is even possible without knowledge of the BSL password. While this causes “destruction” of the node from the network’s point of view, in many scenarios this might not matter to the attacker.

The exact amount of time required for the attacks proposed above remains to be determined. It should be noted that some of the attacks outlined above require a high initial investment in terms of equipment and development effort. A possible countermeasure could be checking the presence of the external flash in regular intervals, putting a limit on the time the attacker is allowed to disconnect the microcontroller from the external flash.

2.4.2.4 Sensors

Sensor nodes rely on their sensors for information about the real world, so the ability to forge or suppress sensor data can be classified as an attack. For instance, a surveillance system might be tricked into thinking that the situation is normal while the attacker passes unnoticed through the area under surveillance.

Replacing sensors on the different types of nodes varies in difficulty between child's play and serious electrical engineering, mostly depending on the type of connection between the microcontroller circuit board and the sensors. A pluggable connection—as present on the Mica2 motes—requires an attacker to spend only a few moments of mechanical work. If, on the other hand, the sensors are integrated into the printed circuit board design, replacing them involves tampering with the conductor wires, cutting them, and soldering new connections. The amount of time required for this will vary with the skill of the attacker, but it can be assumed to be in the order of minutes.

2.4.2.5 Radio

Finally, the ability to control all radio communications of a node might be of interest to an attacker, e.g., in order to mount an attack using deliberate collisions on the medium access level. We are unaware of any work trying to evaluate the effort necessary to perform such an attack and compare its effort with other attacks that target resource exhaustion and denial-of-service.

2.4.3 Summary

To summarize, we can classify the above attacks into three categories depending on the effort necessary. We list these classes in order of increasing severity:

1. The class containing the "easy" attacks: Attacks in this class are able to influence sensor readings, and may be able to control the radio

function of the node, including the ability to read, modify, delete, and create radio messages without, however, having access to the program or the memory of the sensor node. These attacks are termed "easy" because they can be mounted quickly with standard and relatively cheap equipment.

2. The class containing the "medium" attacks: Attacks in this class allow to learn at least some of the contents of the memory of the node, either the RAM on the microcontroller, its internal flash memory, or the external flash memory. This may give the attacker, e.g., cryptographic keys of the node. These attacks are termed "medium" because they require non-standard laboratory equipment but allow to prepare this equipment elswehere, i.e., not in the sensor field.

3. The class containing the "hard" attacks: Using attacks in this class the adversary complete read/write access to the microcontroller. This gives the attacker the ability to analyze the program, learn secret key material, and change the program to his own needs. These attacks are termed "hard" because they require the adversary to deploy non-standard laboratory equipment in the field.

A different way to classify the above attacks is to look at the time during which the node cannot carry out its normal operation. Here we have the following classes:

1. Short attacks of less than five minutes. Attacks in this class mostly consist of creating plug-in connections and making a few data transfers over these.

2. Medium duration attacks of less than thirty minutes. Most attacks which take this amount of time require some mechanical work, for instance (de-) soldering.

3. Long attacks of less than a day. This might involve a non-invasive or semi-invasive attack on the microcontroller, e.g., a power glitch attack where the timing has to be exactly right to succeed, or erasing the security protection bits by UV light.

4. Very long attacks which take longer than a day. These are usually invasive attacks on the electronic components with associated high equipment cost.

In the following section we take these practical insights as the basis for devising a framework of adversary models that can be used for security analysis of WSNs.

2.5 Adversary Models

Adversary models should be determined with respect to applications. Who are adversaries and what goals do they have? A military sensor network has other security requirements than a network for habitat monitoring. The adversaries can be classified according to the following parameters: goals, intervention, presence, and available resources. We first give an overview over these parameters and then treat the paramaters *intervention* and *presence* in detail later.

2.5.1 Overview

2.5.1.1 Goals

When designing security mechanisms in practice, it helps to know the *goals* of the adversary as precisely as possible. Which of the three classical security requirements (Confidentiality, Integrity, Availability) does the adversary try to violate? If the data is valuable (i.e., legitimate users have to pay) or privacy relevant, the adversary would try to gain unauthorized access. If the data is critical (e.g., building or perimeter security), the adversary would try to modify data, such that the alarm is not raised in case of intrusion. Also a denial-of-service attack can successfully disable the network (violating Availability).

Identifying the goals of the adversary is probably the most difficult aspect of security analysis in practice. Therefore the three security requirements Confidentiality, Integrity, and Availability are often treated in a uniform manner (all of them are goals of equal importance).

2.5.1.2 Presence

The parameter *presence* basically identifies *where* the adversary acts in a wireless sensor network. The basic distinguishing factor is the number and location of the nodes which are in the range of his influence. We distinguish *local*, *distributed* and *global* adversaries.

2.5.1.3 Intervention

While the presence parameter identifies *where* the adversary can act, the *intervention* parameter identifies *what* the adversary can do in these places. We distinguish *eavesdrop*, *crashing*, *disturbing*, *limited passive*, *passive*, and *reprogramming* adversaries. These classes offer incremental steps of power: Briefly spoken, an eavesdropping adversary can only listen to communication, a crashing adversary can additionally destroy sensor nodes, a disturbing adversary can additionally upset sensors by manipulating their location or their readings, a limited passive adversary can additionally open a node and steal all its secrets by temporarily removing it from the network, a passive adversary can steal all secrets without displacing the node, and a reprogramming adversary can additionally take full control of a node and cause it to act in arbitrary ways.

2.5.1.4 Available Resources

There are several resource classes to consider: funding, equipment, expert knowledge, time. In the world of tamper resistance, the adversaries are divided into three classes: *clever outsiders*, *knowledgeable insiders* and *funded organizations* [3]. Commodity sensor networks are likely to be attacked by clever outsiders (who are probably just trying things out) and possibly by knowledgeable insiders, if a substantial gain from the attack can be expected. Available resources are interdependent with the parameters *intervention* and *presence*. However, the precise connection is not always clear. For example, a global adversary need not to be a funded organization. A hacker could subvert the entire sensor network by exploiting some software bug. Or, if a local adversary manages to capture a sensor node and read out its cryptographic keys, he can turn into a distributed or global adversary, depending on how many sensor nodes he is able to buy and deploy as clones of the captured node.

2.5.1.5 Outlook and Notation

In our view, the presence and intervention parameters are the most relevant ones for basic security analysis. This is why we now look at these two in detail. Before we present them, we need some formal notation.

We model a sensor network as a graph $G = (V, E)$ where the set V is the set of sensor nodes and the set $E \subseteq V \times V$ defines a neighborhood relation. The *number of all sensor nodes* $|V|$ is denoted N. Two sensor nodes v_1 and v_2 are *neighbors* if and only if they are in the neighborhood

relation, i.e., $(v_1, v_2) \in E$. A set of nodes $\mathcal{V}$ is *connected* if and only if for all $v_1, v_2 \in \mathcal{V}$ holds that either $(v_1, v_2) \in E$ or $(v_2, v_1) \in E$. Note that in our notation E is *not* necessarily reflexive and transitive because we wish to model sensor networks at a very low layer, i.e., without any routing or transport mechanisms.

2.5.2 Presence

We now discuss the different and increasingly severe levels of the presence parameter. These levels are defined in an incremental fashion, i.e., every level includes the behavior of the previous one. This implies a total order of parameter values defined by the subset relationship of behaviors. We begin our explanations by starting with the weakest level first.

- local adversary

  A *local adversary* can influence a small localized part of the network [24], for example he has one receiver which can only eavesdrop on several meters, or can manipulate only the sensor node which is the closest to him (for example, it is installed in his office).

  Formally, an adversary is local if his range of influence contains a small connected subset $S \subset V$ of all sensor nodes. The set S can also be given as a Boolean function $S : V \mapsto \{\text{true}, \text{false}\}$. Such a set is small if its size is several orders of magnitude smaller than the number of total nodes, i.e., $|S| \ll N$. This set can also change over time, but changes are rather slow.

  In general it is always possible to define S in a time-dependent manner, i.e., $S : V \times T \mapsto \{\text{true}, \text{false}\}$ where T is the domain of time values.

- distributed adversary

  A *distributed adversary* models either a mobile adversary (a car with receiver driving around) or managed to install his own sensor nodes in the sensor field and coordinates them [4].

  Formally, an adversary is distributed if its range of influence consists of multiple unconnected small subsets of nodes of the sensor network. As an example of such an adversary, consider the assumption that sensor nodes are attacked randomly following a uniform distribution [159]. In such a case, compromised nodes are distributed over the entire network, but not wide enough to actually see, hear or listen anything.

local → distributed → global

Figure 2.4: Adversary presence parameter.

- global adversary

  A *global adversary* is the most powerful level of presence an adversary can exhibit. Such an adversary can analyze the complete network, hence his area of influence consists of *all* sensor nodes.

We define the relation → to order attacker models in the direction of stronger (i.e., more severe) attacker behavior. Formally, model $A \rightarrow B$ if and only if all behaviors which are possible in A are also possible in B (behavior subsetting). The levels of ability ordered with respect to → are depicted in Figure 2.4.

2.5.3 Intervention

Now that presence has been investigated and an ordered set of presence abilities has been given, the same needs to be done for the intervention capabilities an adversary can be ascribed.

The intervention order is constituted of the following levels, starting from the weakest and proceeding towards the strongest. Every level contains the behaviors of all preceding levels.

- eavesdropping adversary

  An eavesdropping adversary can just listen to network traffic, do nothing else, in particular no jamming etc. It can then analyse this traffic either online or offline.

- crashing adversary

  A crashing adversary exhibits the simplest form of failure: The node simply stops to operate (execute steps of the local algorithm). Elsewhere this is also called *failstop adversary* [20]. A *crashing* adversary attacks sensors such that they completely break down. The sensors can be destroyed or drained of energy.

- disturbing adversary

  A *disturbing* adversary can try to partially disturb protocols, even if it does not have full control over any sensors. It can selectively jam the

eavesdrop → crash → disturbing → limited passive → passive → reprogramming

Figure 2.5: Adversary intervention parameter.

network, or fool some sensors into measuring fake data. For example, it can hold a lighter close to a temperature sensor [142] or re-arrange the topology of the sensor network by moving sensors around.

- limited passive adversary

  An adversary of this level can retrieve all information on a node, but in order to do so it needs to completely remove the node physically from the network. This means that the danger of being caught is higher.

- passive adversary

  A passive adversary can directly go and open up arbitrary sensor nodes to get the information currently stored in such a node, there is no need to leave the network to do so. Furthermore, such an adversary is assumed to be able to modify the data the node contains.

- reprogramming adversary

  A *reprogramming* adversary can run arbitrary programs on a sensor node. This can be achieved by exploiting some software bug, or by probing out cryptographic keys and other secret information and then cloning the node as described above. The problem of *node capture* is typical for a reprogramming adversary.

The order above defines the different abilities of intervention that can be ascribed to an adversary. These can be ordered according to the relation → as defined above (subsets on behaviors), see Figure 2.5.

2.5.4 Adversary Model Lattice

Having defined ordered levels for both presence and intervention, those two abilities are now combined to form the complete set of adversary models.

A *basic adversary model* $\mathcal{A}$ is a pair (Presence, Intervention), where the first component specifies the level of presence the adversary model assumes and analogously the second component states the level of intervention. A basic adversary model thus looks like $\mathcal{A}(local, disturbing)$, in case of an adversary with local presence and disturbing skills. The result is a

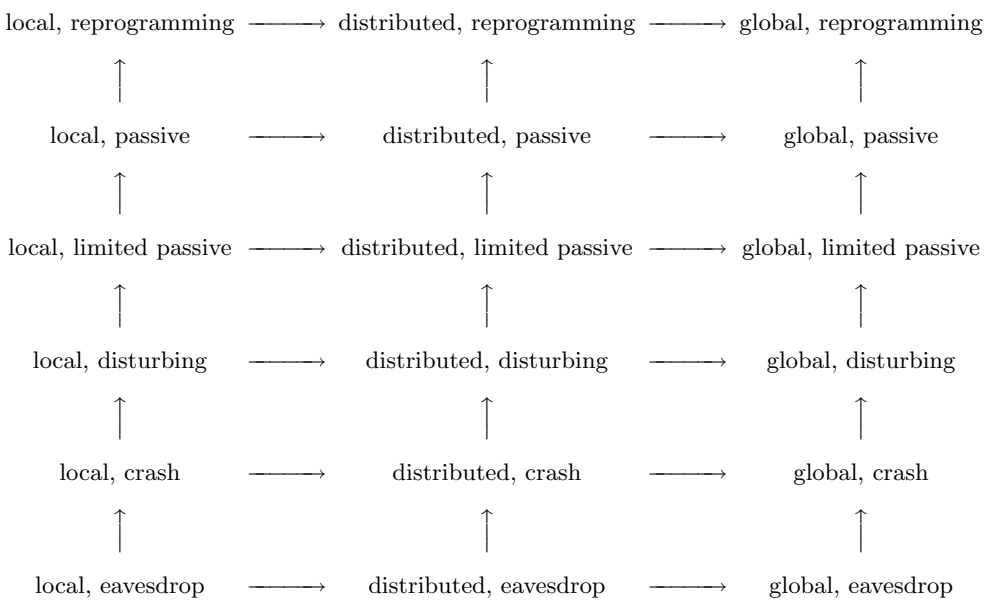


Figure 2.6: Adversary Model Lattice

partial order of basic adversary models. This partial order is depicted in Figure 2.6 as a lattice. It can be seen that the most powerful basic adversary is $\mathcal{A}(global, reprogramming)$ (in the upper right corner) and the weakest is $\mathcal{A}(local, eavesdrop)$ (in the bottom left corner). As it is a partial order, not all basic adversary models are comparable in the sense that one model is truly stronger than another. $\mathcal{A}(global, eavesdropping)$ and $\mathcal{A}(local, reprogramming)$ are such a pair, as neither of them is stronger than the other. However, some models can be put into relation. We take the relation $\rightarrow$ to be the combination of the behavior subsetting ordering defined for presence and intervention parameters above. The resulting partial order is depicted in Figure 2.6.

There are issues that only concern one ability, either presence or intervention, thus being ignorant about the other ability. For such cases, a $*$ symbol can be inserted for notational convenience into the appropriate component of the model to stand of all different levels possible. For instance, wanting to talk about an adversary that has local presence skills and neglecting completely intervention, the basic adversary model $\mathcal{A}(local, *)$ would be appropriate to refer to such an adversary.

The ordering of the levels of abilities yields the nice property that a statement ascribing a basic adversary of a low level (for example a $\mathcal{A}(local, *)$) to perform a malicious action, will automatically ascribe the same for adversaries of higher levels ($\mathcal{A}(distributed, *)$ and $\mathcal{A}(global, *)$). Similarly, it should be clear that an algorithm that can cope with a $\mathcal{A}(global, reprogramming)$ adversary will be able to sustain all other presented adversaries. However, as the lattice presents only a partial order, an algorithm coping with a $\mathcal{A}(local, limited\ passive)$ adversary will not necessary work as well with a $\mathcal{A}(global, eavesdrop)$ adversary.

A fully specified *adversary model* is a set of basic adversary models $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. This adversary definition makes it possible to define hybrid attacker assumptions. For example, an adversary can be global eavesdropping and local reprogramming at the same time and is then specified as follows:

$$\{\mathcal{A}(global, eavesdrop), \mathcal{A}(local, reprogramming)\}$$

2.5.5 Summary

We have presented an abstract framework for modeling attackers in WSNs. We claim that due to our experiments described in Section 2.4 these abstract classes model well the critical differences between the hardness of soultions.

2.6 Discussion of Protection Mechanisms

In Section 2.4 we systematically investigated physical attacks on current sensor node hardware, paying special attention to attacks which can be executed directly in the deployment area, without interruption of the regular node operation. We found out that most serious attacks, which result in full control over a sensor node (*node capture*), require absence of a node in the network for a substantial amount of time. We also found simple countermeasures for some of the most serious attacks.

Thus, in order to design a WSN secure against those node capture attacks described in this chapter, the following steps should be applied:

- take standard precautions for protecting microcontrollers from unauthorized access;
- choose a hardware platform appropriate for the desired security level, and keep up-to-date with new developments in embedded systems security;

- monitor sensor nodes for periods of long inactivity;
- allow for revocation of the authentication tokens of suspicious nodes.

Standard precautions for protecting microcontrollers from unauthorized access, such as disabling the JTAG interface, or protecting the bootstrap loader password, are an absolute prerequisite for a secure sensor network. We developed a method of protecting the bootstrap loader password by randomization of the interrupt vector table. This allows the developers to make source code of their products public without fearing that their WSN can be taken over by everybody who compiles the source code using the same compiler, thus obtaining the same interrupt vector table, and therefore, the same BSL password.

As security is a process, not a product, system designers should keep up-to-date with the developments in attacks on embedded systems. The security of important systems should be constantly re-evaluated to take new discoveries into account, as newly found attack methods on microcontrollers or previously unknown vulnerabilities might make a previously impossible low-cost attack in the field possible.

The level of security required from the application should also be kept in mind when choosing hardware. In some cases it might make sense to build additional protection, such as a secure housing, around a partially vulnerable microcontroller.

Finally, the removal of a sensor node from the deployment area can be noticed by its neighbors using, e. g., heartbeat messages or topology change notifications, as well as by the sensor node itself using, e. g., acceleration sensors. Appropriate measures can then be taken by the network as well as by the node itself. The network might consider a node that has been removed as "captured" and revoke its authorization tokens or initiate recovery when this node returns to the network [140]. The node itself might react to a suspected physical attack by erasing all confidential material stored on it.

Mechanisms should be developed that allow a sensor node which has been absent for too long from the network to be revoked by its neighbors. This is our future work. Note that depending on the WSN design, local revocation could be insufficient. For example, if an attacker removes a single sensor node from the network and successfully extracts the node's cryptographic keys, the attacker would be able to *clone* nodes, to populate the network with new sensor nodes which all use the cryptographic keys of the captured sensor node. Thus, a WSN should also be protected from node cloning.

In general, for passive adversaries, encryption techniques often suffice to ensure inside security. Symmetric key encryption techniques will be favored

over asymmetric ones because of the computational advantage and the comparatively small key sizes. The problems arise in the setup phase of the network where shared secrets need to be distributed either by the manufacturer at production time or by clever protocols at deployment time [4, 79].

For stronger adversaries, active attacks like impersonation and node capture must be taken into account. Ideally, sensor nodes should be made tamper proof to prevent node capture, e.g., by applying technology known from smart cards or secure processing environments. There, memory is shielded by special manufacturing techniques which makes it more difficult to physically access the stored information [3]. Similarly, sensor nodes could be built in a way that they loose all their data when they are physically tampered with by unauthorized entities.

For large sensor networks, cost considerations will demand that sensor nodes are not tamper proof. Therefore, node capture must be taken into account. A first step to protect a sensor network from node capture against a local or partially present adversary is to use locally distributed protocols that can withstand the capture of a certain fraction of nodes in the relevant parts of the network. We give examples of such protocols in Chapters 5 and 6 of this thesis.

2.7 Conclusions

We have described security goals, adversary models and protection mechanisms which are relevant and specific for sensor networks. There are a lot of interesting problems and open questions in this area:

- *Realistic* adversary models should be derived with respect to existing and future applications. Here, experiences with GSM and WLAN security (and security failures) can be used as a guideline, but every application needs to define its own adversary model to be able to talk about security.

- What other attack possibilities exist for sensor networks and how much effort do they cost to be pursued? For example, are software-based node capture attacks a real threat? Are side-channel attacks on sensor nodes possible? We believe both to be true, but we are unaware of any work which has tried it.

- As cross-layer integration is especially important for resource-constrained sensor nodes, careful design decisions must be taken concerning which

security means to put into which layer. For example TinySec [84], a link layer encryption and message integrity protection mechanism, is integrated into the radio stack of MICA Motes.

- Building secure sensor networks, especially with respect to active adversary, remains a challenge. Can it be done by combining existing solutions, such as random key predistribution, secure routing, secure data aggregation, or would it be too expensive in terms of energy?

Overall, we speculate that probabilistic algorithms which exploit the redundancy of the sensor network to cause high effort for the adversary will be good candidates to establish security in such networks. These algorithms are not suitable to establish perfect security, but offer better scalability and save resources. The security goals of sensor networks will be probabilistic and depend on the strength of the adversary.

Chapter 3

Access Control Issues in Wireless Sensor Networks

3.1 Introduction

In contrast to traditional types of computer networks, sensor networks are supposed to be application-specific. This means that the design of a sensor network will depend on its application area. Moreover, resource-efficiency will ask for cross-layer optimization rather than for clearly layered protocol design of the ISO/OSI or TCP/IP style. Therefore, the intended application of the sensor network will affect many aspects of the network, from hardware and radio communication to topology control, routing mechanisms and communication patterns. Sensor networks for habitat monitoring, home automation, wildfire detection, supply chain management etc. will differ considerably from each other. Thus, it is difficult to design security solutions for an "abstract" sensor network. On the other hand, it would be useful to have security solutions for specific design patterns which are likely to be present in many sensor networks.

In this thesis, we assume that the considered sensor networks operate according to the following paradigm. The sensor network is spread over some geographic area (it can also be a building) and consists of a large amount of nodes. The maintainer of the sensor network offers services to a large number of users. The users can post queries to the sensor network. These queries are propagated into the network, the requested data is collected and sent back to the user. We believe this paradigm to be generic and useful for many applications.

In a formal notation, we describe the procedure for the user U to get the

service from the sensor network *WSN* as follows:

$$U \rightarrow WSN\colon\ q$$
$$WSN \rightarrow U\colon\ a(q)$$

The user sends a query q to the WSN, and receives the answer $a(q)$ to this query.

With the security goals from Section 2.3 in mind, a robust, confidential and mutually authenticated communication channel should be implemented between the user and the WSN. This channel would guarantee authenticity of the communication partners and integrity, confidentiality and availability of the messages.

One of the most obvious methods to set up such a channel is to set up individual channels between the user and every single sensor node. However, this solution contradicts the concept of sensor networks which includes cooperation between the sensor nodes such as multi-hop communication, in-network data processing and aggregation of the requested data while it is transported to the user. A secure authenticated channel between two entities implies that nobody can interfere in the channel, which excludes any cooperation.

Therefore, other solutions are required in sensor networks. In the following, we consider an interesting subproblem that arises in securing sensor networks with the described communication pattern: *access control to the sensor network data*.

The problem of access control can be considered as restricting access to resources to privileged entities [106]. That is, only legitimate users should be able to access the data.

Which possibilities does an adversary have for gaining unauthorized access to the data? On the one hand, the adversary can try to impersonate a legitimate user. Thus, the sensor network should implement an access control procedure for the users. On the other hand, the adversary can also attack other mechanisms and protocols of the sensor network. This means that the access control mechanism for the users (outside security, see Section 2.3) as well as internal mechanisms, e.g., routing, in-network processing and data aggregation (inside security), should be secure against the considered adversary type.

The most severe security breach with respect to access control to the data is the possibility for the adversary to send arbitrary queries to the sensor network. In this case, the adversary would receive the same service as legitimate users. In this thesis we consider the mechanisms that prevent

the adversary from sending its own queries into the network in the presence of node capture attacks.

The rest of this chapter is organized as follows. System and adversary models employed for the analysis of access control throughout this thesis are presented in Section 3.2. Furthermore we consider special characteristics of access control in sensor networks with respect to the specific system and adversary model in Section 3.3, and present access control phases in Section 3.4. Finally, we develop a design space for access control solutions in Section 3.5 and conclude in Section 3.6.

3.2 System and Adversary Model

Sensor network architecture. We consider a sensor network which is deployed over a large geographic area. The network consists of a large number of resource constrained sensor nodes and a base station that has more resources than the sensor nodes. For example, it can be laptop class device. The base station is a trusted entity which can be used for network management, including security-related tasks. The adversary cannot gain control over the base station.

Users. The maintainer of the sensor network offers services to a large number of mobile users. Legitimate users can send queries to the sensor network.

Throughout this thesis we consider two alternative possibilities of accessing the network for the users: They can either log in into the base station which communicates with the sensor nodes on their behalf, or they can go into the network area with some wireless mobile device like a PDA or a mobile phone and access the nodes in their communication range.

In the latter case, we assume the number of nodes that are in communication range of the user to be not less than some threshold, this threshold being a system parameter.

Queries. Queries can be injected into the sensor network either at a base station (like in TinyDB [101] or Cougar [150]) or at any sensor node (like in Directed Diffusion [81]). The queries may be first optimized or otherwise processed at the place of injection and then they are disseminated into the sensor network using multihop communication according to some query processing mechanism.

Adversary. The goal of the adversary is to post *arbitrary* unauthorized queries to the sensor network or to disrupt the sensor network operation such that the queries of the legitimate users are not answered by the network.

Following the terminology and the security goals described in Section 2.3, the adversary seeks to violate *confidentiality* and *availability* of the sensor network data, and *integrity* of data requests with respect to *outside security*.

We do not consider availability and integrity of sensor network data, as well as confidentiality of data requests.

The adversary can capture a small amount of sensor nodes, that is, it can read out all their memory contents, and make them run arbitrary programs.

This means that in the terminology developed in Section 2.5 our adversary consists of two basic adversary models:

$$\{\mathcal{A}_1(global, eavesdrop), \mathcal{A}_2(distributed, reprogramming)\}$$

In particular, this means that the adversary can analyze network traffic before compromising nodes. Thus, we do not consider any special distribution of sensor nodes and assume that the distribution of the captured nodes is the worst possible.

As the adversary can capture some sensor nodes, it would have access to all data measured by these sensor nodes, and to all data routed through them (in case the data are unprotected or can be decrypted by means of captured cryptographic keys). This data disclosure cannot be prevented in face of node captures. Nevertheless, the adversary should not be able to post *arbitrary* queries to the sensor network, thereby receiving the same service as the honest users.

3.3 Access Control Problems in Sensor Networks

According to our system model, the users can inject queries into the sensor network either at the base station or at some sensor nodes. We consider these two possibilities in the face of node capture attacks.

3.3.1 Access Control via Base Station

In the case the users can post their queries using the base station, an access control mechanism should be implemented in the base station. However, this "front-end" access control mechanism alone does not suffice since an impostor could masquerade as a sensor node and access the data *behind* the base station.

For example, in TinyDB [101] the queries are disseminated using flooding. During the flooding process, a spanning tree for sending the answer to the query back to the base station is constructed. Thus, even if there is an access control mechanism at the base station, the adversary can still send arbitrary queries if it captures one sensor node. The captured node would send the adversary's queries to its neighbors, the queries would propagate into the network, and the spanning trees for the answers will be rooted at the compromised node.

This means that some form of access control must also apply in the "back-end" of the communication chain. That is, each query should be accompanied by a proof of its legitimacy, such that each sensor node can verify that the query was inserted into the network by the base station. This proof of legitimacy must be of course secure in the presence of the anticipated adversary, in our case, in the presence of node captures.

3.3.2 Decentralized Access Control

If the users can post their queries using the ordinary sensor nodes, then the access control mechanism should be built into each node, as the "front-end" for the user can be any sensor node. Moreover, just like in the case of the base station, a proof of legitimacy is needed for the "back-end" of the network, that is, for all nodes which did not participate in the process of access control directly. Alternatively, the user could directly log in into each sensor node which holds the requested information. However, as we assume a large sensor network, this solution is impractical.

Are there any additional problems which arise when the users can start their queries at the sensor nodes? Consider the following example.

Directed Diffusion [81], a popular paradigm for organizing sensor networks, allows the user to post queries at any arbitrary sensor node (called the *sink*). The sink then floods the network with the query. After some time, sensor nodes start sending their aggregated data towards the sink. The sink gives the data to the user. In this case, to prevent the adversary from querying the sensor network, an access control mechanism should be built into each sensor node.

Consider an adversary who wants to gain unauthorized access to the data. It can try either to subvert the access control mechanism, or to find some weaker point in the sensor network architecture. For example, if the communication between the sensors happens without encryption and authentication, the adversary would bypass access control mechanism by directly attacking the communication protocol (eavesdrop, insert its own

messages).

However, even if all communication between the nodes is properly encrypted and authenticated, access control remains a separate problem which has to be solved. To illustrate this, consider a sensor network with Directed Diffusion mechanism where the sink is able to organize secure and authenticated communication with all other sensor nodes.

Suppose that, additionally to secure authenticated communication with the sink, some access control mechanism is built into each sensor node. However, if the adversary can disable the access control mechanism on a single sensor node, for example by capturing it, it would be able to query the entire sensor network. This single sensor, acting as a *new* sink, will build a secure authenticated channel to other sensor nodes, but this would not prevent the adversary from unauthorized data access. This happens because any arbitrary sensor is authorized to act on behalf of the user.

From the above example we can see that in case the users can start their queries at any node in the network, the "front-end" access control mechanism for the users cannot rely on a single sensor, but should be able to withstand node capture. Thus, to prove user's legitimacy, the user should be able to log in into multiple sensor nodes. Moreover, these nodes should be able to provide a proof of the query legitimacy for the rest of the nodes.

3.4 Two Phases of Access Control to Sensor Network Data

In the previous section we have seen that a practical and secure in the face of node captures access control solution can be divided into the "front-end" part, where the user directly logs in into the base station or some sensor nodes, and the "back-end" part, where the nodes which did not communicate with the user directly can assure themselves that the query was posted by a legitimate user by means of some proof appended to it. In the following we call this proof the *authenticator*.

We call the "front-end" part *user access control* and the "back-end" part *authenticated querying*. In the following we formally define both phases of access control.

3.4.1 User Access Control

The process of access control can be separated into authentication and authorization [66, 106]. Authentication means establishing a relation between

the user and some identity. Authorization means establishing a relation between a user and a set of privileges (access rights or allowed operations). Here we consider user authentication as a key operation for user access control.

3.4.1.1 User Access Control at the Base Station

Menezes et al. [106, p. 386] define the term *entity authentication* as "... the process whereby one party is assured [...] of the identity of a second party involved in a protocol...". We call the first party the *prover* P, and the second party the *verifier* V.

We now define the properties of authentication protocols. These properties are defined with respect to the two primitive operations of authentication: (1) *authenticate*(V, I) is invoked by the prover P whenever P would like to be authenticated by V using identity $I \in \mathcal{I}$; (2) *associate*(P, I) is invoked by the verifier whenever it has established the relation between P and some identity I.

An authentication protocol is correct if the identity associated to P by V is the "real" identity of P. If P is dishonest or claims to have a fake identity this is indicated by a special value $\perp$ which is supposed to be distinct from any value in $\mathcal{I}$. Authentication is *successful* if V invokes *associate*(P, I) with some $I \neq \perp$.

Definition 1 (Authentication) *A protocol solves* authentication *if it satisfies the following properties:*

- *(Termination) If P invokes authenticate(V, I) then eventually an honest V invokes associate(P, I) for some $I \in \mathcal{I}$ or $I = \perp$.*
- *(Validity) An honest verifier V invokes associate(P, I) for $I \in \mathcal{I}$ only if P in fact has identity I.*

3.4.1.2 Decentralized User Access Control

In the case of decentralized access control, the user has to perform authentication and authorization not with a single party (the base station), but with a distributed entity, which is composed of several sensor nodes, as the access control operation cannot depend on a single node. One possibility to do this is for the user to execute a separate access control protocol with each of the nodes. We denote the distributed verifier as a set of entities $\mathcal{V} = \{V_1, \ldots . V_n\}$

Essential is that the distributed access control protocol should be resilient to the capture of sensor nodes. That is, even if the adversary captures some nodes, these nodes together should not be able to give the adversary access to the data. We discuss corresponding examples in Sections 5.5 and 6.6.

We now introduce the notion of n-authentication, a resilient version of simple authentication. To distinguish the primitive operations of simple authentication from those of n-authentication we denote the latter ones with n-*associate*(P, I) and n-*authenticate*$(\mathcal{V}, I)$. Note that n-authenticate refers to the entire set of verifiers while n-associate just refers to a single prover.

Definition 2 (n-Authentication) *A protocol solves n-authentication if it satisfies the following properties:*

- *(Termination) If P invokes n-authenticate$(\mathcal{V}, I)$ then eventually all honest $V_i \in \mathcal{V}$ invoke n-associate(P, I_i) for some $I_i \in \mathcal{I}$ or $I_i = \perp$.*
- *(Validity) An honest verifier V_i invokes n-associate(P, I) only if P in fact has identity $I \in \mathcal{I}$.*
- *(Agreement) If honest verifier V_i invokes n-associate(P, I') and honest verifier V_j invokes n-associate(P, I'') then $I' = I''$.*

If we assume that at most t verifiers fail, then n-authentication ensures that the remaining (at least $n - t$) verifiers eventually successfully authenticate an honest prover and that they agree on his identity. If a prover is dishonest or claims to have a fake identity then all honest verifiers will return $\perp$ so that the prover is not authenticated.

3.4.2 Authenticated Querying

We now define the second phase of the access control which belongs to the realm of inside security in sensor networks and is called *authenticated querying*.

Let *WSN* be a sensor network. After receiving a query, each sensor node *decides* whether the query was inserted by a legitimate user. If its decision is positive, we say that the node *accepts* the query. Consider an arbitrary query q. Let S_q be the set of all nodes which must process the query in order to give the required answer to the user. The *WSN* design satisfies *authenticated querying* if it satisfies the following properties:

- (Safety) If a sensor node s in *WSN* accepts the query q as a legitimate query, then q was originated by a legitimate user.

- (Liveness) Any legitimate query q will be processed by at least all nodes in S_q.

The problem of authenticated querying is often considered in the literature on sensor network security under the name of *broadcast authentication*. Broadcast authentication is a classical computer security problem where a singe sender needs to authenticate its messages to multiple receivers. Some of the receivers are supposed to be "dishonest" with the goal of computing a fake authenticated message that the "honest" receivers accept as being sent by the sender [106].

In case the users post their queries using the base station, the sensor nodes need to ascertain that the query originates from the base station. In this case the base station appends an authenticator to the query. For example, the base station could digitally sign the query, and the sensor nodes could verify the signature using the preloaded public key of the base station. However, this solution implies the use of public-key cryptography which is often considered too resource-intensive for being practical in sensor networks. For example, using the most efficient (to our knowledge) library for asymmetric cryptography [70], verification of a digital signature would still require half a second.

In Chapters 5 and 6 we consider more efficient ways of generating an authenticator. More precisely, we develop novel solutions for *broadcast authentication* in presence of node capture attacks.

If the users directly log in into locally available sensor nodes, these nodes have to jointly compute the authenticator. This method is bound to be more difficult to realize, as the sensor nodes can be captured, and therefore, in contrast to the base station which cannot be captured, a single sensor node cannot be entrusted with the full information needed to generate the authenticator. In Sections 5.5 and 6.6 we make suggestions on how to organize this kind of authenticated querying.

3.4.3 Putting the Two Phases Together

If the number of users is large, the natural method to use for authentication is public-key cryptography because of its scalability. On the other hand, public-key cryptography is power-consuming, so the sensor nodes should communicate with each other using symmetric-key cryptography.

In the concept presented here we let the base station, or the nodes in the communication range of the user to serve as interpreters (or a gateway) between the "public-key crypto world" of the user and the "symmetric-key

crypto world" of the WSN. The user communicates with the base station or to the nodes in its communication range using public-key cryptography, and the base station or the nodes then communicate with the remainder of the sensor network on behalf of the user using symmetric-key cryptography. This happens in the following two phases:

1. *Secure channel setup between the user and the WSN*: The user executes a mutually authenticated key establishment protocol [106] using public-key cryptography with the base station or with some specified sensor nodes. For example, the number of such nodes can be specified, or the nodes can be selected according to some other criteria. The protocol results in the establishment of shared session keys between the user and each honest node which participated in the protocol run.

2. *Authenticated querying*: After the successful secure channel setup, the base station or the nodes in user's proximity forward user's queries into the sensor network and append to them some additional information enabling the other nodes to verify the legitimacy of the query.

The first phase naturally includes the user authentication phase considered in Section 3.4.1. Although a secure channel is not required for access control, it is considered here because secure channel setup is a very well studied standard procedure and incurs marginal additional costs in comparison to unilateral user authentication. Additionally, secure channels between the users and the WSN are very likely to be required in the overall WSN design. For example, the answer to the query should be kept confidential. Moreover, the user should also be able to ascertain that it communicates with the genuine sensor network.

In contrast to the first phase, we do not consider authenticated answers (reports) to the query in the second phase. Although authenticated reports are just as important as authenticated queries, there is no standard solution to this problem. Moreover, this is one of the most interesting current research topics (see Sections 4.8 and 4.9). However, this problem is out of scope of this thesis.

We briefly outline a possible solution to access control assuming that the query is addressed to a single sensor node s. The user first sends the query to the surrounding nodes using the previously established secure channels. Each node computes a message authentication code (MAC) on the query using the key shared with the node s. For example, these keys could be computed using polynomial-based key predistribution [95]. The computed

MACs are sent back to the user who appends them to the query. The node s answers the query only if enough MACs are appended.

Note that in this solution, no coordination between the nodes in user's proximity is required. The node s answers the query only if enough MACs are appended to it. Such solutions should generally be preferred, as coordination requires additional messages, and therefore, additional resource consumption.

The above solution explains the idea and shows the feasibility of our approach. However, it has many drawbacks. It does not scale well, and the legitimacy of the query cannot be verified by intermediate sensor nodes which route the query to the node s. The presence of the latter feature would enable early rejection of illegitimate queries. We consider more practical and sophisticated solutions in Chapters 5 and 6.

3.4.4 Choice of the Public-Key Cryptosystem

We now consider which public-key cryptosystems are well suited for the user access control.

RSA with small public exponent ([71, 145]) and Rabin public key cryptosystems [63] have got fast algorithms for encryption and digital signature verification. However, decryption and signature generation are slow and resource-demanding. Therefore, these cryptosystems can be used in sensor networks only if the sensor nodes are not required to decrypt or to sign messages.

In contrast, elliptic curve cryptosystems (ECC) [71, 102] require more overhead for encryption and signature verification than for decryption and signing. Nevertheless, with ECC, not only encryption and signature verification, but also decryption and signing are feasible for sensor nodes. As we target mutual authentication, ECC seems to be more suitable for implementation of robust user authentication, even though the sensor nodes in this case have to execute relatively expensive for ECC signature verification.

The feasibility of ECC for sensor networks was demonstrated by Gupta et al. [70]. They implemented the SSL protocol on MICA2 motes using elliptic curve cryptography. Their SSL handshake takes less than 4s.

3.5 Design Space for Access Control

Two following dimensions can be identified for access control in sensor networks:

	multihop: no	multihop: yes
BS: yes	direct base station AC Section 3.5.1.1	remote base station AC Section 3.5.1.2
BS: no	direct decentralized AC Section 3.5.2.1	remote decentralized AC Section 3.5.2.2

Table 3.1: Design space for realizing access control (AC) in sensor networks. "BS: yes" means that the base station must be accessed before the query processing can be started by the network. "Multihop: yes" means that multihop communication is needed before the query processing can be started by the network.

- The user has to communicate with the base station in order to post queries vs. the query can be started at some sensor nodes.
- The sensor network has to forward some data using multihop communication before the user can start posting queries vs. the query can be started locally.

These two dimensions give four possibilities for access control (AC) as depicted in Table 3.1: direct base station AC, remote base station AC, direct decentralized AC, and remote decentralized AC.

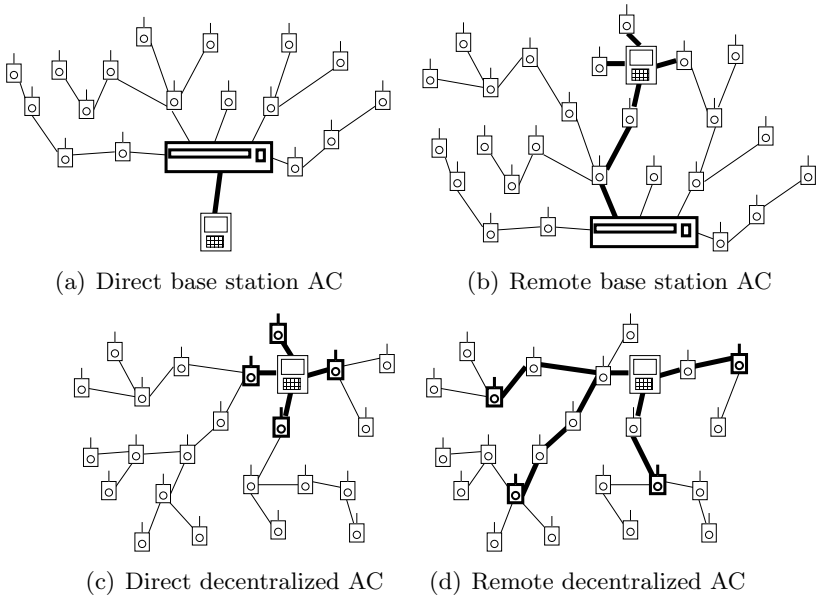
Each of these mechanisms is appropriate in different situations and requires different solutions. In Sections 3.5.1 and 3.5.2 each mechanism from the design space is discussed. The mechanisms are also depicted in Figures 3.1(a)-(d).

3.5.1 Base Station Access Control

Base stations are supposed to have more resources and to be better protected against attacks than sensor nodes. Therefore, using base station is a natural approach to organize such a critical task as access control.

3.5.1.1 Direct Base Station Access Control

Here the query is always started at the base station, either by physically approaching it with a device and connecting to it (wirelessly or not), or by routing the query through some external network (e.g., the Internet) which is connected to the base station. Users log in into the base station using an arbitrary client/server authentication protocol [106]. If the user is



(a) Direct base station AC (b) Remote base station AC

(c) Direct decentralized AC (d) Remote decentralized AC

Figure 3.1: Design space for access control (AC) in sensor networks. Entities that generate the authenticator for the query, as well as the connections (sometimes, multihop) between the network and the user, are depicted with thick lines.

successfully authenticated, it can post arbitrary queries to the base station. The base station forwards (possibly optimized) user's queries into the sensor network.

In this case, the base station has to implement authenticated querying (Section 3.4.2). Then all nodes that process the query can verify that it originated at the base station. Any method for broadcast authentication presented in Section 4.6 can be used here.

3.5.1.2 Remote Base Station Access Control

The query can be started on some sensor nodes. Nodes are not concerned with query authentication, but just route the authentication information to the base station, the base station authenticates the user and then gives permission to the sensor network to answer user's queries. Thus, the base station helps to establish trust between the sensor network and the users. Here, at least two scenarios are possible:

- User's queries are always routed to the base station. The base station

sends authenticated queries into the network on behalf of the user, receives the answers, and sends the answers back to the user. In this case, an SSL-like protocol can be used to set up a secure authenticated channel between the user and the base station.

- The base station generates a kind of "ticket" (using Kerberos terminology) which enables the user to talk to the sensor network for some time. The ticket is sent back to the user who uses it to generate authenticated queries.

Note that the user may still have to authenticate to several sensor nodes, because otherwise an adversary could flood the network with fake requests that will have to be routed to the base station.

3.5.1.3 Access Control Using Base Station: Pros and Cons

Advantages. The base station has more resources than a sensor node and therefore, can implement stronger security measures. It can be placed in dedicated locations and maintained by humans. This makes the base station more reliable and secure: It can be protected from physical access, and DoS or penetration attacks are more likely to be spotted quickly.

Disadvantages. The base station serves as a dedicated authentication server. Therefore, it must be very well protected from both physical and remote access by unauthorized entities. In the literature it is usually assumed that to take over a base station is more difficult than a sensor node. In practice, the reverse could be the case. For example, if the base station is connected to a popular web server with known vulnerabilities, the penetration of the base station could be a matter of utilizing an available exploit. Besides, it is not always possible or desirable to place a base station into a dedicated secure location, especially if it is supposed to communicate with the sensor nodes wirelessly.

Furthermore, if the direct physical access is needed for the authentication (e.g., wireless communication with the base station), it might be inconvenient to the user to walk through the half deployment area to the base station while needing the data from nodes in user's proximity. In case of remote access, several messages have to be routed between the base station and the user by the sensor network, which can be impractical if the base station is far away. The user might also have to wait for the answer of the far away base station while needing the data from the nodes close-by.

And finally, as the nodes close to the base station are more heavily loaded with communication, their energy is exhausted more quickly, which leads to shorter network lifetime.

3.5.2 Decentralized Access Control

In cases where the base station cannot be used for access control, an access control mechanism should be built into sensor nodes. However, as shown in Section 3.3.2, relying on any arbitrary node for access control is not sound in face of node capture attacks. A natural solution here would be to use a distributed algorithm for access control.

3.5.2.1 Direct Decentralized Access Control

Here the legitimacy of the user is verified by the nodes in user's location, e.g., in his communication range. Of course, even if the nodes in user's proximity successfully verified the query, they still need some means to tell to the rest of the sensor network that this query originates from a legitimate user. That is, at least all nodes that process the query should be able to verify its legitimacy.

3.5.2.2 Remote Decentralized Access Control

The legitimacy of the user is verified by several nodes which not need to be close to the user. These nodes can be specially chosen for this purpose, then the network architecture might be heterogeneous, with dedicated authentication devices placed in some locations. On the other hand, these nodes might be selected from the set of all nodes according to some algorithm. We do not consider remote decentralized access control in this thesis and leave it to future work.

3.5.2.3 Access Control without Base Station: Pros and Cons

Advantages. Users can start queries locally in the sensor network, without going to the base station or some other access point (e.g., an Internet terminal). Furthermore, the query can be processed and answered locally. No routing to the base station is needed for answering the query. Still, routing can be needed if the query concerns sensor data which are not in the user's proximity. And last but not least, if the base station is overloaded or taken over, the data are still available and not compromised (at least,

as long as the compromised base station can be excluded from the network management).

Disadvantages. The most severe disadvantage is that user authentication costs extra computation and communication power, especially if it is done in distributed fashion, using replication and agreement techniques, or public-key cryptography. Distributed algorithms have to be applied in order to cope with unreliability and insecurity of individual nodes.

3.6 Conclusions

Access control to sensor network data consists of two phases. During the *user access control* the user proves its legitimacy to the base station or to the surrounding sensor nodes. This phase belongs to the realm of outside security. In the phase of *authenticated querying* user's query is forwarded into the network accompanied by an *authenticator*. The authenticator enables the nodes that did not participate in the user access control phase to verify the legitimacy of the query.

Depending on the sensor network architecture, centralized (via base station) or decentralized data access is possible. This calls for different solutions to the data access control.

In the following Chapter 4 we present some background information and related work on sensor network security that is relevant for access control mechanisms developed in this thesis. Furthermore, in Chapters 5 and 6 we consider some approaches to the centralized access control, and also discuss issues that arise in case these solutions should be adapted for decentralized access control.

Chapter 4

Background And Related Work on WSN Security

4.1 Introduction

We now present some background on sensor network architecture and security, and also some related work that considers access control in sensor networks from different points of view.

In order to design an access control system for sensor networks one needs to know how the sensor networks work. Therefore, in Section 4.2 we introduce query processing systems for sensor networks. These systems are used to extract data from the sensor network. They include mechanisms for query dissemination, in-network data processing and data collection. We further concentrate on query dissemination in Section 4.3. This is due to the fact that any access control mechanism should prevent the adversary from interfering with the query dissemination mechanism. This means that the network should be protected from injection of unauthorized queries.

In Section 4.4 we give background information on the access control problem in traditional security, and also an overview of related work on this topic.

In the rest of this chapter we consider security problems from the realm of sensor networks that are especially relevant for our work. We start with the establishment of cryptographic keys (Section 4.5) that is the precondition for every security mechanism. One of most distinctive features of key establishment techniques in sensor networks is the frequent use of probabilistic mechanisms that guarantee key secrecy only with some predefined probability. In Chapter 6 of this work we develop an access control mechanism that

follows the same paradigm.

Section 4.6 considers broadcast authentication in sensor networks. As noted in the previous chapter, access control in sensor networks consist of two phases (Section 3.4): user access control and authenticated querying. The latter is a special case of broadcast authentication. Here we consider desirable properties for broadcast authentication in sensor network such that it can be employed for authenticated querying and give a detailed overview of the most relevant related work.

In Sections 4.7 we consider secure code update techniques. Secure code update constitutes a specific variant of broadcast authentication that usually cannot be used for authenticated querying in general. Nevertheless, secure code update techniques can be used in some specific cases, e.g. if the content of queries is known beforehand.

In Sections 4.8 and 4.9 we present related work on the authentication of the answers to the query, also called report authentication. Although the information flow in this case is from the sensor nodes to the user, report authentication and query authentication techniques are inherently interdependent. Indeed, in Chapter 5 of this thesis we present a mechanism for authenticated querying that is based on an already existing mechanism for report authentication.

We conclude our overview in Section 4.10.

4.2 Query Processing Systems

Query processing systems for sensor network comprise mechanisms for query optimization at the base station, query dissemination, local query processing that takes place on single sensor nodes, and report collection. Thus, any access control mechanism should work together with the query processing system employed in the particular sensor network and ensure that only legitimate entities can start the queries.

One of the most popular query processing systems is Directed Diffusion [81]. It allows the user to post queries (*tasks*) starting at any arbitrary sensor nodes (called the *sink*). The sink firstly floods the network with an exploratory query (an *interest*), and the process of the interest dissemination, as well the process of getting the answers to the exploratory query, is used to build up a network structure (usually a spanning tree) for the dissemination of the "real" query and the report collection. Directed Diffusion uses a customized language for the task description.

TinyDB [101] and Cougar [150] suggest distributed query processing sys-

tems where the queries are formulated in the SQL-like manner. The queries are submitted to a powerful base station that optimizes them and are disseminated into the network via flooding. The routing tree for the data collection is constructed during the flooding process.

Without going into details of how the query is processed by the network, it is important to notice that the query processing systems use flooding for query dissemination. Some useful flooding variants are considered in the next section.

4.3 Query Dissemination

Query dissemination is an especially interesting issue for access control, as the adversary should be prevented from inserting its own queries at any point of the network. Thus, we consider data dissemination in the direction from the base station to the sensor nodes.

As mentioned in the previous section, a widespread way of query dissemination is flooding. In the most basic flooding protocol every node relays the received message to all its neighbors if it has not received this message previously, and drops the message otherwise. Unfortunately, this method is both unreliable (nodes are not guaranteed to receive the packets) and wastes energy because of redundant packet broadcasts [136]. Furthermore, most nodes receive the query more than once. This should be avoided, since receiving messages also consumes energy.

To avoid the disadvantages of simple flooding, several mechanisms have been proposed [129, 146, 152]. For example, in probabilistic flooding the nodes relay messages with some predefined probability. In counter-based flooding, if a node hears more than k of its neighbors rebroadcast the message, it suppresses its own transmission. In neighbor-knowledge-broadcasting schemes, nodes use detailed local topology information to determine which nodes must rebroadcast a message.

To improve reliability of query dissemination, transport protocols for sensor networks were developed. The protocol PSFQ (*pump slowly, fetch quickly*) [143] reliably disseminates data streams consisting of many packets. The packets are distributed "slowly" in the network, such that if some node lost a packet with a specific sequence number, it does not broadcast the packets with higher sequence number to its neighbors, but asks the neighbors to supply it with the lost packet first. This way, packet losses are not propagated through the network, but recovered "quickly". There is also a mechanism to report to the base station about the delivery status.

The transport protocol GARUDA [114] disseminates data streams using a structure called *core* that approximates the minimum dominating set of the network and is constructed during the dissemination of the first packet. Packets are first reliably distributed to the core members, the latter relay the packets to all other nodes.

We note that any security mechanisms for query dissemination should be able to work with the corresponding query dissemination protocols. In this thesis, we develop protocols for secure query dissemination along a spanning tree (see Chapter 5) as well as protocols that can work on top of any query dissemination technique (see Chapter 6).

4.4 Access Control

A method of access control should provide access to legitimate users and deny access to illegitimate ones. There are two main issues in this process [66,106]. *Authentication* means establishing a relation between the user and some identity. *Authorization* means establishing a relation between a user and a set of privileges (access rights or allowed operations). An *identity* is the individuality property of a user which ideally cannot be forged or copied. In practice, identities are implemented by items which users know (passwords), possess (secret keys or security tokens) or properties which they have (biometrics).

In authentication, a user sends its name (e.g., IP address) and proof of its identity to the base station, and the base station should be able to decide whether or not the identity is valid and in fact belongs to the user of that name. This can be done, e.g., by verifying a digitally signed certificate.

In authorization, a user sends its name together with the requested access operations (e.g., read, write) to the base station, and the base station should be able to decide whether or not this user is allowed to perform this operation. This is usually implemented by access control lists.

Finally, in access control, a user sends its name, identity, and the requested operation to the base station. The base station first authenticates the user, i.e., it checks the validity of the name and identity. Upon successful authentication, the base station authorizes the user, i.e., it checks the access permissions of that user. If both checks succeed, the user is granted access to the data.

In practice, authentication, authorization and data access are combined into one single operation. A request is sent, the access control mechanism checks legitimacy (authentication and authorization), and sends a response

back to the user (which may be the data requested or a message "access denied").

To our knowledge, algorithms for access control in sensor networks that can withstand malicious node capture have not been considered so far. However, there is rich literature on authentication in wireless ad hoc networks. In this context, authentication of single nodes to other members of the ad hoc network is considered, which is an aspect of inside security and so unrelated to access control for outsiders (users) that we consider here. Some solutions use threshold cryptography [88, 155], others use hierarchical network architecture and public-key cryptography [137] or symmetric mechanisms [10]. However, none of these methods is really applicable to sensor networks due to extensive use of resource consuming cryptography or unsuitable methods for bootstrapping trust, e.g., using physical contact of devices as in [10].

4.5 Key Establishment

The establishment of pairwise keys is a crucial prerequisite for virtually all security schemes. One of the most popular ideas in sensor network is to organize symmetric key establishment by means of *random key predistribution* [37, 55, 79, 158], where the keys are drawn from a large key pool such that any two sensor nodes receive several keys before the deployment. After the deployment, the nodes have to discover whether they share any keys with their neighbors.

Further schemes combine random key predistribution with the previously known methods for conference key establishment where each node can compute the key it shares with any other node [53, 97].

The *localised encryption and authentication protocol (LEAP)* presented in [156] finds that different tasks require different security policies. Hence, it establishes four types of keys to meet these different requirements. In particular, each node maintains an individual key shared with the base station, a group key for each group the node participates in, a cluster key shared with all its neighbors and a pairwise keys for each of its immediate neighbors. A similar approach is taken in [48].

Under the assumption that the communication in a WSN is not between arbitrary nodes but more like a tree aggregation towards a sink, the key distribution approach presented in [159] can be used to securely establish keys.

4.6 Broadcast Authentication

In order to allow only legitimate entities to query the sensor network for data, the sensor nodes should be able to verify the legitimacy of the queries introduced into the network. Therefore, a legitimate sender should append some non-forgeable authentication information (called the *authenticator* in the following) to its queries.

Precisely this feature offer broadcast authentication protocols. The authentication of of broadcast messages is an essential subject in sensor networks and a core of the access control solutions presented in this thesis. Therefore, we present existing broadcast authentication solutions in some depth here. We concentrate on solutions that are resilient to node compromise, i.e., to the *reprogramming* adversary (see Section 2.5.3).

4.6.1 Desirable Properties for Broadcast Authentication

We firstly present some properties that a broadcast authentication protocol should satisfy in order to be most useful for access control.

Broadcast authentication in sensor networks should have *low computation overhead*, preferably, in order of milliseconds. The authenticator size should not exceed the size of some dozen of bytes, that is, *low communication overhead* is required. Moreover, *robustness to packet loss* is a very desirable property for unreliable wireless communication. This means that the authenticator of the current message should not depend on the previous or future messages. *Sending messages at irregular times* should also be possible, such that the scheme does not depend on periodical transmission of some information. Finally, *immediate message authentication* is a very desirable property. This means that the messages can be authenticated immediately after the reception, and do not have to be disseminated in the whole network before the authenticator can be verified.

Apart from the above properties introduced by Luk et al. [100], another desirable property for broadcast authentication schemes is the ability to authenticate *unbounded number of messages* after the initialization. Periodical re-initialization required in the broadcast authentication schemes is usually quite cumbersome and resources

In the following, we present existing schemes for broadcast authentication. We note that none of these schemes satisfies all desirable properties mentioned above.

4.6.2 Broadcast Authentication using Time Synchronization

The first method to authenticate broadcasts in sensor networks, a protocol called μTESLA, is presented by Perrig et al. [118]. As μTESLA is one of the most efficient broadcast authentication mechanisms for WSNs, its performance is a good reference value and will be used for comparison to the protocols developed in this thesis. Therefore, below we present μTESLA in some depth.

The basic idea of this protocol is that messages are authenticated with a secret key that is revealed in a later time interval. This approach manages the task of broadcast authentication with just one message authentication code (MAC) per message. In return it requires all receivers (all sensor nodes in the network) to be loosely time synchronized, i.e., they must agree on the current time up to a small error.

The sender splits up the time into uniform intervals i_0 to i_N. Next, it generates a random key, k_N, for the interval i_N. By repeatedly applying a one-way hash function F to k_N the sender derives the keys for earlier intervals. That is to say, $F(k_N)$ is the key for interval i_{N-1}, $F^2(k_N)$ the key for interval i_{N-2}, and so on. The idea is to use the key from the current period to authenticate broadcast messages, but to reveal the secret key in a later period, e.g., two periods later.

For a message broadcast in interval i, the sender attaches a MAC to this message, computed with the key from the current time interval. Sensor nodes receiving this message first check whether the key used to authenticate this message is still secret. If so, the nodes buffer this message for later verification. When the sender discloses the secret key in interval $i + 2$, the sensor nodes first verify the key by applying F to this key, i.e. by computing $F(k_i)$. If the result of this application equals the key from the last interval, namely k_{i-1} that was already authenticated, the receiver knows that the key is correct. After this, the key can be used to verify the authenticity of the buffered messages from interval i. Figure 4.1 illustrates this example.

μTESLA is a very efficient protocol, as it only requires transmission and verification of one MAC per message, and additionally transmission of one hash value per time interval. Moreover, it tolerates the packet loss.

We note that secure time synchronization incurs additional computation and communication cost [61, 131]. Precise analysis of the impact of time synchronization protocols on the sensor network is out of scope of this thesis. It is clear, however, that in every such protocol the neighboring nodes have to periodically exchange authentic messages with each other. Therefore, in cases where no global time synchronization is needed in the sensor network,

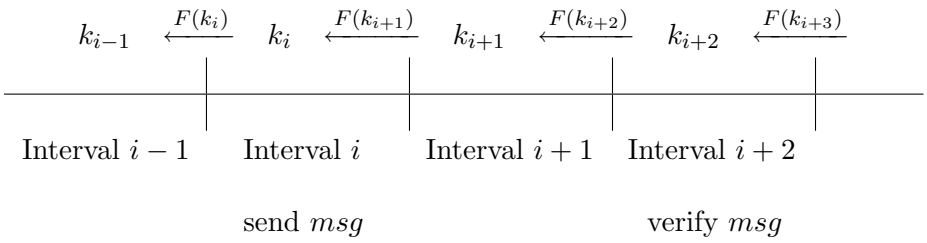


Figure 4.1: μTESLA example. In this example the key is revealed two periods after the message was sent.

μTESLA may perform less efficiently than other solutions that have larger authenticator size but do not require time synchronization.

μTESLA can be extended to provide authentication for messages sent infrequently at irregular intervals [100]. The approach uses a combination of hash trees and hash chains and reduces the verification overhead at the cost of increased authenticator size (around 80 bytes). μTESLA was further developed to scale to the multiple senders [98].

μTESLA requires periodical re-initialization, as the commitments to the key chains have to be distributed periodically. In the original protocol this is done using individual keys shared by the base station with each sensor node. Even if this process can be made more efficient [96], the issue of the re-initialization still remains.

The most severe disadvantage of μTESLA is the delayed message authentication. This opens way to the DoS attack where the adversary sends a lot of bogus messages in some time interval T. As the nodes do not have any possibility to verify the authenticity of the messages, the fake messages spread over the whole network thus exhausting energy.

4.6.3 Broadcast Authentication using Public-Key Cryptography

Digital signatures provide immediate message authentication, does not require time synchronization, and has a moderate communication overhead of around 40 bytes, if elliptic curve cryptography (ECC) is used [70].

Unfortunately, it introduces high computation overhead and high delay of about one second at each node. Clearly, digital signatures cannot be used for time-critical applications. Moreover, if the query has to travel tens of hops, the users may well get impatient till they receive the answer.

For example, Ren et al. [123] present a variety of schemes for broadcast

authentication based on public key cryptography. All schemes allow the participation of multiple and possibly mobile users to query the WSN. A base station is considered a trusted third party. In the simplest approach, for example, the base station serves as certification authority.

Yoon et al. [153] combine μTESLA and public-key cryptography. They divide the network into clusters (cells) of n nodes, each cluster can withstand t compromised nodes. The security of the scheme relies on a tamper-proof *master node* that has to be present in each cell.

As digital signatures require quite a lot of energy for verification, an adversary can run a denial-of-service (DoS) attack against individual nodes by sending to them fake messages. The nodes would have to perform expensive signature verification and would thus exhaust their energy. Dong et al. [51] present a method to thwart this attack by using *pre-authentication filters*.

Another work that prevents DoS attacks on broadcast authentication schemes [144] uses a set of strategies where the sensor nodes decide whether they should forward a message without authenticator verification, or they should verify the authenticator first. The scheme adapts flexibly to the severity of the attack. In this way, broadcast delay is reduced in case most of the messages are authentic. On the other hand, in case of a severe attack, the impact of fake messages is reduced to a small part of the network.

4.6.4 Broadcast Authentication using Symmetric-Key Cryptography

As already mentioned, public-key methods for broadcast authentication introduce high delay and computation overhead. On the other hand, some kind of asymmetric knowledge should be used for broadcast authentication [31] analogous to the public versus private keys in the digital signature schemes.

As we described previously, μTESLA uses time to create asymmetry. This enables the usage of the efficient symmetric-key cryptography. Some other schemes try to avoid time synchronization.

Canetti et al. [35] describe a protocol for broadcast authentication in the Internet. The sender in their setting knows a large number of symmetric keys and uses all of them to compute MACs for a message. Each receiver knows a predefined subset of sender's keys and verify all corresponding MACs. The probability for each receiver to accept a fake message can be made sufficiently small by tuning the parameters. However, the most efficient scheme still uses authenticator size of around 100 bytes, which is too large for sensor networks. In Chapter 6 we use this scheme to develop a more

efficient broadcast authentication protocol.

The paper on *n-layer authentication in sensor networks (n-LQA)* [124] introduces a broadcast authentication scheme that is based on interleaved authentication [139,157]. The authors suggest to divide a network in several concentric rings around the base station, where each ring is assigned its own key. Messages contain authentication information for the next t rings and are only forwarded outwards. This approach guarantees that any injected query will be filtered out unless more than t nodes are compromised. Moreover, it also restricts the area in which the fake query spreads even if more than t nodes are compromised.

Another approach is to use one time signatures [39]. However, these schemes suffer from large storage overhead and have authenticator size of hundreds of bytes.

Finally, a scheme for tree-based networks was introduced recently [36]. The base station shares symmetric keys with each node, and the topological structure of the network is assumed to be static. This scheme constructs a hash tree [107] using the MACs on the broadcast message as the values assigned to the leaves of the tree. The values of all intermediate nodes in the hash tree are computed by concatenating the values of the children and hashing the concatenation. The broadcast message is sent into the network, accompanied by the root of the hash tree. Subsequently, the nodes cooperate in order to verify the authenticity of the message. This scheme induces moderate communication overhead of $O(\log n)$ where n is the number of nodes in the WSN and does not require time synchronization. Unfortunately, it does not provide immediate message authentication.

4.7 Secure Code Update

Secure code update is a variant of broadcast authentication where the data to be authenticated is a new program to be installed on the sensor nodes. This means that the data to be authenticated is known beforehand, and therefore, the full authentication information can be computed before the data broadcast. Moreover, as code update is not a frequent operation, its efficiency is not so critical as the efficiency of query dissemination techniques. Code update is less relevant for this thesis, as we consider authentication of an potentially unbounded number of user's queries, and the content of the queries is not known beforehand.

In the last years, a large number of schemes for authenticated code update emerged, most of them securing the popular code update tool Del-

uge [78]. In Deluge, the code divided into pages, the pages are divided into packets, and the packets are then distributed into the network. All proposals apply either hash chains [54, 86, 93], or hash trees [80], or a combination of the both techniques [45, 91] to the pages and/or to the packets to achieve code authentication. The authentication overhead is of the order of 20 to 400 bytes per page.

In the SCUBA system [125], code updates can not only be used to upgrade sensor nodes, but also to detect and possibly to patch compromised or malfunctioning nodes. The basic idea of this approach is to establish a trusted code base on the sensor node. With this environment created, the base station can determine whether a node has executed a code update correctly or not. In the former case the malicious code is simply overwritten, whereas in the latter case the node can be blacklisted and excluded from the network.

4.8 Authenticated Reports

The authentication of reports sent by sensor nodes to the user is a very important research direction. It is complementary to one aspect of the access control problem investigated in this thesis. Access control seeks to prevent unauthorized data access, resulting in the requirement that all queries be authenticated. Any security architecture for sensor network should implement authentication for both queries and reports. Therefore, it is important to know the existing solutions in order to verify whether they can be combined with the protocols for query authentication.

The *interleaved hop-by-hop authentication scheme* presented in [157] proposes an approach based on multi-hop authentication. Here each node establishes a pairwise key not only with the base station, but also with some of its up- and downstream neighbors. This scheme is also further described later (see Section 5.2.2), as it serves as basis for the authenticated querying scheme developed in Chapter 5.

The *canvas scheme* [139] is very similar to the interleaved hop-by-hop authentication scheme. The main difference between both schemes is that the *canvas* approach generates and sends significantly more MACs, resulting in much larger messages and a higher energy consumption, but also an earlier discovery of bogus messages. Because of high resource requirements, this scheme is less practical for access control and is therefore not considered in the sequel.

Statistical en-route filtering, as introduced in [151], provides a method

to filter out false reports on the path to the base station with a certain probability. Like in *authenticated query flooding* which is introduced in this work (see Chapter 6), all nodes are assigned a subset of a global key pool and can detect the false report with certain probability. Moreover, the base station can accurately detect false reports even if they could not be filtered out before because every node also shares its own pairwise key with the base station.

4.9 Authenticated Aggregation

To save energy, reports in sensor networks are often aggregated. For example, computing the average or the maximum of the sensor readings at intermediate nodes during the data transport to the base station requires significantly less bandwidth than transmitting all individual values to the base station and then computing the required aggregates. As communication requires orders of magnitude more energy than computation, data aggregation is a very important part of sensor network design.

Authentication of aggregated data is one of the most difficult problems, as any malicious intermediate aggregating node can change the aggregated data. Thus, the authenticated aggregation protocols should enforce correct data aggregation, or at least detect the data manipulation.

The first secure aggregation protocol for sensor networks, called SIA [120], employs random sampling mechanisms and interactive proofs. It considers an aggregator node that aggregates the values of multiple sensor nodes. The user can verify that the aggregation value is a good approximation of the true value even in presence of compromised nodes, including the aggregator node.

The ESAWN protocol [30] uses *witnesses* for the verification of the aggregate authenticity. The aggregation happens along an aggregation tree. The witnesses receive the aggregated values as well as the original values, and can therefore verify and confirm the correctness of the aggregation. The user can trade the energy consumption of the verification off against the probability of receiving a faked aggregation value by choosing to verify the aggregation result with some probability p. Thus, ESAWN offers probabilistic aggregate authenticity. In Chapter 6 we present a broadcast authentication protocol that also offers probabilistic security and was developed independently and concurrently to ESAWN.

Finally, Chan et al. [38] present secure hierarchical aggregation. The hierarchical aggregation uses an aggregation tree as a network structure

where the nodes jointly compute a commitment to the aggregated values. The base station then verifies this commitment. The communication overhead of this scheme is logarithmic in the network size [60]. As we described in Section 4.6, the hierarchical aggregation scheme can be used to construct a scheme for efficient broadcast authentication [36].

4.10 Conclusions

In this chapter, we present background information on sensor network architecture and access control, and also related work in the area of sensor network security, such as key establishment, broadcast authentication and report authentication. In the two next chapters (Chapter 5 and 6) we develop schemes for broadcast authentication and utilize them in access control procedures. The developed schemes use some of the presented related work, especially the work on key establishment and report authentication. The used related work schemes are presented in more detail in the corresponding chapters.

Chapter 5

t-Robust Access Control in WSNs

5.1 Introduction

In this chapter we present access control protocols that are able to withstand capture of up to t sensor nodes. We call these protocols *t-robust*. The protocols are fully secure as long as not more than t nodes are compromised, but their security breaks completely in case the number of captured nodes is $t + 1$.

We firstly present a t-robust scheme for authenticated querying, and then consider how to utilize it for t-robust access control.

The developed t-robust solution for authenticated querying has some benefits over other solutions to broadcast authentication presented in Section 4.6. It does not require time synchronization, provides immediate authentication, and can authenticate an unbounded number of messages. It has a low computation overhead, as it uses symmetric-key cryptography, and every node has to compute only two MACs (message authentication codes) per message.

However, there are also some disadvantages. The threshold t is hard to estimate. Would $t = 10$ suffice? How can we guarantee that the adversary would not compromise 11 nodes? Moreover, the communication overhead of the solution grows exponentially in t. This means that large values of t, e.g., $t = 20$, are not practical. Furthermore, the developed scheme requires topological information, as every node has to know at least some of its $(t + 1)$-hop neighbors.

The experience gained during the development of t-robust authenticated

querying shows that for small values of t, the scheme is quite attractive due to the above mentioned advantages. However, for larger values of t, the above disadvantages become severe. Therefore, we developed another scheme that is more suitable for large numbers of compromised nodes, but has probabilistic security guarantees. This scheme is presented in Chapter 6.

This chapter is organized as follows. We firstly discuss related work in Section 5.2 and refine our general system and adversary models (Section 3.2) for the needs of the current chapter in Section 5.3. We then develop and analyze t-robust algorithms for authenticated querying in Section 5.4, and show how to organize t-robust access control in Section 5.5. We conclude in Section 5.6.

5.2 Preliminaries

To our knowledge, the framework presented in this paper is the first approach to build access control for WSNs which can fully withstand capture of up to t nodes (t-*robust* access control).

However, there are t-robust solutions in other areas of the WSN security, such as pairwise key establishment [53,97] and interleaved report authentication [139, 157]. We present some of these schemes in depth in Sections 5.2.1 and 5.2.2, as our solution for authenticated querying is based on them.

Finally, the n-LQA protocol [124] presents a t-robust solution to broadcast authentication that is also based on interleaved authentication and was developed independently and concurrently to our work. In this protocol, the network is organized into concentric layers, the nodes of the ith layer being i hops away from the base station. However, in contrast to our solution, n-LQA cannot be used in decentralized access control. The query in n-LQA always has to start at the base station.

The notion of t-robustness has similarities to work in the area of fault-tolerant and secure data replication. For example, Rabin [121] adapts secret sharing techniques from the area of cryptography to make information available and keep it secure even if up to a certain fraction of nodes behave arbitrarily. As another example, Herlihy and Tygar [73] modify a fault-tolerant replication protocol with secret sharing to maintain the confidentiality of data in a similar setting. However, both papers do not focus on access control.

In the following we present the detailed description of the most relevant for this chapter related work.

5.2.1 Polynomial-based Key Pre-distribution

All schemes considered in the following require that any two nodes can establish a pairwise key with each other. There are several techniques to pre-distribute or establish pairwise keys in wireless sensor networks.

The paper on *polynomial-based key pre-distribution* [95], for instance, presents two suitable approaches. Particularly the *grid-based key pre-distribution* seems to be very promising, since it requires neither much storage nor communication. The basic idea of this approach is as follows:

For n nodes a virtual $m * m$ grid is constructed (where $m = \sqrt{n}$). The base station or a server generates $2m$ bivariate polynomials

$$\{f_i^c(x,y), f_i^r(x,y)\}_{i=0,\ldots,m-1}.$$

In this context, c stands for a column and r for a row in the grid. Each polynomial is of the form

$$f(x,y) = \sum_{i,j}^{t} a_{ij} x^i y^j.$$

Note that the sensor nodes do not have to generate these polynomials on their own. This would be far too expensive in terms of time and energy consumption. A base station, however, is provided with sufficiently large resources to manage this task.

Every node is assigned a unique intersection in the grid, i.e. a position. The node at position $< i,j >$ is informed about its own position and the polynomial shares of $f_i^c(x,y)$ and $f_j^r(x,y)$ by the base station. The polynomial share of a polynomial $f(x,y)$ for node A is $f(A,y)$. Provided with this knowledge, two nodes A and B can compute a common key $f(A,B)$. Node A evaluates $f(A,y)$ at position B and node B evaluates $f(B,y)$ at position A, resulting in $f(B,A) = f(A,B)$ (since f is bivariate). In the grid scheme, two nodes at positions $< i,j >$ and $< i',j' >$ can directly establish a pairwise key if they are in the same column ($i = i'$) or the same row ($j = j'$). Otherwise there must be a node at either $< i,j' >$, $< i',j >$ or both which can be used as mediator to establish a key.

This scheme can tolerate up to t compromised nodes where t is the degree of the polynomials. Even if the t compromised nodes colluded, they would know nothing about the pairwise key between any two non-compromised nodes.

Another benefit of this approach is the storage requirements. Each node has to store only two polynomials and its own position, instead of a single key for each node it needs to communicate with.

5.2.2 Interleaved Hop-by-Hop Authentication

The *interleaved hop-by-hop authentication scheme* [157] provides t-robust authentication for messages sent to the base station along a predefined route. It assumes a large-scale sensor network which is organised in clusters. Large-scale means that there are in average several hops to pass from the centre of a cluster to the base station. Each cluster consist of at least $t+1$ nodes and one of them is considered *cluster head*. Moreover, each cluster is assigned a unique *cluster id* by the base station.

All sensor nodes share a unique secret key with the base station. In addition, they need to know at least a subset of their 1-hop neighbors and must be able to establish a pairwise key with each of them. The authors suggest *LEAP* [156] for this task.

Nodes should also be able to establish pairwise keys with other nodes which are several hops away if needed. For this purpose, polynomial-based key pre-distribution [95] described in the previous section can be used.

Interleaved hop-by-hop authentication tolerates up to t compromised nodes, even if they can collude. This scheme guarantees that the base station will detect any injected false data packets when no more than t nodes are compromised. Furthermore, it also makes it possible to filter out false data packets on their way to the base station. In the worst case, such a false packet is forwarded $O(t^2)$ hops.

The basic idea of the scheme is as follows. In the first phase, each node needs to discover its up- and downstream neighbors which are $t+1$ hops away. When the nodes of a cluster receive a stimulus, i.e. they observe an event, they create two MACs over this event: One using the pairwise key with one of their upstream neighbors and one using their pairwise key shared with the base station.

The $t+1$ cluster members compute MACs for the first $t+1$ nodes along a predefined path to the base station. The cluster head collects the values from the $t+1$ cluster nodes (including its own). If all $t+1$ cluster nodes agree on the report, i.e. they send the same values, the cluster head sends the result to the base station.

For each node on the path from the cluster head to the base station, there exists a downstream node which generated a MAC for it. This node is either a cluster member (if the considered node is less than $t+1$ hops away from the cluster), or the downstream $t+1$-hops neighbor. When a report arrives at such an en-route node, it checks the MAC which was calculated for it.

If the node is able to verify the MAC, it removes it, generates its own

MAC using the pairwise key with its upstream $t+1$-hops neighbor on the path (if there exists one) and appends it to the report. Thus, every node verifies one MAC. If a MAC cannot be verified, the report is discarded and not forwarded further. Each message is in this case accompanied by $2 \cdot (t+1)$ MACs.

Note that accurate information on the up- and downstream neighbors is crucial to this scheme. To provide a node with this kind of knowledge, one must assume that routing paths do not change between the neighbor discovery phase and the actual sending of the report, which is a strong assumption for sensor networks.

At last, the base station checks the $t+1$ MACs which were generated by the cluster nodes. Thus, the base station can accurately detect any false report, even if this report could not be filtered out before.

Figure 5.1 illustrates a simplified version of this scheme for $t = 1$, where node 0 does not collect the MACs for its upstream neighbors, but generates them on its own. Moreover, the source does not add MACs for the sink. Here, node 0 wants to send a message to node 4.

A rigorous security analysis of interleaved hop-by-hop authentication can be found in the original paper [157]. Informally considered, to compute a fake report, the adversary would have to compromise an entire cluster, i.e., all $t+1$ cluster members.

However, the adversary can also try to change a legitimate report on its way to the base station. Consider a legitimate report r_1. The adversary wants to change it into r_2. The nodes compute MACs for their $t+1$-hop neighbors, and no intermediate node can verify the correctness of the MACs. Thus, if the one compromised node was to compute its MAC for r_2 instead on r_1, and there was a compromised node $t+1$ hops away, then the forgery would go unnoticed.

From this observation follows that if the adversary compromises $t+1$ nodes on the considered path such that the distance in hops between every two consecutive compromised nodes is not more than t, then the adversary can successfully change the report. Moreover, if the adversary compromised $t' \leq t$ nodes, it can make a fake report travel $t'(t+1)$ hops before it is discovered. Thus, a fake report travels at most $O(t^2)$ hops. However, the fake report would be discovered at the base station in any case, as the nodes on the path cannot change the MACs which were computed on r_1 by the cluster members.

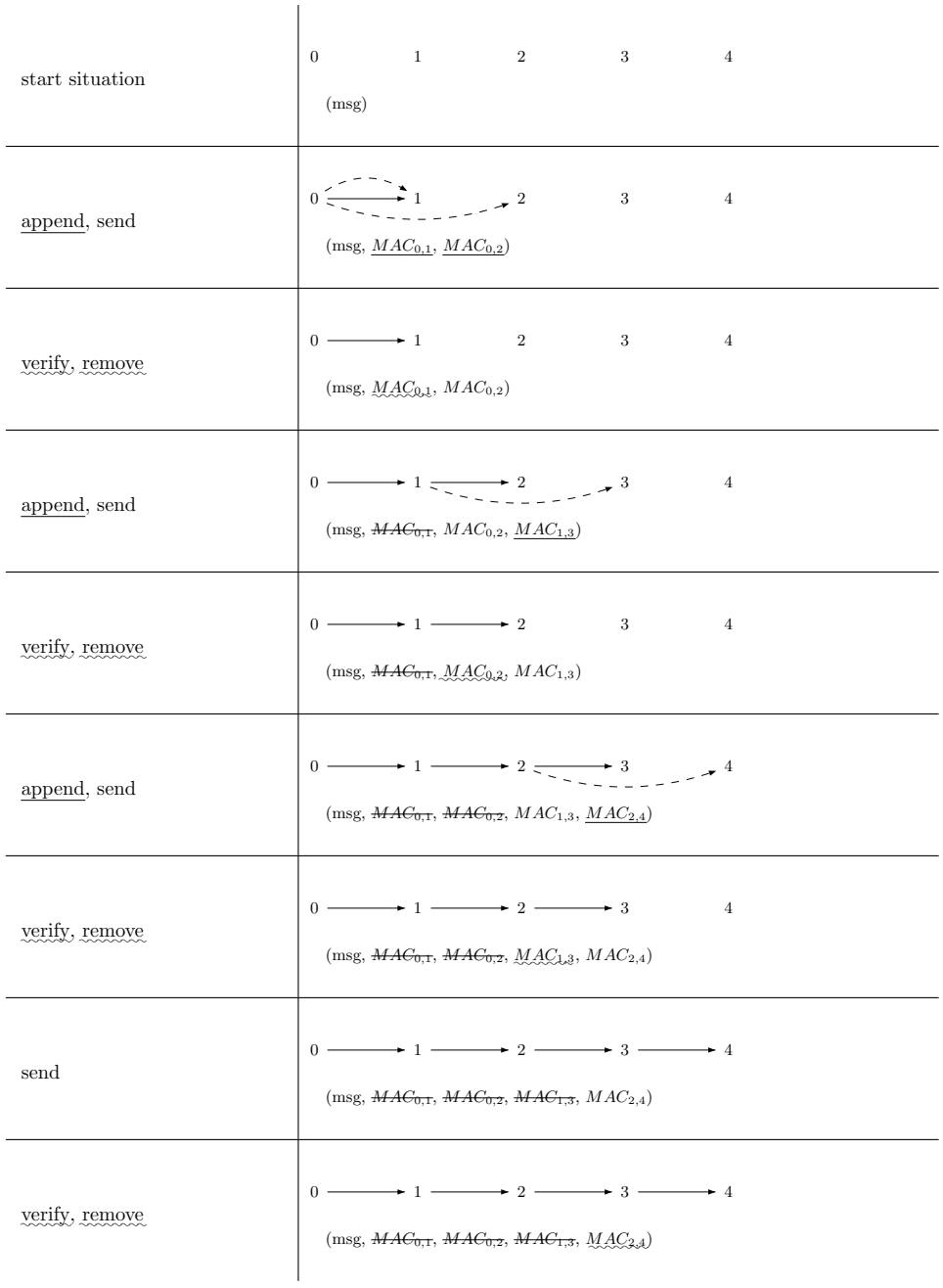


Figure 5.1: Illustration of the *interleaved hop-by-hop scheme* for $t = 1$. The full arrows represent sent messages whereas the dashed arrows indicate for which nodes MACs are generated.

5.3 System and Adversary Model

The generic system and adversary models were specified in Section 3.2. As presented there, we consider a large sensor network with a large number of users. The users can send queries to the network.

We consider an adversary with the goal of sending *arbitrary* queries to the network. The adversary can capture up to t sensor nodes, where t is the system parameter. We do not make any assumptions concerning the distribution of the captured nodes over the network.

That is, for a particular algorithm we always consider the worst case distribution. For example, all the captured nodes may be close to each other. On the other hand, the captured nodes may be distributed over the whole network according to some strategy. Moreover, the adversary can analyze network traffic in order to determine which nodes should be compromised. As stated in Section 3.2, this results in the following adversary, with the number of captured nodes being at most t:

$$\{\mathcal{A}_1(global, eavesdrop), \mathcal{A}_2(distributed, reprogramming)\}$$

Schemes presented in this chapter require an initialization phase where each node has to discover some of its x-hop neighbors for $x \leq t + 1$, i.e., some specific nodes that are $(t + 1)$ or less hops away from it. We assume that the attacker is not present during the initialization phase, and thus the neighbor discovery does not need to be secure. Secure neighbor discovery is an important and challenging problem, but it is out of scope of this work.

5.4 Interleaved Authenticated Querying

5.4.1 Introduction

In this section we present two t-robust schemes for authenticated querying and analyze their security and efficiency. Subsequently we discuss in Section 5.5 how to organize access control using the presented algorithms.

Both introduced schemes are based on the interleaved hop-by-hop authentication scheme [157] which is described in Section 5.2.2. The schemes have basically the same structure. Firstly, the base station and all sensor nodes have to be initialised. This includes gathering information about the neighborhood and establishing pairwise keys with other nodes. Secondly, the base station posts queries to the network and authenticates these queries. Sensor nodes which receive a query verify the corresponding authenticator

and modify it, i.e. they remove obsolete MACs, and add new ones. Finally, the sensor nodes forward both, the query and its authenticator.

The interleaved hop-by-hop authentication scheme was designed for point-to-point communication along a predefined path. The basic idea is that each node generates a MAC for its $t+1$-hop upstream neighbor on the path. The nodes along the path verify and remove the MACs when they receive the message, and add new MACs for their own $t+1$-hop upstream neighbors.

The first introduced scheme, called *basic interleaved authenticated querying* (see Section 5.4.2) is a straightforward extension of the interleaved hop-by-hop authentication scheme and in Section 5.4.3 is shown to be insecure and prohibitively expensive in terms of the authenticator size. Nevertheless, we present this scheme here for didactical reasons.

The second scheme, called *tree-based interleaved authenticated querying*, is much more practical than the basic one, and can successfully withstand the capture of up to t nodes.

The goal of authenticated querying is to flood the whole network with the query. As networks are unlikely to be a linear alignment of nodes, the messages have to spread in several directions. This means that each node has to add MACs not only for its upstream neighbor in one direction but in all possible directions. This dramatically increases either the number of messages which are sent or the size of the authentication information added to a message.

To illustrate this, consider a network with a uniform node distribution where the average number of $(t+1)$-hop neighbors is n. If a node wants to communicate with another node using the hop-by-hop scheme, the number of MACs in one message is $t+1 \in O(t)$ because each node appends one MAC for its $(t+1)$-hop upstream neighbor to the message, and deletes the MAC from its $(t+1)$-hop downstream neighbor. When using the same scheme for broadcast, the number of MACs becomes $n*(t+1)$, since each node has to append one MAC for each of its $(t+1)$-hop neighbors. Note that when t increases, n increases as well. Thus, n can also be seen as a function of t which means that $n*(t+1)$ would be more likely in $O(t^2)$. This leads to a rapidly increasing message size resulting in high transmission costs.

5.4.2 Basic Interleaved Authenticated Querying

We consider a straightforward application of interleaved hop-by-hop authentication to authenticated querying. Roughly speaking, instead of generating a single MAC for its upstream $(t+1)$-hop neighbor on the path to the base station, each node now generates MACs for all of its $(t+1)$-hop neighbors.

That is to say, not only a single path is considered but all possible ones.

The $(t+1)$-hop neighbors of a node are all nodes which are exactly $t+1$ hops away, i.e., which can be reached in $t+1$ steps but not in fewer.

Note that the node will still have to verify only one MAC, as the received query will arrive on a particular path where there is only one $t+1$-hop neighbor for the verifying node. However, the verifying node will have to remove all other MACs generated by this neighbor.

During the initialization phase the base station has to discover all nodes it can reach within $t+1$ hops. For all these nodes it will have to generate MACs and append them to the query. Thus, the base station also has to establish pairwise keys with all these nodes. Note that many schemes assume that the base station establishes keys with all sensor nodes which is not necessary here.

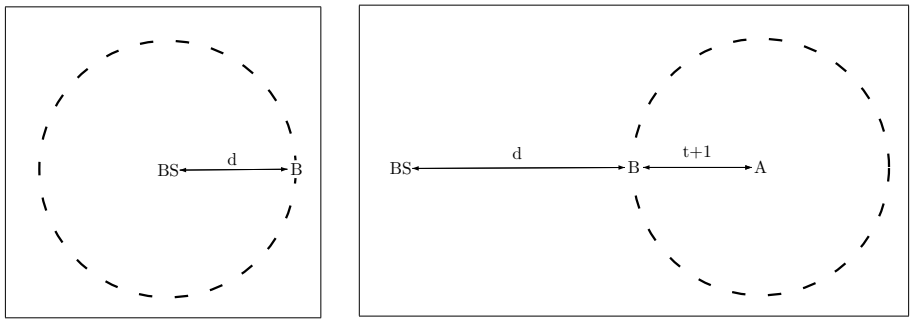
Sensor nodes have to discover their $(t+1)$-hop neighbors, as they will have to generate MACs for them. Therefore, they have to establish a pairwise key with each of these neighbors using, e.g., the polynomial key-predistribution scheme described in Section 5.2.1. Moreover, the MAC on the query which the nodes will have to verify will also be generated either by one their $(t+1)$-hop neighbors or by the base station, in case the node is not more than $t+1$ hops away from it.

Both, base station and sensor nodes, store the discovered neighbors in a list referred to as *ownneighbors*.

The sensor nodes also have to discover some additional neighbors. To illustrate why, imagine node B receives an authenticator. Such an authenticator contains the IDs of the nodes which generated the MACs, and the MACs themselves. Let A be the first of these nodes. Then, according to the considered schemes, A is a $t+1$-hop neighbor of B and it has generated a MAC for B. All MACs generated by A can be removed after the MAC for B was verified, as B would forward the message to nodes that are $t+2$-hop neighbors of A and in consequence, cannot do anything with MACs generated by A. Thus, node B has to discover the IDs of the nodes for which A generated MACs in order to remove these MACs. Alternatively, each MAC can be accompanied by two node IDs: the node that generated the MAC, and the node for which the MAC is destined. This makes the messages even more bulky, however, so we consider the topology-based approach instead.

Here two cases can be distinguished. Let d be the distance (measured in hops) between B and the base station. If d is less than or equal to $t+1$, every query which reaches B is authenticated by the base station, since the base station generates the MACs which are used to authenticate the query within the first $t+1$ hops. In this case it suffices for B to discover all nodes

which have the same distance to the base station as itself, i.e. which are d hops away from the base station. These are the nodes for which the MACs generated by the base station are in the same part of the authenticator. B needs to know these nodes in order to locate the MAC intended for itself, and to remove the remaining MACs. Figure 5.2(a) illustrates this case. Here the dashed line represents the hop on which the discovered neighbors are.



(a) Case 1: $d \leq t+1$ (b) Case 2: $d > t+1$

Figure 5.2: Discovery of additional nodes for the node B.

If B is more than $t+1$ hops away from the base station, then each query which reaches B is authenticated by one of B's $(t+1)$-hop neighbors. Let A be this neighbor (as before). B needs to know the other $(t+1)$-hop neighbors of A in order to locate the MAC destined for itself and to remove all obsolete MACs from the authenticator. As B does not know from which direction the query might arrive, it needs to discover all $(t+1)$-hop neighbors of all possible A's, i.e., all of its own $(t+1)$-hop neighbors. These neighbors are stored in an array of list of node IDs, referred to as *neighborsOf* in the following. Figure 5.2(b) illustrates this second case. In this simple example there is only one possible A. The pseudo-code notation for these initialisation procedures are shown in Algorithm 5.4.1 and Algorithm 5.4.2.

Algorithm 5.4.1: Initialise (Base Station; Basic Scheme)

data: *ownneighbors* $\leftarrow$ *discover* $1, 2, ..., (t+1)$*-hop neighbors;*
foreach *node* $\in$ *ownneighbors* **do**
  establish a pairwise key with *node*;

Every query has a unique identifier, called *query ID* or *qid*. Each time a query is posted to the network, the base station has to compose an authenticator for this query.

Algorithm 5.4.2: Initialise (Sensor Node; Basic Scheme)

```
data: ownneighbors ← discover (t + 1)-hop neighbors;
data: d ← distance to the base station;

foreach node ∈ ownneighbors do
    establish a pairwise key with node;

switch d do
    case d ≤ t + 1 hops
        data: neighborsOf[basestation] ← discover d-hop neighbors of
        the base station;
    case d > t + 1 hops
        foreach node ∈ ownneighbors do
            data: neighborsOf[node] ← discover (t + 1)-hop neighbors
            of node;
```

qid	*nids*[1]	...	*nids*[$t+1$]	MACs from *nids*[1]	...	MACs from *nids*[$t+1$]

Figure 5.3: Structure of an authenticator.

An authenticator consists of the following components. Firstly, it contains the *qid*. Secondly, it contains an array of node IDs referred to as *nids* in the following. This array comprises $t+1$ node IDs indicating which $t+1$ nodes generated the MACs included in the authenticator. Finally, the authenticator also contains the MACs generated by the nodes mentioned in *nids*. This structure is illustrated by Figure 5.3.

When the base station posts a query, all entries in *nids* are set to the ID of the base station because the base station generates all MACs for the first $t+1$ hops. See Algorithm 5.4.3 for the pseudo-code notation of this procedure.

When a node first receives a query, it has to verify the authenticator. To do this, it invokes the procedure `HandleAuthenticator` (see Algorithm 5.4.4).

This procedure first checks if this query was handled before. If so, the message is discarded. The receiver also checks whether all node IDs in the array are either different or the ID of a base station. Without this check an adversary would be able to fake a message with a single compromised node. If this check yields a positive result, the receiving node reads the first field of the *nids* array in the authenticator and looks up the neighbors for this

Algorithm 5.4.3: StartQuery (Base Station; Basic Scheme)

input: a query with a unique identifier (*qid*)
`/* an array containing t+1 node IDs */`
data: *nids;*
`/* a list of MACs */`
data: *macs;*

set all fields in *nids* to the own ID;

foreach *node* $\in$ *ownneighbors* **do**
  generate MAC with pairwise key shared with *node*;
  append this MAC to *macs*;
data: *authenticator* $\leftarrow$ *qid* + *nids* + *macs;*

broadcast *query*||*authenticator*;

node. Note that these neighbors were already discovered in the initialisation phase.

For each of these neighbors there is a MAC in the authenticator destined for it. The neighbors and the corresponding MACs are now processed one after another. That is to say, the first MAC in *macs* is destined for the first neighbor, the second MAC for the second neighbor and so on. If the currently considered neighbor has the same ID as the receiver, i.e., the current MAC is intended for the receiving node, the node tries to verify this MAC and removes it afterwards. Otherwise it just removes the MAC. If a MAC could not be verified, the query is discarded and as it is marked as handled, it will not be considered anymore. This procedure prevents an adversary from performing reply attacks.

After the process of verifying and removing MACs is finished, the node generates new MACs, one for each of its previously discovered $t + 1$-hop neighbors, and appends these MACs to the authenticator. Since all MACs which were generated by the first node in the *nids* array were removed and additional MACs were added, *nids* has to be updated as well. By shifting all entries in the array one field to the left the first entry is dropped and the last entry becomes empty. This last entry has to be set to the node's own ID, i.e. to the ID of the node which has just processed the authenticator.

Finally, the query and the modified authenticator are broadcast into the network. Here, we assume flooding as the method for query dissemination.

Algorithm 5.4.4: HandleAuthenticator (Sensor Node; Basic Scheme)

```
input: query||authenticator

data: qid ← authenticator.qid;
data: nids ← authenticator.nids;
data: macs ← authenticator.macs;
data: verified = FALSE;

if qid was handled before then
    ignore the message;
else
    mark qid as handled;
if values in nids are invalid then
    discard query;
else
    data: node ← first entry in nids;
    foreach x ∈ neighborsOf[node] do
        if x = my ID then
            verify the current MAC;
            if this MAC could not be verified then
                discard query;
            else
                verified = TRUE;
        remove first MAC from macs;
    if verified = FALSE then          /* no MAC was verified */
        discard query;
    else                              /* a MAC could be verified */
        foreach node ∈ ownneighbors do
            generate MAC with pairwise key shared with node;
            append this MAC to macs;
        shift all entries in nids one field to the left;
        set the last field of nids to the own ID;
        data: authenticator ← qid + nids + macs;
        broadcast query||authenticator;
```

5.4.3 Analysis of the Basic Interleaved Scheme

We now analyze security and performance of the basic t-robust authenticated querying algorithm.

5.4.3.1 Security

The security analysis of hop-by-hop interleaved authentication [157] showed that an adversary can forge a report en-route only if it captured more than t nodes on a path, and if the distance (in hops) between every two consecutive nodes is not more than t. Moreover, all reports which were faked en-route will be detected at the latest at the base station. The only feasible way to forge a report undetectably is to compromise a whole cluster ($t + 1$ nodes) which can then fabricate a fake report.

In our scheme, if the adversary compromised $t + 1$ nodes that lie consecutively on some path, the faked query can spread to almost all nodes that are connected to any of the compromised nodes.

To see this, consider a shortest-path spanning tree rooted at an arbitrary non-compromised node s. If there is a branch in this tree which contains the compromised path, then the faked query will arrive to s via this path.

On the other hand, if only a part of the compromised path lies on a branch of the spanning tree, the adversary has to tamper with the authenticators. The node s will accept a query if one of its $(t + 1)$-hop neighbors computed a MAC on it using their shared key. If there is a node q such that q accepted the fake query, and q is a $(t + 1)$-hop neighbor of s, then q generated a MAC for s as it forwarded the query to its neighbors. This MAC can be used to convince s that the query is genuine. On the other hand, if q does not exists, there is no possibility for the adversary to forge a query for s.

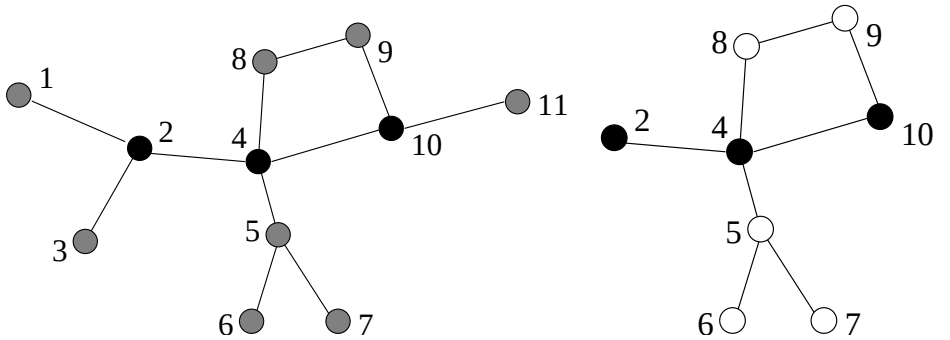
Although there is no guarantee that for all nodes s there is such a node q, this may be true in sufficiently large networks in many cases. The query will firstly spread over all nodes which have the compromised path in their shortest-path spanning trees. Then the adversary can analyze the messages produced by non-compromised nodes which accepted the fake query, and use the produced MACs.

To summarize, the impact of a faked query on the network is highly topology dependent. In the worst case, the fake query can spread over the whole network.

We illustrate the above reasoning using Figure 5.4. Here, $t = 2$ and the path $(2, 4, 10)$ is compromised. In the left picture, the non-compromised

nodes 1, 3 and 11 have the compromised path in their shortest-path spanning tree, so the fake query reach these nodes quite naturally. Then node 11 generates MACs for its 3-hop neighbors 5 and 8, such that the fake query will be accepted by these nodes. Analogously, the query can spread to all other nodes in the network.

In the right picture, none of the non-compromised nodes have the compromised path in their shortest-path spanning tree, and therefore, the fake query cannot reach any of these nodes.



(a) The fake query spreads to all nodes.

(b) The fake query does not reach any non-compromised node.

Figure 5.4: Impact of the compromised path $(2, 4, 10)$ on the network, $t = 2$.

Now assume that the adversary controls $\hat{t} < t + 1$ nodes. An honest node s accepts a query if there is a MAC computed by one of its $(t+1)$-hop neighbors in the authenticator. Thus, if the query is fake, then either a $(t+1)$-hop neighbor of s is compromised, or this neighbor accepted the fake query previously. We now show that in certain circumstances even a couple of compromised nodes can lead to a serious attack.

Consider Figure 5.5. Here, $t = 2$ and nodes 1 and 2 are compromised. Node 1 shares secret keys with its 3-hop neighbors 3, 4 and 5. These neighbors constitute a 3-hop path in the sensor network.

The attack runs as follows. Assume that the adversary wants to send an arbitrary query q. The adversary constructs the authenticator for q using the key shared between nodes 1 and 3. All other MACs in the authenticator can be bogus. The adversary gives the faked query to node 2 which sends it to node 3. Node 3 verifies the authenticator, i.e., it verifies the MAC computed by node 1, and modifies the authenticator according to the algorithm. Among other things, it computes MACs for nodes 6 and 7. The adversary intercepts this authenticator using node 2 and stores the computed MACs

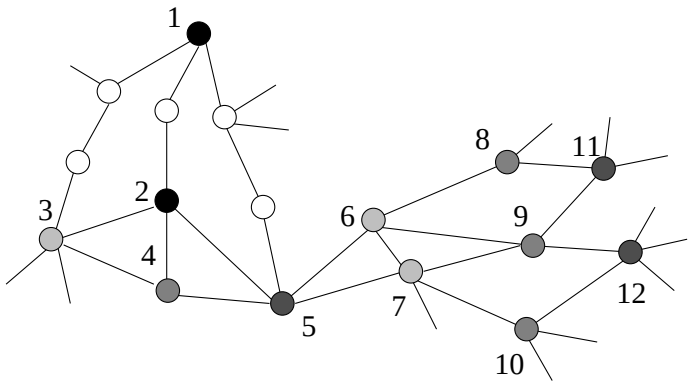


Figure 5.5: Nodes 1 and 2 are captured, $t = 2$. The faked query can reach non-compromised paths $(6, 8, 11)$, $(7, 9, 12)$ and $(7, 10, 12)$ if the adversary tampers with the authenticators generated by non-compromised nodes on the path $(3, 4, 5)$.

for future use.

Using the same technique, the adversary obtains MACs computed with keys shared between nodes 4 and its 3-hop neighbors 8, 9, and 10, and between 5 and its 3-hop neighbors 11 and 12.

In this way, the adversary can make $t + 1$ consecutive nodes on at least one path to accept the fake query. In the example, $t + 1 = 3$, and some of the the paths are $(6, 8, 11)$, $(7, 9, 12)$ and $(7, 10, 12)$. As we have seen above, the fake query may spread over a considerable part of the network in this case.

We conclude that the security of the above straightforward extension of the interleaved hop-by-hop authentication method to authenticated querying is highly dependent of the network topology. In the worst case, one or two captured nodes can suffice for a very serious attack which makes a large part of the network accept an arbitrary fake query. As in our system model we do not make assumptions about network topology, we have to consider the above scheme as highly insecure against node capture.

We note that the above attack can also be applied to the original scheme [157] for report authentication in case the reports are sent to the base station along overlapping paths. Thus, care should be taken when employing interleaved hop-by-hop report authentication in large networks with multiple reporting clusters.

5.4.3.2 Performance

Although the scheme presented in the current section was shown to be insecure, we nevertheless analyze its performance in order to compare it to more secure solutions presented further.

The above straightforward approach does not perform very well. The main reason is the enormous size of the authenticators which have to be sent. As this algorithm does not use any sense of direction, the authentication information for all $(t+1)$-hop neighbors has to be sent to every node.

Let $|nid|_{byte}$ denote the storage required for a node ID. Then the *nids* array consumes $(t+1) * |nid|_{byte}$ bytes. Further, let $|mac|_{byte}$ be the size of a MAC, and $|qid|_{byte}$ the size of a query ID. The number of MACs attached by each node strongly depends on the underlying topology.

As an example, we assume a "diamond" grid as presented in Figure 5.6. In this topology, the number of n-hop neighbors of an "inner" node is $6n$. In a large grid, each node has roughly the same number of $(t+1)$-hop neighbors which is $6(t+1)$. This means that each node appends $6(t+1)$ MACs to an authenticator. As the authenticator contains information from exactly $t+1$ nodes, the upper bound for the size of the authenticator in the basic scheme is

$$auth_{basic} = |qid|_{byte} + (t+1) * |nid|_{byte} + (t+1) * 6(t+1) * |mac|_{byte} \tag{5.1}$$

Assume the following sets of parameters:

- $t = 2$
- $|qid|_{byte} = 2$
- $|nid|_{byte} = 2$
- $|mac|_{byte} = 4$

With these parameters each node would have to send 224 bytes of authentication information:

$$\begin{aligned} auth_{simple} &= |qid|_{byte} + (t+1) * |nid|_{byte} + (t+1)^2 * 6 * |mac|_{byte} \\ &= 2 + 3 * 2 + 9 * 6 * 4 = 224 \end{aligned}$$

However, packet size in sensor networks is usually assumed to be in order of tens of bytes. Thus, apart from being insecure, the basic scheme is also impractical. Therefore, in the next section we develop another scheme for interleaving authenticated querying, and show further show that this scheme is t-robust and its efficiency is optimal.

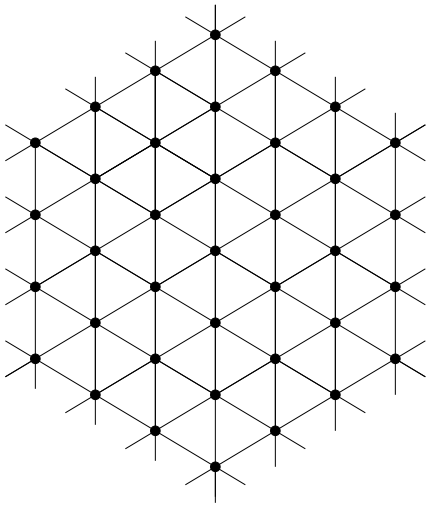


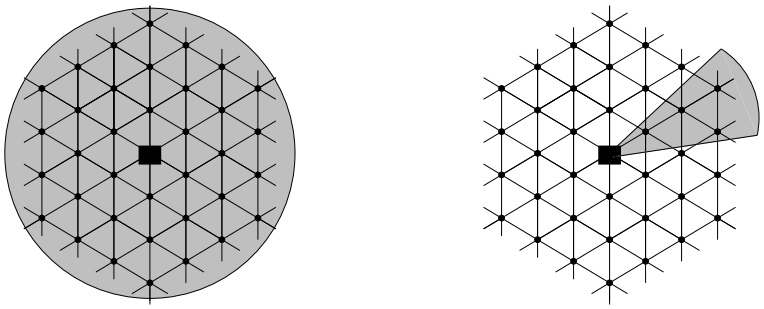
Figure 5.6: Diamond grid topology.

5.4.4 Tree-based Interleaved Authenticated Querying

With an increasing message size the chance of successfully receiving this message decreases. Thus, it would be nice to have shorter messages which are sent into a certain direction. The idea is to use several unicast messages which only contain the authentication information relevant to a restricted area, instead of one large broadcast message that contains all information. Figure 5.7 illustrates this approach.

The amount of data sent by the base station remains the same. In fact it might even increase a bit since sending several messages involves a greater communication overhead, caused by the transport protocol. Hence, there is no improvement at the base station. Despite this fact, energy can be saved in the whole network. The reason is that messages sent within a certain area (as indicated by Figure 5.7(b)) are shorter than the broadcast message in the basic scheme because they contain less information. More precisely, messages in the broadcast scheme contain the complete information required in all direction from the sender whereas each unicast message contains only a small part of this information. Hence, all sensor nodes in this area have to transfer less data. Besides this advantage, shorter messages also increase the chances of receiving these messages. Therefore, the number of nodes that can successfully authenticate a query should also increase.

Another drawback of the basic approach is that multiple MACs are generated for each node. This is because nodes do not keep track of the path



(a) broadcast message (b) unicast message

Figure 5.7: Area for which the authentication information is included in a message, the base station is located in the middle of the grid.

a message has already travelled. Figure 5.8 shows an example for $t = 1$. In this example node A generates MACs for all its 2-hop neighbours. First, B receives the message sent by A, removes some MACs, and adds new ones for its own 2-hop neighbours. Next, C receives the message, removes some MACs, and appends new MACs for its 2-hop neighbours including a MAC for A. But the authenticator from C will never reach A because A already processed this message. Nevertheless, C will generate a MAC for A and so will all of A's 2-hop neighbours. Obviously this should be prevented. The tree-based scheme presented in the following realizes the above ideas.

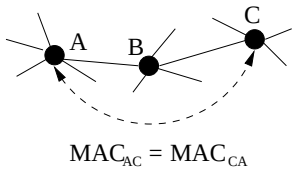
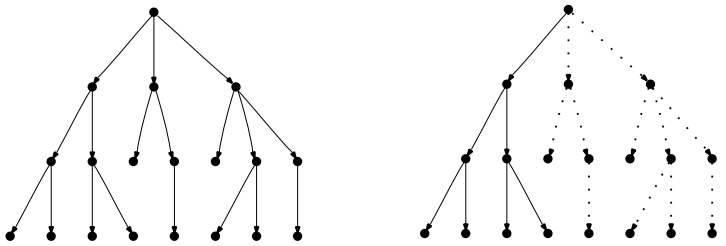


Figure 5.8: Example for the redundant calculation of MACs in the basic scheme. Here $t = 1$, node A generates a MAC for C, and C generates a MAC for A.

In order to provide nodes with a sense of direction, nodes do not only need to know which neighbours they have but also how they are connected to these neighbours. This means that all nodes need some routing information. The simplest approach to obtain this kind of information is to construct a spanning tree of the network with the base station as root node. The base station can now send a unicast message to each of its direct children. Each

of these unicast messages contains the authentication information required in this particular subtree, instead of the information for the whole tree as before (see Figure 5.9).



(a) the whole spanning tree (b) a considered subtree

Figure 5.9: Spanning tree of a network with the base station as root.

Algorithm 5.4.5 describes the initialisation phase for the base station. First, a shortest-path spanning tree of the network is constructed and then keys are established with all nodes on the first $t + 1$ layers of the tree, i.e. with all nodes within the first $t + 1$ hops from the base station.

Algorithm 5.4.5: Initialise (Base Station; Tree Scheme)

data: *tree* ← *discover a shortest-path spanning tree of the network;*

establish pairwise keys with all nodes on the first $(t+1)$ layers of *tree*;

Sensor nodes need to know a part of the spanning tree. Specifically, each node needs to know the $t+1$ layers below the own position and its $(t+1)$st predecessor. Note that other children of the predecessor are not required. All other branches from the spanning tree can be pruned.

Depending on the used topology discovery protocol, this information might be obtained during the construction of the spanning tree. If this is not possible, sensor nodes can request the required part of the spanning tree from the base station.

In the next step, each node has to establish pairwise keys with all nodes on the $(t+1)$st layer below itself. During this process, each node would also have automatically established pairwise keys with its $(t + 1)$st predecessor in the tree. Algorithm 5.4.6 summarises this procedure.

When a query is started, the base station generates an authenticator

Algorithm 5.4.6: Initialise (Sensor Node; Tree Scheme)

request the required subtree;
establish pairwise keys with all nodes which are exactly $t + 1$ layers below;

for each of its direct children. Each of these authenticators only contains the authentication information required in the subtree beneath the selected child. The base station generates MACs for each node in the subtree and appends these MACs to the authenticator (see Algorithm 5.4.7). Note that the *nids* array is not required in this scheme because each node has a well-defined position in the tree and thus knows exactly which nodes processed the authenticator before.

Algorithm 5.4.7: StartQuery (Base Station; Tree Scheme)

```
input: a query with a unique identifier (qid)
/* a list of MACs                                          */
data: macs;
foreach direct child in the spanning tree do
    consider the subtree below this child;
    for n = 1 to t+1 do
        foreach node on the n-th layer do
            generate a MAC with the key shared with node;
            append this MAC to macs;
    data: authenticator ← qid + macs;
    send authenticator||query to the currently considered direct child;
```

As in the basic scheme, the sensor node receiving a query firstly checks the processing status of the query. If the query was already handled before, then it is dropped. Otherwise, the query is marked as handled and the authenticator is processed.

Unlike the basic scheme, sensor nodes do not have to look up the ID of the node that generated the MACs in the *nids* array because they already know which node generated the MACs. If a node is on one of the first $t + 1$ layers of the tree, then the MAC destined for it was generated by the base station. Otherwise, the MAC was generated by its $(t + 1)$st predecessor. After verifying the MAC, the node generates new MACs for all nodes on the $(t + 1)$st layer below itself. These MACs are appended to the query.

The difference is that the authenticator is not broadcast as a whole but split up into several parts which are unicast. Each child gets its own authenticator containing just the MACs required in the subtree below this child. Algorithm 5.4.8 shows the routine for handling an incoming query.

Algorithm 5.4.8: HandleAuthenticator (Sensor Node; Tree Scheme)

```
input: a query and its authenticator
data: qid ← authenticator.qid;
data: macs ← authenticator.macs;
if qid was handled before then
    ignore the message;
else
    mark qid as handled;
verify the first MAC;
if this MAC could not be verified then
    discard query;
else
    foreach node ∈ ownNeighbours do
        generate a MAC with the key shared with node;
        append this MAC to macs;
    foreach child ∈ children do
        data: macs ← select all MACs from the authenticator
        required in the branch below child;
        data: authenticator ← qid + macs;
        send authenticator||query to child;
```

5.4.5 Analysis of the Tree-Based Scheme

5.4.5.1 Security

The insecurity of the first scheme presented in this chapter (see Section 5.4.3.1) was based on the fact that the sensor nodes had a limited knowledge about the query dissemination in the network. Therefore, the nodes accepted the query as soon as it was authenticated by an arbitrary $(t+1)$-hop neighbor of the node.

In the scheme presented in the current section, this is not the case. If the adversary compromises $t+1$ nodes that lie consecutively on a path in the

spanning tree, then the last node on this path will be able to disseminate the fake query in the subtree rooted at it. As in the previous case, the impact of the fake query is highly topology-dependent. In particular, it depends very much on the structure of the spanning tree rooted at the last node.

If the adversary compromised $\hat{t} \leq$ nodes, then the impact of the fake query is localized to the corresponding $\hat{t}$-level subtree of the network's spanning tree.

We now consider the denial-of-service (DoS) opportunities in the tree-based scheme. Firstly, in our scheme the nodes blacklist queries that arrive with an incorrect authenticator. Thus, if the adversary sends incorrect queries with future query IDs, then the corresponding correct queries will not be processed by the affected nodes. The impact of these queries is localized, as the nodes do not forward the queries. However, this means that the whole subtree rooted at the affected node will not receive the correct query. it is not quite clear how to defend against this attack. Most probably, the defense would require some methods that are outside of the protocol design, such as an intrusion detection system [90].

We also note that the tree-based scheme is highly susceptible to node crashes and link failures, that is, to the *disturbing* adversary. One failure means that the whole subtree rooted at the attacked node cannot receive the query. Appropriate countermeasures include monitoring of neighbors for link and crash failures and local repair, such as assigning a new root to the affected subtree. We are, however, unaware of any such protocols that work in the presence of node captures.

5.4.5.2 Performance

As mentioned above, messages sent in the tree-based scheme are considerably smaller than messages in the basic scheme. Shorter messages are favourable because they consume less energy. Nevertheless, each node has to send several messages in this scheme, one for each of its children to be more specific. An increasing number of messages causes an extra communication overhead.

In order to compare the tree scheme with the basic scheme, one has to estimate the number of MACs in each message. The number of MACs in each message is exactly the number of nodes in a $(t+1)$-layered subtree of the network's spanning tree. Unfortunately, the average number of $(t+1)$-hop neighbors and the average degree of the spanning tree cannot be converted into each other in a straightforward manner. The following paragraphs first describe how one can achieve a rough approximation of the number of the

MACs in the authenticator and then illustrates this result using a concrete example.

Approximation. According to our algorithm, the network is organized as a shortest-path spanning tree rooted at the base station. We assume that each subtree rooted at the children of the base station is a tree where each node has d children. Actually, such a tree does not exist in the most cases, thus d should be considered as an approximation. For example, in Figure 5.10 the children of the base station have 12 children, and these have 18 children in the next layer. Thus, each node has on average $d = \frac{1}{2}\left(\frac{12}{6} + \frac{18}{12}\right) = 1.75$ children.

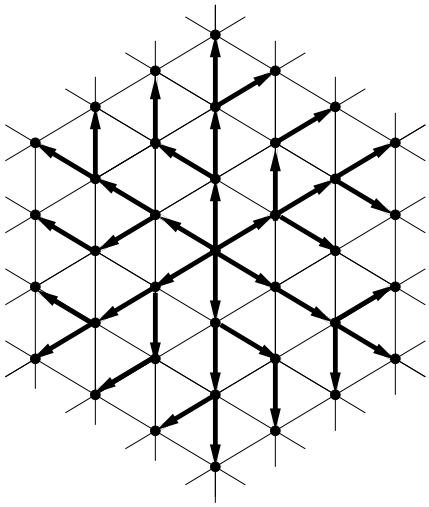


Figure 5.10: Example of a spanning tree for the diamond grid.

To generalize the above reasoning, consider a network of diameter $2r$ (in hops) organized in the diamond structure such as in Figure 5.10 with the base station in the center. Then there are $6i$ nodes in the ith level of the spanning tree which starts at the base station. The depth of the spanning tree is r.

We compute the average number of node's children in the spanning tree as follows:

$$d = \frac{1}{r-1}\sum_{i=1}^{r-1}\frac{6(i+1)}{6i} = 1 + \frac{1}{r-1}\sum_{i=1}^{r-1}\frac{1}{i} \tag{5.2}$$

The average degree of the spanning tree goes to 1 with the growth of the network diameter. This can actually be shown for any network with uniformly distributed nodes.

We now consider how many MACs are included in the authenticator of the tree-based scheme. The incoming message for each node contains a MAC computed by its $(t+1)$-hop predecessor, or by the base station, and MACs for all nodes in the t-layer subtree rooted at the considered node. The number of these MACs can therefore be computed as

$$\sum_{i=0}^{t} d^i = \frac{d^{t+1}-1}{d-1}. \tag{5.3}$$

Finally, the number of bytes in the authenticator can be computed as

$$auth_{tree} = |qid|_{byte} + |mac|_{byte} * \frac{d^{t+1}-1}{d-1}. \tag{5.4}$$

Example. We now compute the authenticator size for the tree scheme using the same set of parameters as in Section 5.4.3.2 where we considered the performance of the basic scheme. We consider a network organized as a diamond grid similar to the grid in Figure 5.6 with the base station in the middle and the diameter of 20 hops. The other parameters remain as in Section 5.4.3.2:

- $t = 2$
- $|qid|_{byte} = 2$
- $|mac|_{byte} = 4$

This means that the distance between any node and the base station is at most 10 hops. Then according to Formula 5.2 $d = 1.3$ and the number of MACs in the authenticator is 4 on average according to Formula 5.3.

An authenticator message in the tree-based scheme does not contain any node IDs, but just the query ID and the above number of MACs. Then an authenticator contains approximately the following number of bytes:

$$auth_{tree} = |qid|_{byte} + |mac|_{byte} * 4 = 16$$

Thus, in the tree-based scheme the size of the authentication information in this example is 16 bytes instead of 224 bytes in the basic scheme.

Actually, when using the interleaved hop-by-hop authentication to flood a network, the tree-based scheme is the most efficient approach. The following section argues why this scheme cannot be further improved.

5.4.6 Optimality of the Tree-Based Scheme.

In order to show that the tree-based scheme is indeed the most efficient one for interleaved authentication, one needs to show three points:

1. no MAC is redundant
2. no path can be omitted
3. the data cannot be further aggregated

Regarding the first point: Every node appears only once in the spanning tree. The base station generates one MAC for each node on the first $t+1$ layers of this tree. Each following node generates one MAC for each of its $t+1$ descendants. As the $t+1$ descendants are unique for each node, there is exactly one MAC generated for each node. This means that if any MAC was omitted, one node would not be able to authenticate the query. Hence, no MAC is redundant.

Regarding the second point: In a spanning tree there is exactly one path from the root to each descendant. Thus, not considering a path implies that at least one node (and the whole subtree beneath this node) would not be able to authenticate the query.

Regarding the third point: A query is authenticated along a path consisting of at most $t+1$ steps or parts. If the root node would send a unique authenticator to each of its descendants, some parts of these paths are used several times. One can avoid this overhead by multiplexing all authentication information which uses the same first step into one message. After the message has travelled one step, the information could be split up again and all information which uses the same next step is put into a single message. This way, each part of each path is authenticated only once and this is exactly what the tree-based scheme does. The authentication information cannot be further aggregated because the spanning tree consists of shortest paths.

5.4.7 Comparison to Related Work

We now consider the properties of the basic and the tree-based schemes for interleaved authenticated querying.

As shown in Section 5.4.2, the basic scheme is insecure and inefficient. However, the tree-based interleaved scheme presented subsequently is secure in presence of up to t captured nodes and is much more efficient for the small values of t, see Figure 5.11.

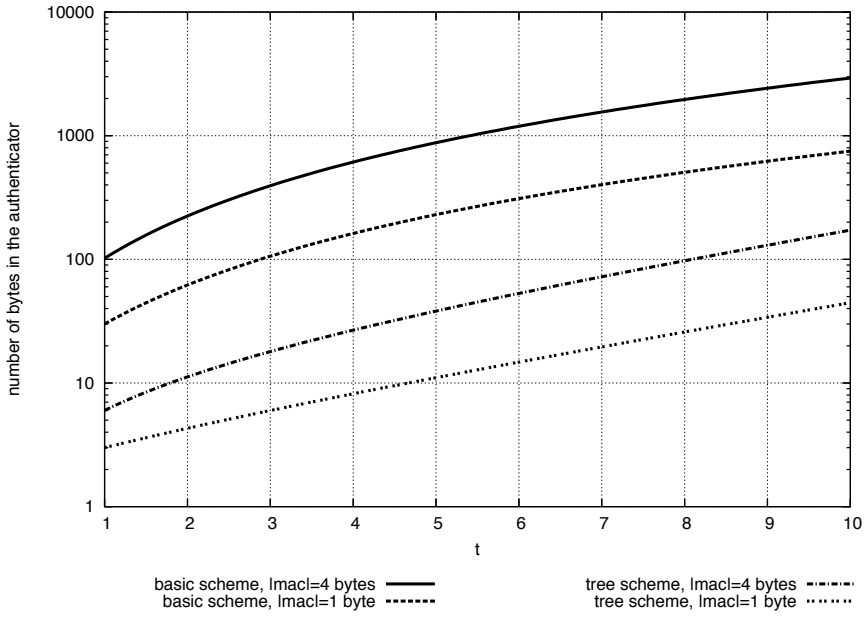


Figure 5.11: The number of bytes in the authenticator for the basic vs. tree-based interleaved authenticated querying according to Formulas 5.1 and 5.4 depending on the maximal number of captured nodes t for the diamond grid topology (Figure 5.6).

It can be seen that for $t = 10$, even the most efficient tree-based scheme with the very small MAC size of 1 byte still requires the authenticator size of more than 40 bytes. This authenticator size may still be affordable for sensor networks, considering that the size of an ECC (elliptic curve cryptography) digital signature is also around 40 bytes. We note, however, that the MAC

size of 1 byte is considered insecure, and the size of at least 4 bytes is recommended [106]. However, for this MAC size the authenticator gets prohibitively large (around 170 bytes) even for the tree-based scheme.

Nevertheless, for the small values of t, the tree-based scheme can be used. For example, for $t = 3$ the size of the authenticator is 18 bytes for 4-byte MACs and 6 bytes for 1-byte MACs.

We now consider which goals of the broadcast authentication presented in Section 4.6.1 on page 56 are satisfied or not satisfied by the tree-based interleaved authenticated querying, and compare our scheme to related work on broadcast authentication (see Section 4.6).

Our scheme incurs small computation overhead for small values of t in uniformly distributed networks, as each node needs to verify only one MAC, and to compute MACs for the nodes that are $t+1$ tree levels below them in a tree with the average number of children going to 1 in networks of large diameter. For example, if the average number of children is 1.3, then for $t = 3$ each node computes $1.3^4 \approx 3$ MACs on average.

The communication overhead can be computed as the number of MACs attached to the query. As we have shown above, the authenticator size for $t = 3$ and $d = 1.3$ can vary from 6 to 18 bytes. This is quite acceptable considering that the scheme does not require time synchronization, provides immediate authentication, and can authenticate an unbounded number of messages sent at irregular times after the first initialization. Thus, for small values of t, our scheme compares favorably with the existing broadcast authentication solutions.

We note that our scheme requires the maintenance of the spanning tree with reliable links in the network and incurs in this sense additional costs similar to these of n-LQA [124] and the authenticated broadcast schemes based on hierarchical aggregation algorithms [36].

5.4.8 Summary

We investigated how to extend the scheme for interleaved hop-by-hop report authentication [157] to be used for authenticated querying. The first developed scheme (Section 5.4.2) was presented for didactical reasons, as it turned out to be insecure and inefficient.

However, the tree-based interleaved scheme presented subsequently is secure in presence of up to t captured nodes and exhibit desirable properties for the small values of t.

In the next section we utilize the tree-based scheme for access control.

5.5 t-Robust Access Control

5.5.1 t-Robust Access Control via Base Station

As described in Section 3.5.1, access control (AC) at the base station can be organized in two phases:

1. The user logs in to the base station using conventional access control methods.

2. The base station sends user's query into the sensor network and authenticates it using some method for authenticated querying.

In the direct AC, the user connects to the base station using some external network e.g., the Internet, or by directly approaching it with a user device. In this case, t-robust access control at the base station can be organized in the straightforward manner. After the user logged into the base station, user's queries are authenticated by the base station using the tree-based scheme for interleaved authenticated querying.

For the remote t-robust AC, the user is located in the sensor field and uses the surrounding sensor nodes for remote communication with the base station. In this case, the user has to authenticate itself to at least $t + 1$ sensor nodes. These nodes forward user's queries to the base station using the interleaved hop-by-hop report authentication [157]. The base station then sends the queries into the network using the tree-based scheme for authenticated querying.

We note that in order to use interleaved hop-by-hop report authentication, the network should be organized in clusters, and each cluster should know a predefined path to the base station. In the tree-based scheme, the nodes can use the path in the spanning tree from the cluster head to the base station. In this case, each node in the cluster should be assigned a node on the path (within the first $t + 1$ hops) for which it generates the MAC on the user's query.

5.5.2 t-Robust Decentralized Access Control

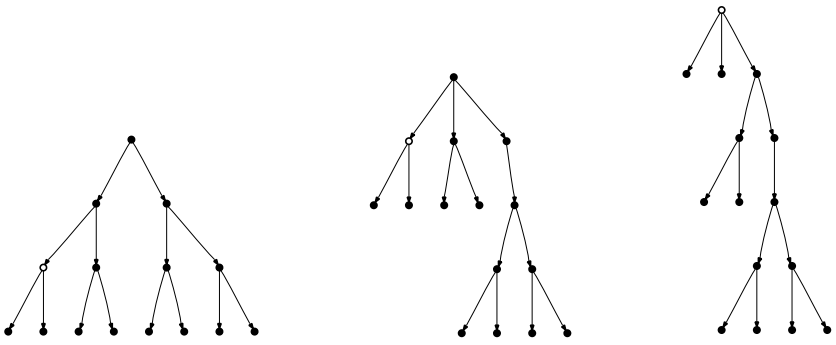
In the previous section we considered t-robust authenticated querying via the base station. We now consider how the tree-based scheme has to be altered in order to authenticate user's queries without the participation of the base station.

In the tree-based scheme the spanning tree was constructed by the base station and rooted at it. In the decentralized access control the user can appear everywhere in the network.

A straightforward solution would be to construct a new spanning tree for each query with the user device as root. However, this is not a feasible solution because discovering a spanning tree requires much communication. Therefore, it is favourable to discover a spanning tree only once and use this information for each query. This spanning tree should not contain user devices or other mobile nodes because this would cause the tree to become obsolete every time a device moves. In the following we assume that the spanning tree is still generated by the base station and rooted at it.

However, in contrast to the previous tree-based scheme, each node does not store the first $t+1$ layers of the subtree rooted at it, but $t+1$ layers of a transformed spanning tree. The original spanning tree has to be transformed by each node s in such a way that s becomes the root of the new spanning tree. All nodes that are $t+1$ or less hops away from the node s in the original tree become its children in the transformed spanning tree.

An illustration of this transformation is shown in Figure 5.12. In this example, the white node is the selected source node. Figure 5.12(a) shows the original tree, Figure 5.12(b) an intermediate step and Figure 5.12(c) the result of the transformation.



(a) original (b) intermediate step (c) revised

Figure 5.12: Transformation of the original spanning tree into a tree with another root node.

Furthermore, just as in the case of remote access control via the base station, we assume that the network is organized in $(t+1)$-node clusters. Similar to the interleaved authentication along a path, the nodes in the

cluster should be able to authenticate messages for the first $t+1$ layers of the transformed spanning tree which begins at the cluster head. Each of the $t+1$ nodes should be assigned one of the first $t+1$ levels in the transformed spanning tree.

We now show how the transformed spanning tree is used for decentralized access control. According to the general scheme presented in Section 3.5.2.1, the user device logs in to the nodes of some cluster using public key cryptography. After a successful login, these nodes compute the authenticator, and the cluster head forwards user's query using t-robust interleaved authenticated querying along the transformed spanning tree.

5.6 Conclusions

In this chapter, we investigated t-robust access control in sensor network. We extended the interleaved report authentication scheme to work as a t-robust authenticated querying scheme. The straightforward extension turned out to be insecure and to have very high resource requirements. However, a modification to the tree-based scheme showed to be t-robust and have moderate resource requirements for small values of t.

We furthermore showed how to utilize the tree-based interleaved authenticated querying scheme for access control.

According to our performance analysis, t-robust access control is very resource intensive for large values of t. Especially the gathering and maintaining of the required topological information seems to be an expensive operation in this case.

Moreover, as the security breaks completely in case the adversary compromised more than t nodes, correct setting of the threshold t is very important for t-robust solutions. To be on the safe side, the value of t should actually be in the order of tens of nodes. However, in this case the presented scheme becomes prohibitively expensive. Furthermore, we showed that our scheme is the most efficient of its kind.

Motivated by the experience in devising t-robust solutions for access control, in the next chapter we investigate the paradigm of *gracefully degrading probabilistic* security for access control. In this paradigm, the security of the schemes degrades gracefully with the increasing number of captured nodes. This paradigm seems to be especially appropriate for sensor networks due to uncertainty concerning the number of nodes the adversary is able to capture. Additional advantage is that the probabilistic solutions are often more practical than the deterministic ones.

Chapter 6

Gracefully Degrading Access Control in WSNs

6.1 Introduction

In the previous chapter we considered t-robust authenticated querying. This method does not scale well, and requires the nodes to keep much state. Moreover, it is highly susceptible to link and node failures.

Furthermore, the adversary model where the number t of captured nodes is fixed beforehand and the security breaks completely after just one additional node capture, is difficult to justify. Indeed, if the adversary captured 5 nodes, there is actually no guarantee that it would not capture the 6th node. The obvious solution setting t very high is not practical, because t-robust schemes incur high communication and storage cost that grow rapidly with t.

One of the most popular ways to overcome impossibility or inefficiency of solutions in distributed systems is to make some of the protocol goals *probabilistic* [13, 94].

Another popular paradigm is *graceful degradation* [9, 28, 50] where the properties to achieved by the protocol depend on the power of the adversary. For weaker adversaries, these protocols achieve better performance or security than for stronger adversaries.

In this chapter, we present a novel probabilistic, gracefully degrading approach to authenticated querying, and discuss how to utilize this approach for access control.

The central idea of our approach is based on the fact that in general, before the query reaches the nodes that are able to answer it, it passes

through some significant part of the network. Thus, if some mechanism allows the sensor nodes to verify the legitimacy of the query probabilistically, illegitimate queries will be dropped before they can penetrate deeply into the network and reach the target nodes.

Our protocol restricts the propagation of fake queries to a constant area of the network, regardless of the network size. Some sensor nodes in our protocol may fail to recognize a fake query. As long as their number is very small, the effects of these false acceptances are negligible since other nodes will detect the fake query and not forward it further. By tuning the protocol parameters, our protocol also allows to trade off efficiency with security and hence can be tuned for different application scenarios.

Our protocol uses only symmetric cryptography and is based on the ingenious protocol by Canetti et al. [35], but it has a much better performance, as it relies on the implicit cooperation between the sensor nodes that occurs when the authenticated query is forwarded into the network.

6.2 Preliminaries

The resource constraints of sensor networks make them ideal candidates to employ probabilistic protocols. Such protocols are usually much more efficient than deterministic protocols at the price of ensuring the desired protocol properties with high probability rather than always. Probabilistic security protocols have been successfully employed in sensor networks for key establishment [37, 55, 79], secure aggregation [30, 120], filtering false data reports [151], or intrusion detection [115], but to the best of our knowledge have not been studied in the context of query authentication in sensor networks.

In the following, we present the detailed description of the most relevant for this chapter related work.

6.2.1 ID-Based Random Key Predistribution

Random key predistribution for sensor networks originates from [55]. The idea is that each sensor node is preloaded with randomly chosen k keys, called *key ring*, from the key pool of size l. The values of l and k can be chosen such that any two nodes have at least one common key with a given probability.

However, here we do not care about the probability that two neighboring nodes share a key, because in our scheme, key predistribution is used not

for secure and authenticated communication between the neighboring sensor nodes, but for authenticated querying.

ID-based random key predistribution was introduced in [158]. The keys in the key pool are numbered from 1 to l. Each sensor node s with a unique identifier id_s is first assigned k distinct integers between 1 and l by applying a pseudorandom number generator $PRG()$ with the seed id_s. Then the node s is preloaded with the keys whose identifiers are these k numbers from the sequence of pseudorandom numbers $PRG(id_s)$.

This method of choosing key rings enables to characterize sets of key identifiers very efficiently, as only the corresponding short seed needs to be known. This helps to save energy in a sensor network if the set of key identifiers needs to be transmitted over the air, as radio communication is very expensive in terms of energy. In this case, only the seed x is transmitted. Then, any node can determine if it knows some keys from a set of key identifiers KID_x characterized by the seed x. It computes $PRG(x) = KID_x$ and compares its own key identifiers to the key identifiers from KID_x.

6.2.2 1-bit MACs

In our protocol, we use message authentication codes (MACs) with a single bit output. The idea of using 1-bit MACs for broadcast authentication originates from [35]. We view a 1-bit MAC under a given key as a random function, i.e., we require the following:

- A single 1-bit MAC (under an unknown random key) cannot be feasibly guessed with any probability significantly exceeding $\frac{1}{2}$.
- Similarly, an m-bit string of m 1-bit MACs under m independent random keys cannot be guessed with probability significantly more than $\frac{1}{2^m}$.

1-bit MACs can be constructed from the usual MACs by taking their first bit of output.

6.3 System and Adversary Model

The generic system and adversary models were specified in Section 3.2. As presented there, we consider a large sensor network with a large number of users. The users can send queries to the network.

We consider an adversary with the goal of sending *arbitrary* queries to the network. The adversary can capture some sensor nodes, the number of

captured nodes is not limited but should be significantly smaller than the number of nodes in the network. We do not make any assumptions concerning the distribution of the captured nodes over the network. As stated in Section 3.2, this results in the distributed reprogramming adversary.

However, the schemes developed in this chapter can withstand a more powerful adversary than the one presented in Section 3.2. Whereas previously we considered the global eavesdropping adversary, in the sequel we consider a global adversary that can delete and replay messages, and also inject his own messages into the network. These capabilities result in the following adversary type:

$$\{\mathcal{A}_1(global, disturbing), \mathcal{A}_2(distributed, reprogramming)\}$$

6.4 Basic Authenticated Query Flooding

6.4.1 Protocol bAQF

We assume that ID-based random key predistribution as described in Section 6.2.1 is used in the network, and we want to reuse the predistributed keys for authenticated querying.

We now describe our basic protocol for authenticated query flooding, called `bAQF`. Pseudo-code descriptions of the algorithms for the base station and the sensor nodes are given in Algorithms 6.4.1 and 6.4.2.

Base station. The base station first computes the query q and a hash $x = h(q)$ of the query using a cryptographic hash function $h()$ [106]. Then it generates m key identifiers for the underlying ID-based key predistribution scheme: $KID_x = PRG(x) = (kid_1, \ldots, kid_m)$. We denote the corresponding key sequence by $K_x = (key_{kid_1}, ..., key_{kid_m})$.

Then the base station computes m 1-bit MACs on $h(q)$ using the keys from K_x. We call these m 1-bit MACs *authenticator* for q, denoted as $macs(q)$.

Finally, the base station floods the query q into the sensor network, accompanied by the authenticator for the query.

Sensor nodes. Upon receiving the query q with the authenticator $macs(q)$, each node s computes $x = h(q)$ and the sequence of key identifiers $KID_x = PRG(x)$. It compares the key identifiers from KID_x to its own key identifiers in order to find out if it knows some keys from K_x.

If s knows some keys, it verifies the corresponding 1-bit MACs from $macs(q)$. If any one of them does not verify correctly, the sensor drops the query. If all verifiable MACs are correct or if the node is not able to verify any MACs (i.e., it does not know any keys from K_x), the node forwards the query to its neighbors according to the underlying flooding mechanism or transport protocol.

Algorithm 6.4.1: bAQF-generate (Query q, KeyPool $(key_1, \ldots, key_l)$)

$x = h(q)$ /* compute the hash value */
$KID_x = (kid_1, \ldots, kid_m) = PRG(x) \in \{1, \ldots, l\}^m$
$macs(q) = (mac_1, \ldots, mac_m) =$
(1-bit-MAC$(key_{kid_1}, x), \ldots,$ 1-bit-MAC$(key_{kid_m}, x))$
return $macs(q)$

Algorithm 6.4.2: bAQF-verify (q, $macs(q)$, KeyRing $(key_{r_1}, \ldots, key_{r_k})$)

$x = h(q)$ /* compute the hash value */
PRG-init(x)
for $i = 1$ to m **do**
  kid_i =PRG-next()
  if $kid_i \in \{r_1, \ldots, r_k\}$ **then**
    if 1-bit-MAC$(key_{kid_i}, x) \neq mac_i$ **then**
      return $false$ /* reject the query */
    end if
  end if
end for
return $true$ /* forward the query */

6.4.2 Analysis

The query of a legitimate user will be flooded into the sensor network without any obstacles. However, a query forged by an adversary will only be able to reach a limited part of the network, as some sensor nodes will discard the query. In the following, we analytically determine how many 1-bit MACs should be appended to a query in order to limit the propagation of a fake query to a small constant part of the network.

The variables used in the analysis are summarized in Table 6.1.

meaning of the variable	variable	typical values
number of nodes in the sensor network	n	$1000 - 10000$
number of keys in the key pool	l	10000 - 100000
number of keys in the key ring of a node	k	50 - 250
node density (average number of neighbors of a node)	d	5 - 50
number of captured senor nodes	$\tilde{n}$	$0 - 50$
number of captured keys	$\tilde{b}$	Formula 6.1
number of keys in the authenticator which the adversary knows	$E_{\tilde{b}}$	Formula 6.2
number of correct bits in the fake authenticator	B	Formula 6.3
probability that the message will be forwarded	p_f	Formula 6.5
size of the authenticator	m	100 - 500 bits

Table 6.1: Variables used in the analysis of `bAQF`

6.4.2.1 Probability of Accepting a Fake Query by an Arbitrary Node

Using a common model for cryptographic hash functions [12], it is infeasible to first choose some x and then search for an appropriate value q with $h(q) = x$, or to fix *any* properties for the desired x and then search for a query q with satisfying $h(q)$. For different queries q, the adversary always receives independent random values $x = h(q)$.

Therefore, we assume that the adversary uses the following strategy: It computes the seed $x = h(q)$ for its query q, computes the appropriate sequence of key identifiers KID_x using $PRG(x)$, and hopes that it knows enough keys with identifiers from KID_x in order to be able to construct a fake query.

In the following, we compute the probability of a fake query generated as above to propagate successfully through the sensor network assuming that the adversary captured $\tilde{n}$ sensor nodes and guessed the bits of authenticator which it could not compute.

If $\tilde{n}$ sensor nodes are compromised, then the adversary knows on average $\tilde{b}$ keys:

$$\tilde{b} = l(1 - (1 - \frac{k}{l})^{\tilde{n}}) \tag{6.1}$$

Formula 6.1 assumes that the keys are distributed according to the uni-

form probability distribution. Given that the adversary knows $\tilde{b}$ keys out of l, we can compute the average number of bits in an authenticator of length m that will be correct due to the adversary's partial knowledge of the key space l:

$$E_{\tilde{b}} = m\frac{\tilde{b}}{l} \tag{6.2}$$

Since the attacker knows nothing about the other keys in the authenticator, it has to guess the other bits. There it will have the probability of 50% to guess the correct value. This lets us compute the total number of correct bits in the faked authenticator:

$$B = E_{\tilde{b}} + \frac{m - E_{\tilde{b}}}{2} = \frac{m(\tilde{b} + l)}{2l} = \frac{m(2 - (1 - \frac{k}{l})^{\tilde{n}})}{2} \tag{6.3}$$

We can finally compute the probability p_f that a sensor accepts the query with the fake authenticator:

$$p_f = \left(\frac{l-k}{l} + \frac{k}{l}\frac{B}{m}\right)^m \tag{6.4}$$

The expression in the parentheses gives the probability that one bit of the authenticator passed the test by a particular sensor node. The first summand expresses the probability that the sensor node does not share any keys with the claimed set of key identifiers $PRG(x)$. The second summand shows the probability that the adversary either could compute the appropriate bit or guessed it.

Substituting B in Formula 6.4 using Formula 6.3 finally yields:

$$p_f = \left(1 - \frac{1}{2}\frac{k}{l}\left(1 - \frac{k}{l}\right)^{\tilde{n}}\right)^m \tag{6.5}$$

6.4.2.2 Limiting the Propagation of Fake Queries

The last section calculated the probability that each single node forwards a fake query. It is yet open, however, how the network as a total behaves, namely, whether the query reaches a significant number of nodes, or is stopped from doing so.

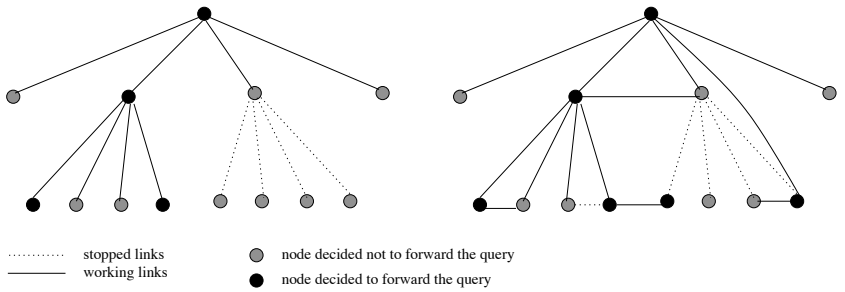


Figure 6.1: Query propagation in a branching-process-like network (to the left), and in a wireless sensor network.

To assess the network-wide behavior of our scheme, the propagation of fake queries can be roughly approximated by the development of a variant of a *branching process* [56]. In this process, entities which can produce entities of the same kind are considered. A single entity starts the process. Each entity creates d descendants with probability p, and does not create any descendants with probability $1 - p$. Thus, the population of entities may extinct after some generations if too little entities are produced. It is well known that if $dp < 1$, the extinction happens with probability 1. A branching-process-like network is a d-regular tree, see Figure 6.1. The root starts the query propagation. After receiving the query, each node forwards the query with probability p, and remains silent with probability $1 - p$.

We can consider the dissemination of a fake query as a branching process with $p = p_f$. However, this is only a rough estimation, because of the higher connectivity of sensor networks. For example, in Figure 6.1, only four nodes of the last depicted generation receive the query, and only two nodes forward the query. However, in the example sensor network, 7 nodes receive the query, and four of them forward the query. Nevertheless, we will see in the following that this estimation gives sufficiently good results which are also confirmed by simulation.

If we see the fake query dissemination as a branching process, we have the following criterion for the propagation of the fake query to be stopped:

$$p_f d < 1 \tag{6.6}$$

Another justification for Formula 6.6 is as follows. If a node has d neighbors, each of them forwarding the fake query with the probability p_f, then the average number of nodes which forward the query is $p_f d$. In this case,

the query propagation should be limited if less than one neighbor on average forward the fake query.

From Formulas 6.5 and 6.6 it follows:

$$\frac{1}{d} > p_f = \left(1 - \frac{1}{2}\frac{k}{l}\left(1 - \frac{k}{l}\right)^{\tilde{n}}\right)^m \tag{6.7}$$

$$\Leftrightarrow \quad m > \frac{-\log d}{\log\left(1 - \frac{1}{2}\frac{k}{l}\left(1 - \frac{k}{l}\right)^{\tilde{n}}\right)} \tag{6.8}$$

6.4.3 Choosing Parameters

In Formula 6.8, we have variable parameters $d, l, k, \tilde{n}$ for the length of the authenticator m. The administrators of the network control parameters d, l, k and m, while the adversary controls $\tilde{n}$.

The next task is to find suitable ranges for d, l, k and m, such that the adversary is unable to send fake queries for reasonable ranges of $\tilde{n}$. We did so analytically, as well as by simulation (see Section 6.4.4).

Firstly, we comment on the choice of the node density parameter. If the node density is too high, then the capacity of wireless networks decreases. On the other hand, if the network density is too low, the network may become disconnected. Node density required to ensure connectivity can be estimated as $\Theta(\log n)$ [149], but the exact number of neighbors remains an open problem. For networks of moderate size, 6 to 8 neighbors may be considered [113].

The size of the key ring k is constrained by the amount of memory available on sensor nodes. In a typical sensor node the RAM size is about 10 kB. Then, if the key size is 80 bits,the nodes would be able to hold up to 200 keys. After choosing k, we can choose l such that $\frac{k}{l}$ is sufficiently small. As the key pool is stored at the base station, we do not have serious limits on the size of l.

Figure 6.2 depicts the necessary authenticator sizes for node density 7, depending on the ratio $\frac{k}{l}$, and the number of compromised nodes $\tilde{n}$ according to Formula 6.8. It can be clearly seen that there is an optimal ratio $\frac{k}{l}$ for any particular number of compromised nodes. Below we analytically determine this ratio.

Firstly, we find first derivation of the Expression 6.9, which evolves from

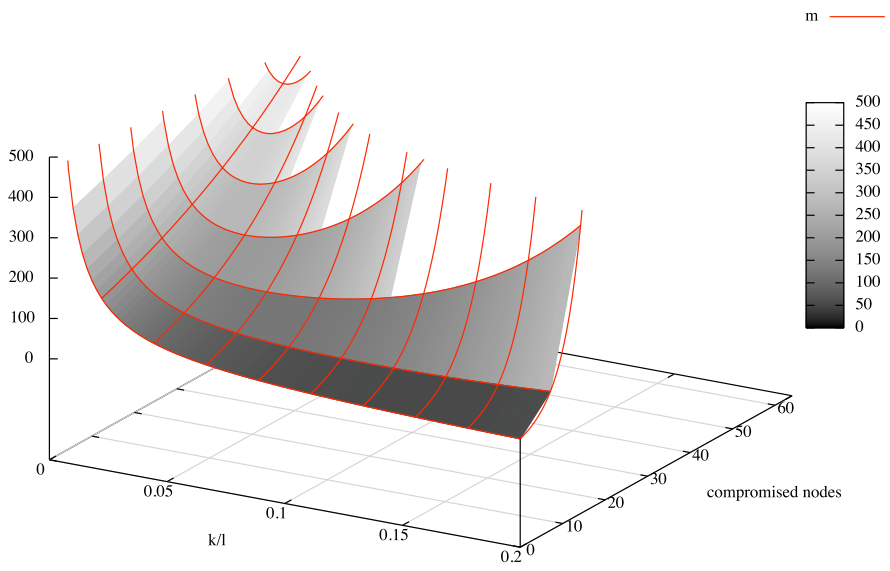


Figure 6.2: Necessary authenticator size for node density $d = 7$, depending on the ratio of the key ring size k to the size of the key pool l, and the number of compromised nodes $\tilde{n}$.

Formula 6.8 with the substitution $\frac{k}{l} = \lambda$:

$$m = \frac{-\log d}{\log\left(1 - \frac{1}{2}\lambda\left(1-\lambda\right)^{\tilde{n}}\right)} \Rightarrow \tag{6.9}$$

$$\frac{\mathrm{d}m}{\mathrm{d}\lambda} = \frac{\log d}{2} \frac{-(1-\lambda)^{\tilde{n}} + \lambda\tilde{n}\frac{(1-\lambda)^{\tilde{n}}}{(1-\lambda)}}{\log\left(1 - \frac{\lambda}{2}(1-\lambda)^{\tilde{n}}\right)^2 \left(1 - \frac{\lambda}{2}(1-\lambda)^{\tilde{n}}\right)} \tag{6.10}$$

Then we can find the optimal λ such that m is minimized:

$$\frac{\mathrm{d}m}{\mathrm{d}\lambda} = 0 \quad \Rightarrow \quad \lambda_{min} = \frac{1}{1+\tilde{n}} \tag{6.11}$$

Then we can find the minimum authenticator size m depending on the number of captured nodes $\tilde{n}$:

$$m_{min} = \frac{-\log d}{\log\left(1 - \frac{1}{2}\frac{1}{1+\tilde{n}}(\frac{\tilde{n}}{\tilde{n}+1})^{\tilde{n}}\right)} \tag{6.12}$$

Figure 6.3 depicts m_{min} for node density = 7. The function is almost linear, which is also confirmed by simulations in Section 6.4.4.

6.4.4 Simulation Results

We simulated `bAQF` using Shawn [89], a discrete event simulator for large wireless sensor networks. We used a key pool of $l = 10,000$ keys and varied node density $d \in \{7, 15\}$ and key ring size $k \in \{75, 150\}$. In each simulation run, 1000 nodes were randomly and uniformly placed such that the given node density d was satisfied. The source of the query (base station or the adversary) was also placed randomly in the sensor field. The query was sent, accompanied by the authenticator of size m. We looked into the number of nodes reached by an illegitimate query for $m = 50, 100, 150, \ldots, 500$ assuming that the adversary captured $\tilde{n} = 0, 1, 2, 4, 16, 32, 64, 128$ nodes. For each combination of parameters, 50 protocol runs on different network topologies were performed. We only present the most significant results here.

To make the comparison to the analytical results easier, we also show the the results of the analytical evaluation using Formula 6.8 for the parameters above in Figure 6.4.

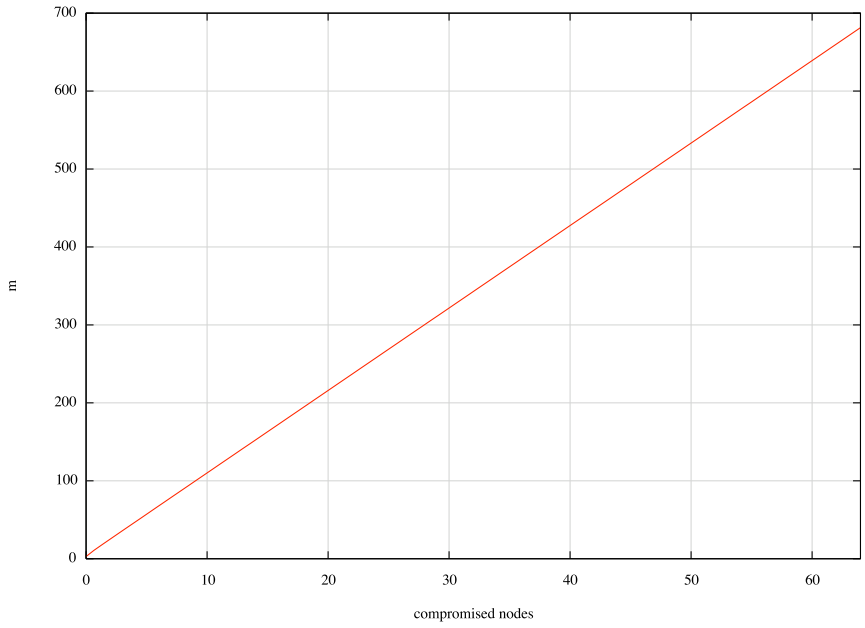


Figure 6.3: Minimum authenticator size m_{min} for node density 7 depending on the number of compromised nodes.

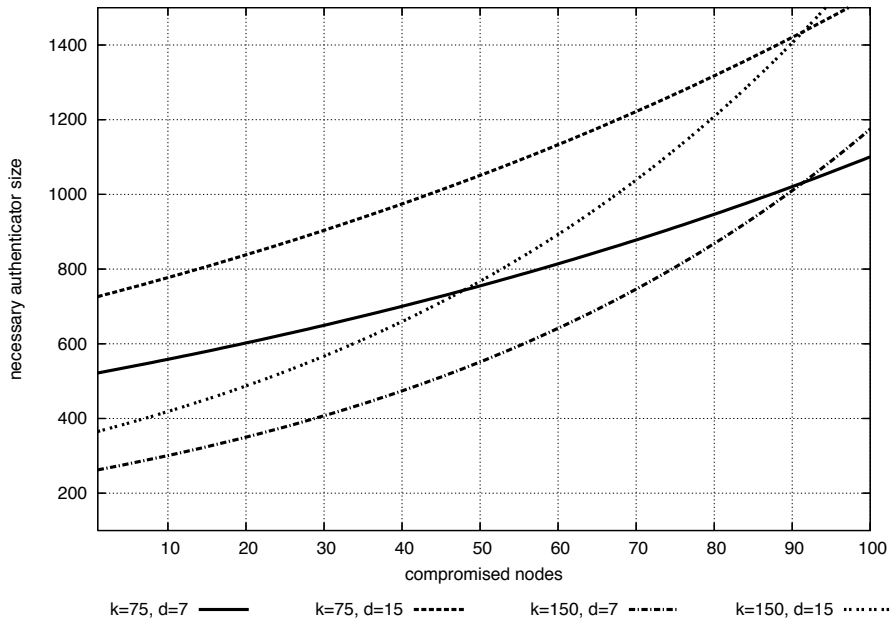


Figure 6.4: Necessary authenticator size, depending on node density, key ring size, and the number of compromised nodes. The size of key pool is $l = 10000$.

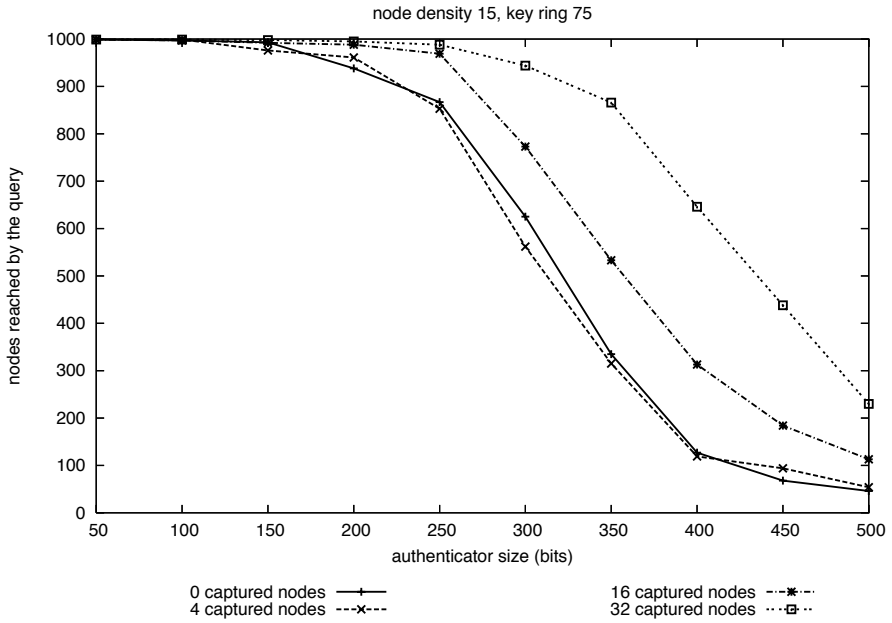


Figure 6.5: Number of nodes reached by the fake query depending on authenticator size, with node density 15 and key ring size 75.

At node density 15 and key ring size 75 all the networks were fully connected, that is, legitimate queries always reached all of nodes. However, also illegitimate queries reached a significant part of the network even if no nodes were captured, until the authenticator size reached the unacceptable 500 bits. This confirms our analytical results showing that in this case, authenticator size of around 700 bits are needed, see Figure 6.4.

Formula 6.8 indicates that the size of the authenticator decreases with the decreasing node density and increasing key ring size. In Figure 6.6, results for node density 7 and key ring size 150 are presented. With node density 7, the network may already become disconnected. However, the number of sensors which were disconnected from the network in this case was negligible. According to simulations, authenticator size $m = 250$ bits is adequate in case the adversary captured around 30 nodes. Analytical results suggest the authenticator of around 400 bits here.

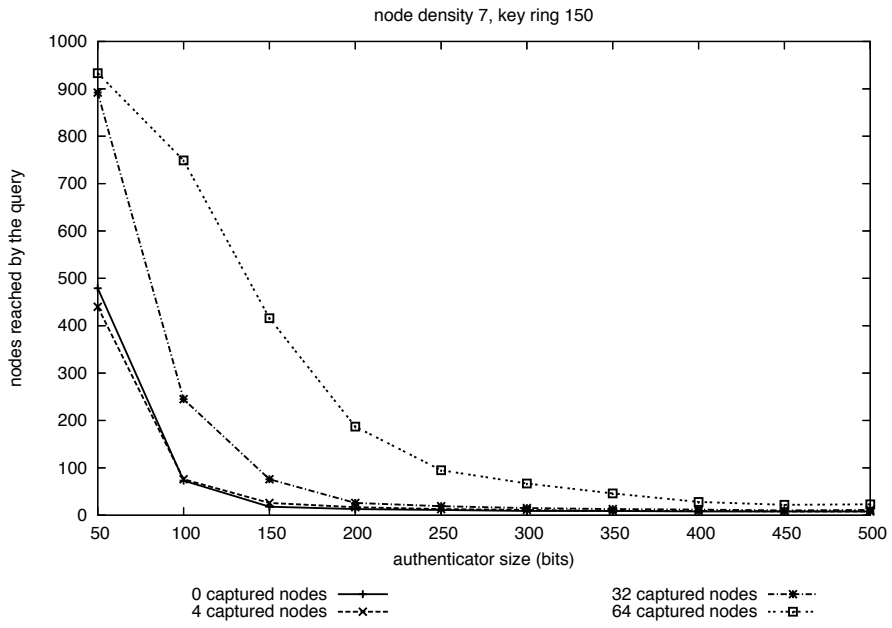


Figure 6.6: Number of nodes reached by the fake query depending on authenticator size, with node density 7 and key ring size 150.

According to Formula 6.11 , in case the network density $d = 7$ and the number of compromised nodes $\tilde{n} = 64$, the optimal ratio $\frac{k}{l} = \frac{1}{65} = 0.015$ can be computed. The minimum authenticator size for this case is around 700 bits according to Figure 6.3. Our simulation for this scenario, with $\frac{k}{l} = \frac{150}{10000} = 0.015$ depicted in Figure 6.6, suggest the authenticator size around 500 bits.

In general, we can see that our analysis gives an estimate of the authenticator size very much "on the safe side", and therefore, can be used for determining parameters of our scheme with an adjustment.

6.4.5 Improving Efficiency

The protocol `bAQF` works efficiently for sparse sensor networks. That is, the sparser the network the less the size of the authenticator. The reason that in denser networks larger authenticator size is required lies in the fact that if each sensor node has a lot of neighbors, each forwarding event increases the probability of a query to get trough the network. And as the adversary always can guess at least half of the bits in the authenticator, even queries with completely guessed authenticator propagate successfully.

To thwart this disadvantage, more sophisticated flooding mechanisms, such as studied in [146, 152] should be used. These mechanisms reduce the communication cost of flooding through restricted dissemination of messages. The nodes do not forward all messages to all their neighbors. Instead, the nodes may forward messages only to a restricted set neighbors. Another technique is that only a restricted set of nodes may forward the messages, and the other nodes remain silent in the flooding process.

Reducing the number of all possible paths which a faked query may take results in the reduced authenticator size. For example, if the network is organized as a spanning tree, then stopping the fake query at one node means that all nodes in the subtree rooted at this node do not receive the fake query.

6.4.6 Attacks and Countermeasures

6.4.6.1 Thwarting the Global Disturbing Adversary

As presented in Section 6.3, our adversary model is a combination of global disturbing and distributed reprogramming adversary. We consider the global disturbing adversary first.

As the global disturbing adversary can send messages to all network nodes simultaneously, the obvious attack is global jamming. Defenses against

this attack were presented, e.g., by Wood et al. [147] and are not considered here. Another DoS attack is sending meaningless messages to all nodes, such that they have to spend energy in verifying them. However, as the principal goal of the adversary considered here is to send arbitrary queries to the network, we also do not consider sophisticated defenses against this attack here. A simple approach is for the nodes to shut down for some time after the rate of received invalid queries exceeds a predefined threshold.

On the other hand, principal goal of the adversary, according to Section 6.3, is to send arbitrary queries to the WSN, i.e., to gain the same access to the data as the legitimate user. We now consider an adversary that sends the same fake query to all nodes in the network. If the network size is n, then $p_f n$ nodes on average will accept the fake query as a legitimate one. As the probability p_f can be as large as 0.3 (or even 0.5 in some cases), the fraction of nodes that accept the fake query can be fairly large.

In order to find a countermeasure to this attack, we note that the query sent by a global adversary should appear to come from a legitimate node (a node should not accept any messages coming out of the thin air). Thus, if all nodes know their neighbors and have means to verify that the message came from a particular neighbor, the global disturbing adversary is unable to send the same message to all nodes simultaneously.

We assume that the nodes know their neighbors, and that each node can authenticate its local broadcast using, e.g., one-way key chains as in LEAP [156]. Local broadcast authentication actually precludes the global disturbing adversary from sending any legitimate messages at all, as each node only accepts authenticated messages from its immediate neighbors. To send a legitimate message, the adversary has to know the authentication information of the node. Thus, in our adversary model, only the distributed reprogramming part of the adversary can send meaningful messages to legitimate nodes.

In the following, we always assume that appropriate measures were taken for defending the global disturbing adversary, and consider the distributed reprogramming adversary in the sequel.

6.4.6.2 Denial-of-Service attacks

The adversary may try to drain the energy of sensor nodes by sending illegitimate queries which the sensor nodes have to verify. However, as our protocol guarantees that fake queries propagate only to a small constant part of the network, this attack will only affect a small network region.

We note that any kind of DoS attack requires the adversary to propagate

a large amount of queries into the sensor network. Below, we discuss the attacks where the goal of the adversary is to propagate illegitimate queries in order to receive answers to them. The countermeasures to these attacks will also protect the network against DoS attacks.

6.4.6.3 Simple replay attack

By just listening to valid conversations, the adversary can easily learn many pairs $(q, macs(q))$ of query and valid authenticator. Our protocol does not provide any defense against the adversary sending such pairs again – at any point in time, and as often as useful for the adversary.

Countermeasures The application can defend against such attacks, e.g., by using timestamps and storing the queries from the current time-frame. A query (q, x) is then not forwarded if q has been sent in the current time frame, or if the time-stamp inside q does not fit. Note that knowing the authenticator $\text{macs}(q)$ for a query q does not help the adversary in guessing the authenticator $\text{macs}(q')$ for $q' \neq q$, even if q' is identical to q except for the timestamp. The disadvantage of this countermeasure is that it relies on time synchronization, whereas the goal of this work was to devise an authentication protocol which does not use synchronized clocks.

However, with our scheme we can use other approaches which guarantee message freshness, e.g., counters [67]. In this case, the base station disseminates the queries accompanied by a counter instead of the timestamp, and the sensor nodes discard queries with "old" counters.

The latter countermeasure opens a possibility for a denial-of-service (DoS) attack. The adversary may send a faked query with the future counter c. Then the sensor nodes which accept the query as a legitimate, or cannot verify the query, would not accept the legitimate query accompanied by the counter c. However, this DoS attack is localized. As the query is faked, only a small part of the nodes will accept it. Then, also only a small part of the nodes would be unable to answer the legitimate query with the same counter.

From now on, we assume that the adversary sends a query q without knowing the correct authenticator $\text{macs}(q)$ in advance.

6.4.6.4 Key identifier disclosure attack

Let $e(i) = (0^{i-1}10^{m-i-1}) \in \{0,1\}^m$ be the m-bit vector with exactly one bit set to 1, namely at position i. Let an authentic message $(q, macs(q))$

be given. The defense from simple replays would allow a sensor node s to reject any message (q, x).

Let us assume, however, that s did not hear the query $(q, macs(q))$, because the adversary shielded it from the network. Then the adversary can easily find out which single-bit MACs are verified by s (or, equivalently, the identifiers of the keys which s holds):

- Send $(q, macs(q) \oplus e(i))$ to s.
- If s accepts this as authentic, i.e., forwards q, then s does not check the i-th MAC.
- Else, s rejects $(q, macs(q) \oplus e(i))$ – and thus, holds the i-th key.

Countermeasures The above attack assumes that the adversary sends out a lot of queries – most of them being invalid. To defend, a node simply counts the number of invalid requests it receives. If the counter exceeds a certain threshold, the node considers itself under attack and switches off. An advanced variant of this countermeasure would allow to forward queries according to some probability distribution that depends on the number of invalid requests, e.g., some exponentially decreasing probability.

6.4.6.5 Modifying replay attacks

If the adversary is able to send a query q with different authenticators $macs(q)$, it might gain a broader access to the network in case the adversary can observe parts of the network. The attack works as follows: if the adversary has no knowledge about a single bit in $macs(q)$, it sends one message with the bit set, and one with the bit cleared. It can then guess from the number of nodes accepting the message, whether the bit should be set or not. This can be repeated for all bits that are unknown to the adversary, until the message either reaches a sufficient number of sensors, or the adversary knows how to set all bits correctly.

Countermeasures As in the previous key identifier disclosure attack, the adversary sends out a high number of invalid queries. Then switching off sensor nodes that receive invalid queries too often effectively isolates the compromised node, such that it is unable to send future requests.

6.4.7 Comparison to Related Work

We now consider which goals of the broadcast authentication from Section 4.6.1 on page 56 are satisfied by `bAQF`.

We first note that `bAQF` is moderately communication efficient. For example, with the authenticator size of 250 bits (32 bytes), we are able to restrict the propagation of the fake query to approximately 10 nodes in sufficiently sparse networks. Thus, the `bAQF` authenticator is smaller than an ECC digital signature that is 40 bytes long, but it is much less efficient than μTESLA that requires 4 bytes per message and 20 bytes per time interval. However, `bAQF` does not need time synchronization. Moreover, it provides immediate authentication of an arbitrary number of messages (without the need in re-initialization), and the impact of DoS attacks is contained to a small part of the network.

`bAQF` is also quite computation efficient, as it uses only a moderate amount of symmetric cryptography operations. In order to estimate the performance of the query verification algorithm `bAQF-verify` of our scheme `bAQF`, we assume the key pool size $l = 10,000$, and the key ring size $k = 150$. Therefore, the size of a key identifier is 14 bits, we take 16 bits for convenience of computation. As our analysis and simulations suggest authenticator size m of 200-300 bits, we assume $m = 256$ for convenience. The verification of the query q consists of computing its hash value $h(q)$, generating 256 key identifiers using $PRG(h(q))$, and finally computing some MACs on $h(q)$. On average, a sensor node would know $\frac{m \cdot k}{l}$ keys from the authenticator. Thus, the sensor node needs to generate $256 \cdot 16 = 4096$ pseudorandom bits, and to compute $\lceil \frac{256 \cdot 150}{10,000} \rceil = 4$ MACs on average. We note that the pseudo-random number generator does not need to be cryptographically secure, and can therefore be implemented very efficiently.

6.4.8 Summary

We presented a probabilistic protocol `bAQF` for authenticated query flooding in sensor networks. The protocol guarantees that the faked queries can only reach a small part of the network that does not depend on the number of nodes in the network, but on the network density. For example, using the authenticator of size around 250 bits, we are able to restrict the propagation of the fake query to approximately 10 nodes in sufficiently sparse networks. However, our protocol also may be used in denser sensor networks if they employ an advanced query dissemination strategy that reduces the number of nodes forwarding the query.

In the next section, we present a more efficient protocol for authenticated querying that satisfies the same properties.

6.5 Simple Authenticated Query Flooding

The protocol `bAQF` presented in the previous section was developed under the assumption that the considered sensor network employs an ID-based random key predistribution scheme for organizing secure communication between sensor nodes. We aimed at reusing the key rings of the nodes for authenticated querying.

However, Formula 6.5 shows that the probability for node to recognize a fake query depends on the ratio $\frac{k}{l}$, where k is the number keys per node, and l is the overall number of keys in the key pool. Thus, if we do not consider a network which employs random key predistribution, more efficiency can be gained by minimizing the value of k. Then the value of l would also decrease. Actually, it can be as small as m, the number of bits in the authenticator.

Based on this idea, we developed the protocol `sAQF` (*simple Authenticated Query Flooding*) which is described and analyzed in the following.

6.5.1 Protocol `sAQF`

We now describe a more efficient than `bAQF` protocol for authenticated query flooding, called `sAQF`. Here, the key pool comprises m keys, and all these keys are used to generate the authenticator. Thus, the authenticator contains m 1-bit MACs, and each sensor node can verify k of these MACs using its key ring. The m keys are numbered, such that each node knows at which place at the authenticator it should verify the k 1-bit MACs which are known to it.

Pseudo-code descriptions of the algorithms for the base station and the sensor nodes are given in Algorithms 6.5.1 and 6.5.2.

Base station. The base station knows m symmetric keys $key_1, \ldots key_m$. This collection of keys is called *key pool*.

The base station first computes the query q and a hash $x = h(q)$ of the query using a hash function $h()$. Then it generates m 1-bit MACs on $h(q)$. We call these m 1-bit MACs *authenticator* for q, denoted as $macs(q)$.

Finally, the base station floods the query q into the sensor network, accompanied by the authenticator for the query.

Sensor nodes. Each sensor knows k keys from the key pool which are selected randomly and independently for each sensor according to the uniform probability distribution. The nodes know the IDs of their keys. Node's collection of keys is called *key ring*.

Upon receiving a new query q with the authenticator $macs(q)$, each node s verifies the k 1-bit MACs from $macs(q)$ which should be computed using its k keys.

If all verified MACs are correct, the node forwards the query to its neighbors according to the underlying flooding mechanism or transport protocol. Otherwise, the query is dropped.

Algorithm 6.5.1: sAQF-generate (Query q, KeyPool $(key_1, \ldots, key_m)$)

$x = h(q)$ /* compute the hash value */
$macs(q) = (mac_1, \ldots, mac_m) =$
$(\text{1-bit-MAC}(key_1, x), \ldots, \text{1-bit-MAC}(key_m, x))$
return $macs(q)$

Algorithm 6.5.2: sAQF-verify (q, $macs(q)$, KeyRing $(key_{r_1}, \ldots, key_{r_k})$)

$x = h(q)$ /* compute the hash value */
for $i = 1$ to k **do**
  if $\text{1-bit-MAC}(key_{r_i}, x) \neq mac_{r_i}$ **then**
    return $false$ /* reject the query */
  end if
end for
return $true$ /* forward the query */

6.5.2 Analysis

6.5.2.1 Probability of Accepting a Fake Query by an Arbitrary Node

Assume that the adversary captured $\tilde{n}$ nodes and, therefore, knows their key rings. The adversary computes its query and tries to compute the corresponding authenticator.

Consider the ith authenticator bit. If the adversary knows key_i, it can compute the corresponding 1-bit MAC. If the adversary does not know key_i, its best strategy is to guess the corresponding 1-bit MAC with probability $\frac{1}{2}$.

Assume that node s which is not captured by the adversary receives the fake query and starts verification of some authenticator bit b_i. We firstly compute the probability p_{bit_known} that the adversary was able to compute b_i due to its partial knowledge of the key pool. The probability that one of the captured nodes does not have the corresponding key key_i in its key ring is

$$1 - \frac{k}{m} \tag{6.13}$$

As the nodes are captured independently, the probability that none of the captured nodes knows key_i is

$$\left(1 - \frac{k}{m}\right)^{\tilde{n}} \tag{6.14}$$

Thus, the probability that at least one captured node knows key_i is

$$p_{bit\_known} = 1 - \left(1 - \frac{k}{m}\right)^{\tilde{n}} \tag{6.15}$$

On the other hand, the adversary does not know b_i with probability $1 - p_{bit\_known}$. In this case, the adversary has to guess the bit. Thus, it guesses b_i with probability

$$p_{guessed} = \frac{1}{2}\left(1 - \frac{k}{m}\right)^{\tilde{n}} \tag{6.16}$$

Putting together Equations 6.15 and 6.16, the adversary computes b_i correctly with probability

$$1 - \frac{1}{2}\left(1 - \frac{k}{m}\right)^{\tilde{n}} \tag{6.17}$$

Therefore, the adversary correctly computes all k bits that a particular node can verify with probability

$$p_f = \left(1 - \frac{1}{2}\left(1 - \frac{k}{m}\right)^{\tilde{n}}\right)^{k} \tag{6.18}$$

6.5.2.2 Limiting the Propagation of Fake Queries

Using the same reasoning as in the analysis of `bAQF` in Section 6.4.2.2, we suggest the following criterion for limiting the propagation of the fake query:

$$p_f d < 1 \tag{6.19}$$

Here d is the average number of node's neighbors. Of course, this criterion works especially well in networks with uniformly distributed nodes.

6.5.3 Choosing Parameters

Analogous to Section 6.4.3, it is also possible to find the optimal k such that m is minimized for every pair of parameters d and $\tilde{n}$. As the method and the results of this analysis are very similar to the results of the analysis of `bAQF`, we do not present it here. Instead, we present some clarifying examples.

6.5.3.1 Query Propagation by Flooding

We consider the authenticator size $m = 200$ bits, as the analysis and simulations in Section 6.4 showed that a smaller authenticator size does not work well for realistic node densities if the query dissemination is organized using flooding. We also consider $m = 400$ bits as an upper bound on the authenticator size. This is due to the fact that a digital signature using elliptic curves (e.g., the ECDSA standard [46]) has size of about 320 bits.

The upper bound on the number of compromised nodes was shown to be at around $\tilde{n} = 50$. On the other hand, for $\tilde{n} = 20$ the simulation of `bAQF` showed quite a good results.

Having fixed the parameters m and $\tilde{n}$, we can now find the optimal k such that p_f is minimized. In Figure 6.7 p_f is computed according to Formula 6.18 for all four combination of the parameters m and $\tilde{n}$ above is depicted.

We can see that for each pair $(m, \tilde{n})$ there is an optimal value of the key ring size k which minimizes the probability of forwarding the fake query. For example, for $m = 200$ and 20 compromised nodes, the optimal size of the key ring is $k = 8$, and the corresponding probability $p_f \approx 0.14$. Then, according to our criterion, the fake query should be stopped effectively for the networks where the nodes have 7 or less neighbors on average. In Section 6.5.4 we verify this result further using simulations.

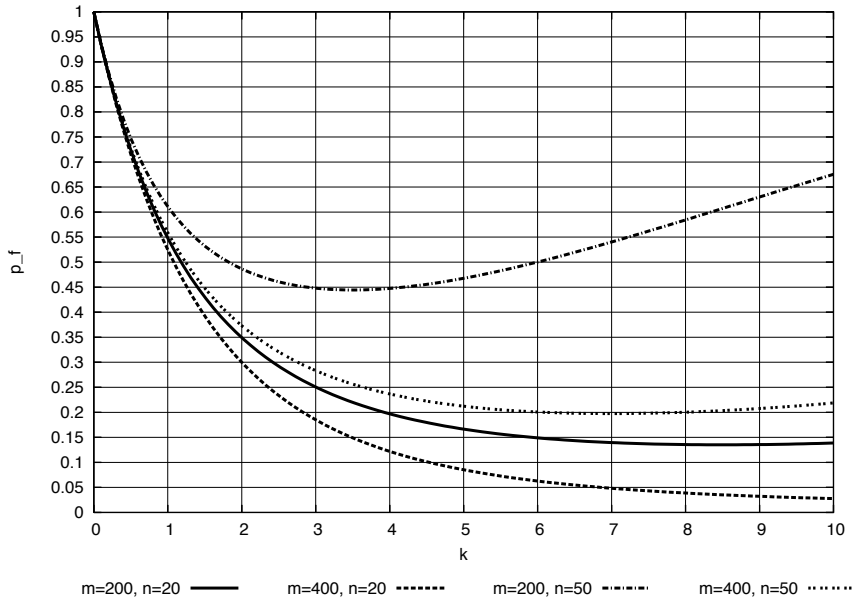


Figure 6.7: Probability of forwarding the fake query for the authenticator size $m \in \{200, 400\}$ and the number of compromised nodes $\tilde{n} \in \{20, 50\}$.

6.5.3.2 Query Propagation along a Spanning Tree

If the network is organized as a spanning tree with a moderate degree, we can afford even quite small authenticator sizes such as $m = 100$.

In a spanning tree, the fake query can propagate only along subtrees rooted at the compromised nodes. Moreover, for each honest node s, there is a single path along that a fake query may arrive.

Consider the propagation of the fake query along a single path. Each non-compromised node along the path drops the fake query with probability $1 - p_f$. Thus, we can consider the fake query forwarding as a Bernoulli process. Then the expected number of forwarding events till the first drop is $\frac{1}{1-p_f}$.

For $m = 100$ and $\tilde{n} = 30$ the optimal number of keys pro node can be computed as $k = 3$, and the minimal probability of forwarding the fake query is $p_f \approx 0.51$. Then the expected number of forwarding events for the fake query is 2.04. This means that a query that started at some node in the spanning tree would reach on average the third level of the subtree rooted at this node.

It can be shown that in a uniformly distributed network, the average degree of the spanning tree goes to 1 with the increasing network diameter. Thus, for such a network, authenticator size $m = 100$ could suffice.

We note, however, that if the security of an authentication scheme relies on topology control in the network, then secure topology control schemes should be developed. Secure spanning tree construction, as well as other secure schemes for topology control, would be an interesting and challenging future work.

6.5.4 Simulation Results

We simulated `sAQF` on the network of 1000 nodes with node density $d = 7$ for the authenticator sizes $m \in \{200, 400\}$ and key ring sizes $k \in \{4, 6, 8, 10\}$. We varied the number of compromised nodes $\tilde{n} \in \{10, 20, 30, 40, 50\}$. The goal was to validate the analysis in Section 6.5.2.

Figure 6.8 shows the number of nodes that accepted the fake query for the authenticator size 200 bits. Key ring size $k = 8$ is analytically optimal for $\tilde{n} = 20$ compromised nodes, as shown in Figure 6.7. On the other hand, $k = 4$ is optimal for $\tilde{n} = 50$.

We can see that `sAQF` with $k = 8$ indeed performs very well for $\tilde{n} = 20$. Less than 2 nodes on average accept the fake query. Also for $\tilde{n} = 30$ the scheme behaves well, as only approximately 8 nodes accept the fake query.

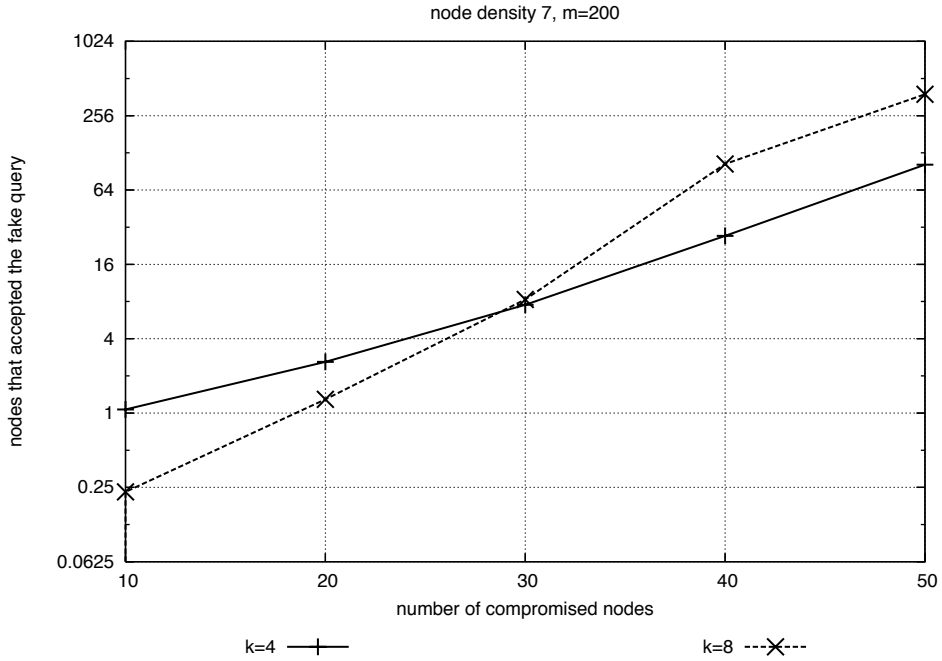


Figure 6.8: Number of nodes that accepted the fake query for the authenticator size 200 and key ring sizes 4 and 8.

In the latter case, $p_f \approx 0.28$ according to Formula 6.18. This means that the node density for that the scheme works well should be less than $\frac{1}{0.28} \approx 3.6$. We can see that the criterion in Formula 6.19 slightly underestimates the security of `sAQF`.

On the other hand, `sAQF` with $k = 4$ also behaves well for $\tilde{n} \leq 30$, even if its performance is slightly worse than for $k = 8$. However, for $\tilde{n} = 40$ the situation changes rapidly. Here, the number of nodes that accepted the fake query for $k = 4$ is more than tree times less than for $k = 8$ (30 versus 100).

The latter observation shows that the security of the proposed scheme degrades gracefully with the increasing number of compromised nodes until a threshold is reached where the scheme becomes completely insecure. Therefore, a realistic upper bound on the number of compromised nodes to be tolerated should be fixed beforehand, and the other parameters of the scheme should be computed accordingly.

6.5.5 Summary

We presented `sAQF`, a probabilistic protocol for authenticated querying which degrades gracefully with the growing number of captured nodes ans is more efficient than `bAQF` from Section 6.4.

Just as `bAQF`, the protocol `sAQF` restricts the propagation of the fake query to a small constant part of the network that is dependent on the network topology and on the underlying topology control mechanism. For example, using the authenticator of size around 200 bits, we are able to restrict the propagation of the fake query to less than 10 nodes given 20 compromised sensor nodes in the network with average node density 7. Moreover, in the networks organized in spanning trees with moderate tree degree the authenticator size can be reduced to around 100 bits.

The security properties of `sAQF` are the same as of `bAQF` (see Section 6.4.6). Furthermore, `sAQF` also exhibits several properties that are desirable for broadcast authentication. It does not need time synchronization, provides immediate authentication of an arbitrary number of messages (without the need in re-initialization), and the impact of DoS attacks is contained to a small part of the network.

6.6 Conclusions

In this work we presented probabilistic protocols `bAQF` and `sAQF` for authenticated query flooding in sensor networks. The protocols guarantee that the faked queries can only reach a small part of the network that does not depend

on the number of nodes in the network, but on the network density. Our protocols outperform many other broadcast authentication solutions while having no special requirements such as time synchronization, reliable links, or special topological structures. Moreover, are resilient to DoS attacks, they can authenticate arbitrary number of messages and do not require re-initialization. To our knowledge, `bAQF` and `sAQF` are the first broadcast authentication protocols that combine all above features. The security of the protocols degrades gracefully with the growing number of compromised nodes.

The price that one have to pay for all these good properties is the probabilistic nature of authentication. That is, each individual sensor node can accept a fake query with some small probability.

The utilization of `bAQF` and `sAQF` for direct access control at the base station is quite straightforward. The remote access control via base station can be organized using a probabilistic method for report authentication such as presented in Ye et al. [151]. We leave the organization of remote base station access control and of decentralized access control based on `bAQF` and `sAQF` to future work.

Chapter 7

Conclusions and Future Work

This thesis investigates how to protect the sensor network data from unauthorized access and makes two main contributions to this challenging field. Firstly, we develop the first flexible and extendable framework for defining adversaries in sensor networks. Our framework is based on systematic investigation of vulnerabilities in sensor networks, specifically focusing on physical attacks on sensor node hardware. These are all attacks that require direct physical access to the sensor nodes. Most severe attacks of this kind are also known as node capture, or node compromise.

After the adversary models took a clearly defined shape, we proceed to the investigation of access control mechanisms in sensor networks in presence of node capture attacks. We decompose the control access procedure into two logically separated phases and consider different solutions to each phase. We firstly develop deterministic solutions that offer complete security in case not more than a threshold number t of nodes is compromised, and completely break down otherwise. The developed algorithms offer nice properties for small values of t, but become prohibitively inefficient already for around 10 compromised nodes.

Trying to increase the efficiency of the results, we turn to probabilistic security. We develop access control protocols that retain all desirable properties of the deterministic solutions apart from allowing the sensor nodes to fail in recognizing an illegitimate claim for data with a small probability. The security of the developed probabilistic solutions degrades gracefully with the increasing number of compromised nodes.

We conclude that probabilistic security is more appropriate for resource-

constrained sensor networks than the traditional deterministic security in the distributed systems area. This thesis represents one of the first steps towards probabilistic, gracefully degrading security solutions. More research is needed to fully understand the impact of probabilistic security on the ubiquitous systems such as sensor networks.

Chapter 2: Security Threats and Adversary Models. Experiences with the real-world deployment of sensor network are very limited, and therefore, the development of security solutions for sensor networks is impeded by the absence of realistic adversary models. However, if the impact of the attacks in the system is not defined beforehand, developing security solutions becomes not only difficult, but also meaningless. Chapter 2 presents the first approach to formal definition of adversaries in WSNs. This approach is flexible and extendable, and is based upon the first (to our knowledge) systematical investigation of real-world attacks on sensor node hardware.

In the last few months, many other researchers gained interest in the investigation of sensor network vulnerabilities. Most notably, code injection attacks were very recently presented by Francillon and Castelluccia [59]. Attacking sensor networks is a rapidly growing research area. New attacks may facilitate extensions to the adversary model presented in this thesis. On the other hand, we believe that further formalization and refinement of adversary models for sensor networks will help to engineer less vulnerable sensor networks in the future.

Chapter 3: Access Control Issues in Wireless Sensor Networks. In Chapter 3 we firstly precisely describe the system and adversary models for the algorithms in this thesis. Using the framework from the previous chapter, we consider the adversary that can eavesdrop on messages in the entire network, and can additionally gain full control over a limited number of sensor nodes. We then identify and precisely define two phases of access control to sensor network data. During the *user access control* the user proves its legitimacy to the base station or to the surrounding sensor nodes. This phase belongs to the realm of outside security where the sensor network is considered as a single entity communicating with the user. In the phase of *authenticated querying* user's query is forwarded into the network accompanied by an *authenticator*. The authenticator enables the nodes that did not participate in the user access control phase to verify the legitimacy of the query and belongs to the area of inside security, i.e., secure communication between the individual sensor nodes.

We also consider design space for access control solutions in sensor networks, and identify the following four access control methods: direct versus remote *base station* access control, and direct versus remote *decentralized* access control. In the first two solutions, the base station acts as a trusted entity. In the last two cases, access control is organized using cooperation between sensor nodes.

The core of solutions to authenticated querying constitute *broadcast authentication* protocols where one sender is required to authenticate its messages to the multiple receivers in presence of an attacker that can gain full control over some receivers.

Chapter 4: Background And Related Work on WSN Security. In Chapter 4 we present background information on sensor network architecture and security. We consider query processing and dissemination in sensor networks, access control in general, and also specific sensor network security topics: key establishment, broadcast authentication and secure code update, report authentication and authenticated aggregation.

We identify the desirable properties for broadcast authentication that are most suitable for authenticated querying. These properties are (see also Section 4.6.1): low computation overhead, low communication overhead, independence on time synchronization and on network topology, robustness to packet loss, the possibility to send messages at irregular times, immediate message authentication and authentication of unbounded number of messages.

The study of related work on broadcast authentication in sensor networks shows that none of the previously developed protocols satisfies all these requirements. Thus, the next goal of this thesis is the development of a broadcast authentication protocol that satisfies all or most of the requirements.

Chapter 5: t-Robust Access Control in WSNs. We firstly look into development of novel access control solutions under the assumption that the adversary can compromise t or less sensor nodes. This restriction on the adversary seems to be justified at first glance, as the node compromise incurs some cost, and therefore, the adversary has to stop sometime, because otherwise it might be cheaper deploy adversary's own sensor network in order to get the required data.

The developed tree-based interleaved authenticated querying scheme has some advantages. It is fully secure as long as not more than t nodes are

compromised, and it satisfies some of the above requirements on broadcast authentication. The scheme does not require time synchronization, can authenticate messages sent at irregular time, provides immediate message authentication and can authenticate an unbounded number of messages. It also exhibits low computation and communication overhead for small values of t, such as $t = 3$. We also show how to utilize the tree-based scheme for access control.

However, the developed scheme also has some disadvantages. It depends on the stability of the network topology as it works along a shortest-path spanning tree of the network, it does not tolerate message loss, and, more importantly, its communication cost gets prohibitively high already for $t = 10$.

Moreover, the threshold adversary model has a serious drawback, as it is difficult to justify the choice of t. Why should not the adversary be able to capture 4 nodes if it has already captured 3 nodes? In general, we think that the threshold adversary model that was very popular in traditional distributed systems (see e.g. Byzantine Agreement [92]), does not apply to the sensor networks due to their large size and unattended operation of small nodes that are relatively easy to capture.

Nevertheless, developed schemes can be used for sensor networks where node compromise is considered an unlikely event, and thus t can be set to a small value.

Chapter 6: Gracefully Degrading Access Control in WSNs. As the threshold adversary can not be applied to all types of sensor networks, we turn our attention to gracefully degrading protocols. Their security degrades with the increasing strength of the adversary. That is, there is no distinct threshold on the number of captured nodes, but the ability of the protocols to reach their security goals decreases with the increasing number of captured nodes.

We develop two probabilistic protocol for broadcast authentication: `bAQF` (basic Authenticated Query Flooding) that works with ID-based random key predistribution [158], and `sAQF` (simple Authenticated Query Flooding) that uses its own key predistribution method. Both protocols add probabilistically verifiable authenticators to broadcast messages. Each node can efficiently verify a legitimate authenticator. However, the nodes can also fail to recognize a fake authenticator with some small probability. Therefore, some nodes forward the fake query to their neighbors. This process is called flooding. However, if the probability of forwarding a fake query is sufficiently small, the fake query will only travel into the network along a

short path, as any node that recognizes it as a fake will drop the query.

The probability for the adversary to successfully fake an authenticator depends on the number of nodes captured by the adversary and on the authenticator size. Moreover, as long as the adversary did not capture too many nodes, the security of the scheme degrades gracefully. For example, `sAQF` can restrict the propagation of the fake query using the authenticator size of 200 bits to less than 5 nodes on average, independently of the network size, in case the adversary captured around 20 nodes. And even if the number of captured nodes grows up to 30, the propagation of the fake query is still limited to less than 10 nodes.

The security of the AQF schemes depends on the network topology. The schemes work well in sparse networks. Very dense networks where the nodes have more than 10 neighbors on average, should be organized into a tree structure with the moderate number of children per node. However, there is no requirement on particular network topology as long as the networks have an appropriate density.

To summarize, we develop probabilistic, gracefully degrading broadcast authentication protocols. These protocols are the first to satisfy all requirements formulated in Chapter 3. They provide low computation and communication overhead, independence on time synchronization and on network topology, robustness to packet loss, the possibility to send messages at irregular times, immediate message authentication and authentication of unbounded number of messages.

The price that one pays for these good properties is probabilistic message authentication. We argue that probabilistic security is more appropriate for sensor networks than deterministic approaches.

The validity of this assumption has to be verified. This is one of the most important areas of future work in security for wireless sensor networks.

Bibliography

[1] Andreas Achtzehn, Zinaida Benenson, and Christian Rohner. Implementing agreement protocols in sensor networks. In *2nd IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'06)*, October 2006.

[2] Edward G. Amoroso. *Fundamentals of computer security technology.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[3] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, Inc., 2001.

[4] Ross J. Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *ICNP*, pages 206–215, 2004.

[5] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136, London, UK, 1998. Springer-Verlag.

[6] Atmel Corp. AT45DB041B datasheet. Atmel document no. 3443, available at `http://www.atmel.com/dyn/resources/prod_documents/doc3443.pdf`.

[7] Atmel Corp. ATmega128 datasheet. Atmel document no. 2467, available at `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[8] Sasikanth Avancha, Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. Secure sensor networks for perimeter protection. *Comput. Netw.*, 43(4):421–435, 2003.

[9] Gildas Avoine, Felix Gärtner, Rachid Guerraoui, and Marko Vukolic. Gracefully degrading fair exchange with security modules. In *5th European Dependable Computing Conference (EDCC '05)*, volume 3463 of *LNCS*, pages 55–71. Springer-Verlag, 2005.

[10] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of Network and Distributed System Security Symposium 2002 (NDSS'02)*, San Diego, CA, February 2002.

[11] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In *Security in Pervasive Computing, Third International Conference, SPC 2006*, Lecture Notes in Computer Science 3934, pages 104–118. Springer, 2006.

[12] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[13] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *PODC '83: Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30, New York, NY, USA, 1983. ACM.

[14] Zinaida Benenson. Authenticated queries in sensor networks. In *2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'05)*, July 2005.

[15] Zinaida Benenson. *Algorithms for Sensor and Ad Hoc Networks*, chapter Lower Bounds. LNCS 4621. Springer, 2007.

[16] Zinaida Benenson, Markus Bestehorn, Erik Buchmann, Felix Freiling, and Marek Jawurek. Query Dissemination with Predictable Reachability and Energy Usage in Sensor Networks. In *7th International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW 2008)*, Nice, France, September 2008.

[17] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. Simple evasive data storage in sensor networks. In *IASTED PDCS*, pages 779–784, 2005.

[18] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. Advanced evasive data storage in sensor networks. In *8th International Conference on Mobile Data Management (MDM'07)*, May 2007.

[19] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. *Wireless Sensor Network Security*, chapter Vulnerabilities and Attacks in Wireless Sensor Networks, pages 22–43. IOS Press, 2008.

[20] Zinaida Benenson and Felix C. Freiling. On the feasibility and meaning of security in sensor networks. In *Proceedings 4th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze", Zurich, Switzerland*, March 2005.

[21] Zinaida Benenson, Felix C. Freiling, Ernest Hammerschmidt, Stefan Lucks, and Lexi Pimenidis. Authenticated query flooding in sensor networks. In *Security and Privacy in Dynamic Environments, Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006)*, pages 38–49, 2006.

[22] Zinaida Benenson, Felix C. Freiling, Ernest Hammerschmidt, Stefan Lucks, Lexi Pimenidis, Christian Rohner, and Dirk Stegemann. Authenticated query flooding in sensor networks. (journal version, submitted to publication), 2007.

[23] Zinaida Benenson, Felix C. Freiling, Birgit Pfitzmann, Christian Rohner, and Michael Waidner. Verifiable agreement: Limits of non-repudiation in mobile peer-to-peer ad hoc networks. In Levente Buttyán, Virgil D. Gligor, and Dirk Westhoff, editors, *3rd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'06)*, volume 4357 of *Lecture Notes in Computer Science*, pages 165–178. Springer, 2006.

[24] Zinaida Benenson, Felix C. Gärtner, and Dogan Kesdogan. An algorithmic framework for robust access control in wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN)*, January 2005.

[25] Zinaida Benenson, Nils Gedicke, and Ossi Raivio. Realizing robust user authentication in sensor networks. In *Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, 2005.

[26] Zinaida Benenson, Ulrich Kühn, and Stefan Lucks. Cryprographic insecurity metrics. In Irene Eusgeld, Felix C. Freiling, and Ralf Reussner, editors, *Dependability Metrics*, LNCS 4909. Springer, 2008.

[27] FU Berlin. ScatterWeb Embedded Sensor Board. Online at `http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/esb/`.

[28] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Optimal early stopping in distributed consensus (extended abstract). In *WDAG '92:*

Proceedings of the 6th International Workshop on Distributed Algorithms, pages 221–237, London, UK, 1992. Springer-Verlag.

[29] Erik-Oliver Blaß, Joachim Wilke, and Martina Zitterbart. A Security–Energy Trade-Off for Authentic Aggregation in Sensor Networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Extended Abstract*, pages 135–137, Washington D.C., USA, September 2006. ISBN: 1-4244-0732-X.

[30] Erik-Oliver Blaß, Joachim Wilke, and Martina Zitterbart. A security–energy trade-off for authentic aggregation in sensor networks. In *Proceedings of the Second IEEE Workshop on Wireless Mesh Networks (WiMesh 2006)*, pages 135–137, Washington D.C., USA, September 2006. IEEE Press.

[31] Dan Boneh, Glenn Durfee, and Matthew K. Franklin. Lower bounds for multicast message authentication. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 437–452. Springer-Verlag, 2001.

[32] Markus de Brün. Authenticated query forwarding in wireless sensor networks. Master's thesis, RWTH Aachen University, 2008.

[33] Matthias R. Brust, Adrian Andronache, Steffen Rothkugel, and Zinaida Benenson. Topology-based clusterhead candidate selection in wireless ad-hoc and sensor networks. In *2nd International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE'07)*, 2007.

[34] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[35] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM'99*, volume 2, pages 708–716, New York, NY, March 1999. IEEE.

[36] Haowen Chan and Adrian Perrig. Efficient security primitives derived from a secure aggregation algorithm. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 521–534, New York, NY, USA, 2008. ACM.

[37] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, May 2003.

[38] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 278–287, New York, NY, USA, 2006. ACM.

[39] Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 311–320, New York, NY, USA, 2006. ACM.

[40] Chipcon AS. CC1000 datasheet. Available at `http://www.chipcon.com/files/CC1000_Data_Sheet_2_3.pdf`.

[41] Chipcon AS. CC2420 datasheet. Available at `http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf`.

[42] Crossbow, Inc. MICA2 data sheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[43] Crossbow, Inc. MPR, MIB user's manual. Available at `http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf`.

[44] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *2nd IEEE International Workshop on Information Processing in Sensor Networks (IPSN 2003)*, April 2003.

[45] Jing Deng, Richard Han, and Shivakant Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 292–300, New York, NY, USA, 2006. ACM.

[46] Digital signature standard (dss). FIPS PUB 186-2, January 2000. Available at http://csrc.nist.gov/groups/ST/toolkit/digital_signatures.html.

[47] Tassos Dimitriou and Dimitris Foteinakis. Secure and efficient in-network processing for sensor networks. In *First Workshop on Broadband Advanced Sensor Networks (BaseNets)*, 2004.

[48] Tassos Dimitriou, Ioannis Krontiris, and Fotios Nikakis. Fast and scalable key establishment in sensor networks. In Shashi Phoha, Thomas F. La Porta, and Christopher Griffin, editors, *Sensor Network Operations*, pages 557–570. Wiley-IEEE Press, 2006.

[49] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[50] Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.

[51] Qi Dong, Donggang Liu, and Peng Ning. Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 2–12, New York, NY, USA, 2008. ACM.

[52] D. M. Doolin and N. Sitar. Wireless sensors for wildre monitoring. In *SPIE Symposium on Smart Structures and Materials*, March 2005.

[53] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[54] Prabal K. Dutta, Jonathan W. Hui, David C. Chu, and David E. Culler. Securing the deluge network programming system. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 326–333, New York, NY, USA, 2006. ACM.

[55] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM Press, 2002.

[56] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Chapman and Hall, 3 edition, 1971.

[57] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT '99: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 232–246, London, UK, 1999. Springer-Verlag.

[58] Milan Fort, Felix C. Freiling, Lucia Draque Penso, Zinaida Benenson, and Dogan Kesdogan. Trustedpals: Secure multiparty computation implemented with smart cards. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *11th European Symposium on Research in Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, pages 34–48. Springer, 2006.

[59] Aurélien Francillon and Claude Castelluccia. Code injection attacks on harvard-architecture devices. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 15–26, New York, NY, USA, 2008. ACM.

[60] Keith B. Frikken and IV Joseph A. Dougherty. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 68–76, New York, NY, USA, 2008. ACM.

[61] Saurabh Ganeriwal, Christina Pöpper, Srdjan Čapkun, and Mani B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–35, 2008.

[62] Felix C. Gärtner. Byzantine Failures and Security: Arbitrary is not (always) Random. Technical Report LPD-REPORT-2003-009, EPFL, 2003.

[63] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks - revisited. In *ESAS*, pages 2–18, 2004.

[64] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management*, pages 45–62. Springer-Verlag, 2003.

[65] Virgil Gligor. On the evolution of adversary models in security protocols: from the beginning to sensor networks. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer*

and communications security, pages 3–3, New York, NY, USA, 2007. ACM.

[66] Dieter Gollmann. *Computer security*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[67] Li Gong. Variations on the themes of message freshness and replay — or the difficulty of devising formal methods to analyze cryptographic protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 131–136. IEEE Computer Society Press, 1993.

[68] Mark G. Graff and Kenneth R. van Wyk. *Secure Coding: Principles and Practices*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[69] Qijun Gu and Rizwan Noorani. Towards self-propagate mal-packets in sensor networks. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 172–182, New York, NY, USA, 2008. ACM.

[70] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)*, Kauai, March 2005.

[71] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *CHES2004*, volume 3156 of *LNCS*, 2004.

[72] Carl Hartung, James Balasalle, and Richard Han. Node compromise in sensor networks: The need for secure systems. Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado at Boulder, January 2005.

[73] Maurice P. Herlihy and J. D. Tygar. How to make replicated data secure. In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 379–391. Springer-Verlag, 1988, 16–20 August 1987.

[74] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[75] Michael Howard and David LeBlanc. *Writing secure code*. Microsoft Press, pub-MICROSOFT:adr, second edition, 2003.

[76] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 384. IEEE Computer Society, 2003.

[77] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38, 2005.

[78] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA, 2004. ACM.

[79] Joengmin Hwang and Yongdae Kim. Revisiting random key predistribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52. ACM Press, 2004.

[80] Sangwon Hyun, Peng Ning, An Liu, and Wenliang Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. *International Conference on Information Processing in Sensor Networks*, pages 445–456, 2008.

[81] 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.

[82] Information Technology Security Evaluation Criteria (ITSEC), Version 1.2. Commission of the European Communities, June 1991.

[83] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley-Interscience, 2007.

[84] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[85] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols*, September 2003.

[86] Donnie H. Kim, Rajeev Gandhi, and Priya Narasimhan. Castor: Secure code updates using symmetric cryptosystems. *Real-Time Systems Symposium, IEEE International*, 0:479–488, 2007.

[87] François Koeune and François-Xavier Standaert. *Foundations of Security Analysis and Design III*, chapter A Tutorial on Physical Security and Side-Channel Attacks. LNCS 3655. Springer-Verlag, 2005.

[88] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. In *Proceedings of the Ninth International Conference on Network Protocols (ICNP'01)*, page 251. IEEE Computer Society, 2001.

[89] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. In *Design, Analysis, and Simulation of Distributed Systems, SpringSim 2005*, April 2005.

[90] Ioannis Krontiris, Zinaida Benenson, Felix C. Freiling, Tassos Dimitriou, and Thanassis Giannetsos. Cooperative intrusion detection in wsn. In *6th European Conference on Wireless Sensor Networks (EWSN'09)*, 2009.

[91] Ioannis Krontiris and Tassos Dimitriou. Authenticated in-network programming for wireless sensor networks. In *In Proceedings of the 5th International Conference on AD-HOC Networks and Wireless (ADHOC-NOW 2006)*, 2006.

[92] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[93] Patrick E. Lanigan, Rajeev Gandhi, and Priya Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 53, Washington, DC, USA, 2006. IEEE Computer Society.

[94] Daniel Lehmann and Michael O. Rabin. On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *POPL '81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 133–138, New York, NY, USA, 1981. ACM.

[95] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61. ACM Press, 2003.

[96] Donggang Liu and Peng Ning. Multilevel μtesla: Broadcast authentication for distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(4):800–836, 2004.

[97] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[98] Donggang Liu, Peng Ning, Sencun Zhu, and Sushil Jajodia. Practical broadcast authentication in sensor networks. In *MOBIQUITOUS '05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 118–132, Washington, DC, USA, 2005. IEEE Computer Society.

[99] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[100] Mark Luk, Adrian Perrig, and Bram Whillock. Seven cardinal properties of sensor network broadcast authentication. In *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 147–156, New York, NY, USA, 2006. ACM.

[101] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[102] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Network*, Santa Clara, California, October 2004.

[103] Geoff Martin. An evaluation of ad-hoc routing protocols for wireless sensor networks. Master's thesis, University of Newcastle upon Tyne, May 2004.

[104] Ueli Maurer. Secure multi-party computation made simple. *Discrete Appl. Math.*, 154(2):370–381, 2006.

[105] Gary McGraw and John Viega. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison-Wesley, September 2001.

[106] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, FL, 1997.

[107] Ralph Merkle. Protocols for public key cryptography. In *IEEE Symposium on Security and Privacy*, 1980.

[108] Microchip Technology. 24AA64/24LC64 datasheet. Available at `http://ww1.microchip.com/downloads/en/DeviceDoc/21189K.pdf`.

[109] RF Monolithics. TR1001 datasheet. Available at `http://www.rfm.com/products/data/tr1001.pdf`.

[110] moteiv Corp. Telos revision B datasheet. Available at `http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf`.

[111] moteiv Corp. Tmote Sky datasheet. Available at `http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf`.

[112] `http://mspgcc.sourceforge.net/`.

[113] J. Ni and S. Chandler. Connectivity properties of a random radio network. *IEE Communications*, 141:389–296, August 1994.

[114] Seung-Jong Park, Ramanuja Vedantham, Raghupathy Sivakumar, and Ian F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 78–89, New York, NY, USA, 2004. ACM.

[115] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.

[116] Holger Peine. Rules of thumb for secure software engineering. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 702–703, New York, NY, USA, 2005. ACM Press.

[117] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[118] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. Spins: security protocols for sensor netowrks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 189–199. ACM Press, 2001.

[119] Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. Analysis of wireless sensor networks for habitat monitoring. In Cauligi S. Raghavendra, Krishna M. Sivalingam, and Taieb Znati, editors, *Wireless Sensor Networks*, 2005.

[120] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.

[121] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[122] K. Rannenberg, A. Pfitzmann, and G. Müller. IT security and multilateral security. In G. Müller and K. Rannenberg, editors, *Multilateral Security in Communications – Technology, Infrastructure, Economy*, pages 21–29. Addison-Wesley, 1999.

[123] Kui Ren, Kui Ren, Wenjing Lou, Wenjing Lou, Yanchao Zhang, and Yanchao Zhang. Multi-user broadcast authentication in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 223–232, 2007.

[124] I. Rodhe, C. Rohner, and A. Achtzehn. n-LQA: n-Layers Query Authentication in Sensor Networks. *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–6, Oct. 2007.

[125] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. SCUBA: Secure Code Update By Attestation in sensor networks. In *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*, pages 85–94, New York, NY, USA, 2006. ACM.

[126] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6), December 2004.

[127] Sergei P. Skorobogatov. Semi-invasive attacks - a new approach to hardware security analysis. Technical report, University of Cambridge, Computer Laboratory, April 2005. Technical Report UCAM-CL-TR-630.

[128] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 2–12, London, UK, 2003. Springer-Verlag.

[129] Fred Stann, John Heidemann, Rajesh Shroff, and Muhammad Zaki Murtaza. Rbp: robust broadcast propagation in wireless networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 85–98, New York, NY, USA, 2006. ACM.

[130] STMicroelectronics. M25P80 datasheet. Available at `http://www.st.com/stonline/products/literature/ds/8495.pdf`.

[131] Kun Sun, Peng Ning, and Cliff Wang. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 264–277, New York, NY, USA, 2006. ACM.

[132] Texas Instruments. Features of the MSP430 bootstrap loader (rev. B). TI Application note SLAA089B, available at `http://www-s.ti.com/sc/psheets/slaa089b/slaa089b.pdf`.

[133] Texas Instruments. MSP430 F149 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f149.pdf`.

[134] Texas Instruments. MSP430 F1611 datasheet. Available at `http://www-s.ti.com/sc/ds/msp430f1611.pdf`.

[135] Texas Instruments. MSP430x1xx family: User's guide. TI Application note SLAU049E, available at `http://www-s.ti.com/sc/psheets/slau049e/slau049e.pdf`.

[136] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.*, 8(2/3):153–167, 2002.

[137] L. Venkatraman and D. Agrawal. A novel authentication scheme for ad hoc networks. In *IEEE Wireless Communications and Networking Conference (WCNC 2000)*, volume 3, pages 1268–1273, 2000.

[138] John Viega, Matt Messier, and Gene Spafford. *Secure Programming Cookbook for C and C++*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[139] Harald Vogt. Exploring message authentication in sensor networks. In *Security in Ad-hoc and Sensor Networks (ESAS), First European Workshop*, volume 3313 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2004.

[140] Harald Vogt, Matthias Ringwald, and Mario Strasser. Intrusion detection and failure recovery in sensor nodes. In *Tagungsband INFORMATIK 2005, Workshop Proceedings*, LNCS, Heidelberg, Germany, September 2005. Springer-Verlag.

[141] Victor L. Voydock and Stephen T. Kent. Security mechanisms in high-level network protocols. *ACM Comput. Surv.*, 15(2):135–171, 1983.

[142] David Wagner. Resilient aggregation in sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87. ACM Press, 2004.

[143] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 1–11, New York, NY, USA, 2002. ACM Press.

[144] Ronghua Wang, Wenliang Du, and Peng Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 71–79, New York, NY, USA, 2007. ACM.

[145] Ronald Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. TinyPK: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64. ACM Press, 2004.

[146] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205, New York, NY, USA, 2002. ACM.

[147] A. Wood, J. Stankovic, and S. Son. JAM: A mapping service for jammed regions in sensor networks. In *In Proceedings of the IEEE Real-Time Systems Symposium*, December 2003.

[148] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[149] Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wirel. Netw.*, 10(2):169–181, 2004.

[150] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.

[151] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE Jounal on Selected Areas in Communications, Special Issue on Self-organizing Distributed Collaborative Sensor Networks*, 2005.

[152] Yungjung Yi, Mario Gerla, and Taek Jin Kwon. Efficient flooding in ad hoc networks: a comparative performance study. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1059–1063, May 2003.

[153] Sungjune Yoon, Tomoyuki Asano, Masafumi Kusakawa, Hyunrok Lee, and Kwangjo Kim. Hybrid multi-user broadcast authentication for wireless sensor networks. In *The 2008 Symposium on Cryptography and Information Security (SCIS 2008)*, Japan, January 2008.

[154] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[155] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, Nov./Dec. 1999.

[156] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 62–72. ACM Press, 2003.

[157] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy*, pages 259–271, 2004.

[158] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *IEEE International Conference on Network Protocols*, November 2003.

[159] Martina Zitterbart and Erik-Oliver Blaß. An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks. In *ACM Symposium on Information, Computer and Communications Security*, pages 303–310, Taipei, Taiwan, March 2006. ISBN 1-59593-272-0.