Diss. ETH Nr. 18174

Protocols for Secure Communication in Wireless Sensor Networks

A dissertation submitted to
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH
(ETH ZÜRICH)

for the degree of
Doctor of Sciences
(Dr. sc. ETH Zürich)

presented by
Harald Vogt
Diplom-Informatiker, Universität Ulm, Germany
born November 30, 1970
citizen of Switzerland (Basel BS) and Germany

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Joachim Posegga, co-examiner

2009

Abstract

Wireless sensor networks are comprised of large numbers of resource-constrained and wirelessly communicating computing devices. Advances in computing and communication technology have made it possible to integrate sensing capabilities, wireless communication interfaces, and microprocessors into tiny devices that allow to embed compuational power in arbitrary environments. The applications of wireless sensor networks range from surveillance and environmental monitoring to healthcare and the provisioning of context information for computing applications. Many of these applications have a direct impact on the welfare of human beings or are of high economic significance. Security breaches might lead to grave consequences, so it is important to protect wireless sensor networks against such threats.

The specific characteristics of wireless sensor networks make them vulnerable to attacks on their communication channels and their hardware. Cryptographic mechanisms can be employed to protect against some of the possible attacks: eavesdropping on messages is countered by encryption, and the injection of messages by the attacker is prevented by authentication. Unfortunately, direct physical access to the sensor nodes allows an attacker to manipulate them almost arbitrarily. In particular, nodes could be compromised and then made to execute malicious code injected by the attacker. Tamper resistance mechanisms applied to the nodes' hardware, concealment, surveillance and other techniques may be used to mitigate such attacks. However, they cannot be completely prevented and therefore, any communication security scheme being used must be sufficiently resilient to tolerate a certain amount of compromised nodes. Consequently an important objective is to limit the impact of a set of compromised nodes on the legitimate operation of the network to a minimum.

This objective can optimally achieved by cryptographic mechanisms that establish a direct security relationship between communicating end-points. This limits the influence that a single compromised node has to its own resources. Thereby, it cannot tamper with messages that originate at other nodes. However, such mechanisms are overly resource demanding for many sensor nodes in terms of computational or communication complexity, especially due to the

often ad hoc and transient nature of communication relationships. Thus, novel mechanisms are required that provide a sufficient level of security while respecting the constraints in wireless sensor networks.

Our thesis is that *key pre-distribution is an appropriate technique for secret key agreement in wireless sensor networks, and that based on locally shared keys, multi-hop communication can be adequately protected using an interleaved message authentication scheme.*

We argue that combined key pre-distribution schemes provide a feasible mechanism for key agreement in wireless sensor networks. They require only simple operations on sensor nodes and their memory requirements can be adapted to the required security level and the available resources. Based on keys shared between nodes within a k-hop neighbourhood (with small k), a message authentication scheme is devised that allows for the secure transmission of messages over long distances. In particular, our contributions are:

- A key establishment scheme for pairwise key agreement that can be efficiently implemented on resource-constrained wireless sensor nodes and provides resilience against node capture attacks.
- A message authentication scheme that relies on locally shared keys and symmetric cryptographic operations only, and provides a level of security approximating that of end-to-end security mechanisms. The foundation of the scheme's security is the creation of multiple disjoint authentication paths.
- An evaluation of this authentication scheme showing that it provides at least the same security level as a general communication scheme that relies on multiple disjoint physical paths.

The proposed security mechanisms protect the integrity of messages that are exchanged within a wireless sensor network. The achievable level of security is, given an attacker with moderate strength that is only able to capture a fraction of all nodes, comparable to that provided by end-to-end security mechanisms at a significantly lower cost in terms of computational resources.

Zusammenfassung

Drahtlose Sensornetze bestehen aus einer grossen Anzahl von kleinen, kommunizierenden und ihre Umgebung durch Sensoren abtastenden Geräten, die in ihrer Ausstattung bezüglich der Rechenleistung, Kommunikationsreichweite und Energieversorgung stark beschränkt sind. Ihre Entwicklung wurde durch die Miniaturisierung der Mikroprozessoren und Fortschritten bei der Kommunikationstechnik möglich. Durch ihre geringe Grösse können sie unauffällig und wenig störend in einer Vielzahl von Umgebungen eingesetzt werden. Nützliche Anwendungen finden sich in der Überwachung im militärischen Bereich, für den Umweltschutz, aber auch für die Aufzeichung von Gesundheitsdaten. Allgemein liefern sie Kontextinformationen für übergeordnete Anwendungen. Viele dieser Anwendungen haben direkte Auswirkungen auf das Wohlergehen von Personen oder sind von wirtschaftlicher Bedeutung. Sicherheitsübertretungen können daher ernsthafte Konsequenzen nach sich ziehen, so dass es wichtig ist, drahtlose Sensornetze gegen Bedrohungen durch Angreifer zu schützen.

Die besonderen Eigenschaften drahtloser Sensornetze machen sie verwundbar gegen Angriffe, die sich auf die verwendeten Kommunikationskanäle oder ihre Hardware richten. Kryptographische Verfahren können eingesetzt werden, um sie gegen einige Angriffsarten zu schützen, zum Beispiel kann das Abhören von Nachrichten durch Verschlüsselung verhindert werden, oder das Einschleusen nicht-autorisierter Nachrichten durch Authentisierung der Sensorknoten. Der direkte Zugriff auf die Hardware der Sensorknoten erlaubt einem Angreifer, diesen praktisch beliebig zu manipulieren. Insbesondere könnte ein Angreifer Knoten zur Ausführung schadhaften Programmcodes nutzen. Mechanismen zum Schutz der Hardware, das Tarnen der Knoten oder ihre Überwachung durch externe Massnahmen können solche Angriffe teilweise verhindern, sind jedoch teuer und oft nicht anwendbar. Daher muss mit dieser Art von Angriffen immer gerechnet werden und alle Protokolle, die in einem Sensornetz verwendet werden, müssen in der Lage sein, einen gewissen Anteil von Knoten, die unter der Kontrolle des Angreifers stehen, zu tolerieren. Ziel muss sein, die Möglichkeiten zur Einflussnahme dieser Knoten zu minimieren.

Dieses Ziel kann durch kryptographische Verfahren erreicht werden, die auf Ende-zu-Ende-Sicherheit beruhen, d.h. eines Schlüssels, der nur den beiden Partien einer Kommunikationsbeziehung bekannt ist. Solche Verfahren beschränken den Einfluss eines vom Angreifer kontrollierten Knotens auf seine eigenen direkten Kommunikationswege. Ein solcher Knoten kann keinen Einfluss auf die Kommunikation anderer Knoten nehmen. Allerdings sind Ende-zu-Ende-Verfahren oftmals zu teuer, um für drahtlose Sensornetze eingesetzt zu werden, da sie zu viel Rechenkapazität oder Kommunikationsaufwand benötigen. Dies gilt insbesondere dann, wenn eine Vielzahl der Kommunikationswege nur bei Bedarf und nur selten benutzt wird. Daher werden neuartige Verfahren benötigt, die einen vergleichbaren und ausreichenden Schutz bieten, jedoch auch in Sensornetzen eingesetzt werden können, in denen nur geringe Ressourcen zur Verfügung stehen.

Die These der vorliegenden Arbeit ist, dass *die Vorverteilung von Schlüsseln ein geeignetes Verfahren für die Schlüsselerzeugung in drahtlosen Sensornetzen ist. Darüber hinaus bieten Schlüssel, die lediglich zwischen Nachbarknoten ausgetauscht werden, mit Hilfe eines Verfahrens zur verschränkten Nachrichtenauthentisierung einen geeigneten Schutz der Kommunikation in Multi-Hop-Umgebungen.*

Wir legen dar, wie zusammengesetzte Vefahren zur Schlüssel-Vorverteilung für den Schlüsselaustausch zwischen benachbarten Knoten genutzt werden können. Diese Verfahren benötigen lediglich einfache und wenige Rechenoperationen und ihr Speicherbedarf kann den verfügbaren Ressourcen sowie dem Sicherheitsbedürfnis angepasst werden. Auf der Grundlage gemeinsamer Schlüssel innerhalb einer k-Hop-Nachbarschaft (wobei k klein ist) wird ein Verfahren zur Nachrichtenauthentisierung entworfen, das die sichere Übertragung von Nachrichten auch über grosse Entfernungen erlaubt.

Der wissenschaftliche Beitrag dieser Arbeit besteht aus folgenden Punkten:

- Einem Verfahren zum paarweisen Schlüsselaustausch, das effizient auf ressourcenbeschränkten drahtlosen Sensorknoten implementiert werden kann und die Wirkung von Angriffen gegen Knoten beschränkt.

- Einem Verfahren zur Nachrichtenauthentisierung, das lediglich lokale gemeinsame Schlüssel und Operationen der symmetrischen Kryptographie erfordert. Das Verfahren bietet ein Sicherheitsniveau in Annäherung der von Ende-zu-Ende-Sicherheit. Es basiert auf der Erzeugung mehrerer disjunkter Authentisierungspfade.

- Der Bewertung des Verfahrens zur Nachrichtenauthentisierung, welche

zeigt, das es einem Verfahren, das mehrere disjunkte physische Kommunikationwege benutzt, mindestens ebenbürtig ist.

Die vorgeschlagenen Sicherheitsverfahren schützen die Integrität von Nachrichten innerhalbe eines drahtlosen Sensornetzes. Das erreichbare Sicherheitsniveau ist dem eines Ende-zu-Ende-Verfahrens vergleichbar, sofern ein Angreifer moderater Stärke angenommen wird, d.h. der Angreifer ist in der Lage, lediglich einen Teil der Knoten im Sensornetz zu kontrollieren. Die Kosten der vorgeschlagenen Verfahren sind jedoch erheblich geringer, so dass ihr Einsatz insgesamt vorteilhaft sein kann.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Contributions . . . 3
- 1.4 Thesis Outline . . . 4

2 Wireless Sensor Networks and Their Security **7**
- 2.1 Applications of Wireless Sensor Networks . . . 9
  - 2.1.1 Surveillance . . . 9
  - 2.1.2 Context Awareness . . . 10
  - 2.1.3 Other Applications . . . 11
  - 2.1.4 Security Concerns . . . 11
- 2.2 Sensor Node Characteristics . . . 13
  - 2.2.1 Computational Resources . . . 14
  - 2.2.2 Composition . . . 15
  - 2.2.3 Infrastructure . . . 16
  - 2.2.4 Topology . . . 18
  - 2.2.5 Connectivity . . . 19
  - 2.2.6 Network Size . . . 19
  - 2.2.7 Tamper Resistance . . . 20
  - 2.2.8 Further Assumptions . . . 21
- 2.3 Related Network Types . . . 21
  - 2.3.1 Ad Hoc Networks . . . 22
  - 2.3.2 Personal Area Networks . . . 24
  - 2.3.3 Peer-to-Peer Networks . . . 25
- 2.4 Related Device Types . . . 26
  - 2.4.1 Smartcards . . . 26
  - 2.4.2 RFID . . . 28
  - 2.4.3 Embedded Computers . . . 29
- 2.5 Sensor Network Models . . . 31
  - 2.5.1 The Geometric Model of Sensor Networks . . . 31

2.5.2 Small-World Networks 32
2.5.3 Fundamental Properties of Sensor Network Graphs . . 33
2.6 Simulation of Sensor Networks 35
2.6.1 Applications of Simulation 35
2.6.2 Node-based Simulation 38
2.6.3 Network-based Simulation 40
2.7 Security Requirements 41
2.7.1 Message Transmission 42
2.7.2 Routing 43
2.7.3 Access Control 44
2.7.4 Data Aggregation 45
2.7.5 Location Verification 46
2.7.6 Intrusion Detection 47
2.7.7 Intrusion Tolerance 48
2.7.8 Availability 49
2.8 Cryptography for Sensor Networks 50
2.8.1 Hash Functions 51
2.8.2 Message Authentication Codes 52
2.8.3 Symmetric Ciphers 53
2.8.4 Implementation 53
2.8.5 Bandwidth Overhead 54
2.8.6 Key Management 55
2.9 Existing Approaches to Wireless Sensor Network Security . . 57
2.9.1 Key Distribution and Agreement 58
2.9.2 Secure Communication 59
2.9.3 Secure Routing 61
2.9.4 Available Implementations 63
2.10 Summary 64

3 A Security Model for Wireless Sensor Networks 65
3.1 Attack Paths 66
3.1.1 Physical Attacks 67
3.1.2 Interface Attacks 69
3.1.3 Software-Level Attacks 70
3.2 Attack Objectives 72
3.2.1 Properties of Resources 72
3.2.2 Resource Types 73
3.2.3 Detection Evasion 77
3.3 Adversary Characteristics 78

3.3.1 Basic Assumptions 78
3.3.2 Attack Costs 78
3.3.3 Avoiding Intrusion Detection 79
3.3.4 Insider vs. Outsider 79
3.3.5 Technical Capabilities 80
3.3.6 Location-Constrained Attacks 80
3.4 The Cost of End-to-End Security 84
3.4.1 Connection Establishment 85
3.4.2 Public-Key Cryptography 86
3.4.3 Pairwise Key Distribution 87
3.5 Approximating End-to-End Security 88
3.5.1 Threat Model 89
3.5.2 Multipath Communication 89
3.5.3 Assessing the Security Level 91
3.6 Related Work 94
3.7 Summary 95

4 Key Establishment **97**
4.1 Requirements for Key Agreement 98
4.2 Random Key Pre-Distribution 98
4.2.1 A Model for Key Pre-Distribution 99
4.2.2 Pre-Distribution Phase 100
4.2.3 Identity-based Key Rings 101
4.2.4 Establishing the Common Key Set 101
4.2.5 Key Derivation 103
4.2.6 Connectivity 104
4.2.7 Resilience Against Link Key Compromise 107
4.3 Key Agreement Based on Hash Chains 110
4.3.1 Hash Chains 110
4.3.2 Single-Chain Key Agreement 110
4.3.3 Chain Key Resilience 111
4.3.4 Choosing the Length of a Hash Chain 113
4.3.5 Discussion 114
4.4 Multiple Hash Chains for Key Agreement 116
4.4.1 Key Distribution 116
4.4.2 Key Agreement 117
4.4.3 Resilience 117
4.4.4 Comparison with Random Key Pre-Distribution 119
4.5 Strengthening Random Key Pre-Distribution 120

4.5.1 A Combined Approach . . . 121
4.5.2 Resilience . . . 121
4.6 Related Work . . . 123
4.7 Summary . . . 127

5 Multipath Communication 129
5.1 Principles of Multipath Communication . . . 129
5.1.1 Single vs. Multiple Paths . . . 129
5.1.2 Advantages of Multiple Paths . . . 130
5.1.3 Path Setup . . . 131
5.2 Routing on Spanning Trees . . . 132
5.2.1 The Basic Scheme . . . 132
5.2.2 Spanning Tree Construction . . . 134
5.2.3 Addressing on Spanning Trees . . . 135
5.2.4 Message Forwarding . . . 140
5.2.5 Choice of Tree Paths . . . 141
5.3 Properties of Tree Paths . . . 141
5.3.1 Spatial Separation . . . 141
5.3.2 Path Disjointness . . . 142
5.3.3 Traffic Overhead . . . 146
5.3.4 Delivery Rate . . . 148
5.4 Security Evaluation . . . 150
5.4.1 Basic Security Model . . . 150
5.4.2 Resilience Against Attacks . . . 151
5.5 Related Work . . . 152
5.5.1 Multiple Paths for Performance . . . 152
5.5.2 Spatial Separation . . . 154
5.5.3 Threshold Security . . . 155
5.6 Summary . . . 156

6 Integrity-Preserving Communications 157
6.1 Authentication and Integrity Protection . . . 157
6.1.1 Definitions . . . 157
6.1.2 Identity, Integrity, and Authentication in WSNs . . . 160
6.2 Basic Interleaved Authentication . . . 161
6.2.1 Protocol Description . . . 162
6.2.2 Formal Specification . . . 164
6.2.3 Interaction with Routing Protocols . . . 168
6.2.4 Application to Data Aggregation . . . 174

6.3 Performance Evaluation 175
6.3.1 Single Message Overhead 176
6.3.2 Multiple Messages Overhead 177
6.4 Security Evaluation 179
6.4.1 Analytical Assessment of Resilience 179
6.4.2 Numerical Approximation 183
6.4.3 Simulation Results 185
6.4.4 Addressing Message Injection 190
6.4.5 MAC Security 192
6.4.6 Example: A Dynamic Application Scenario 193
6.5 Extended Interleaved Authentication 195
6.5.1 Protocol Description 195
6.5.2 Establishing Shortcuts 199
6.5.3 Long-Range Interleavings 200
6.5.4 Performance Evaluation 205
6.5.5 Security Evaluation 205
6.6 Comparing Interleaved and Multipath Authentication 207
6.6.1 Multiple Physical vs. Virtual Paths 207
6.6.2 Combining Authentication Techniques 208
6.7 Applications 208
6.7.1 Coupling Heterogeneous Networks 210
6.7.2 Physical-World Examples 211
6.7.3 Internet Applications 211
6.7.4 E-Mail Origin Authentication 212
6.8 Related Work 214
6.9 Summary 214

7 Conclusion **217**
7.1 Secure Communication in Wireless Sensor Networks 218
7.1.1 Key Establishment 218
7.1.2 Multiple Path Communication 218
7.1.3 Interleaved Authentication 219
7.1.4 Shortcut Authentication 221
7.2 Future Work 221

Bibliography **223**

Chapter 1

Introduction

1.1 Background

Wireless sensor networks are comprised of small, low-cost, resource-restricted devices that have the capability to communicate and interact with their environment, either only passively by sensing certain parameters, or also actively by triggering actuators. The major limiting factor in their design and operation is their restricted energy supply. Their computational capabilities are comparable to those found in embedded systems, as they are based on similar hardware designs. Sensor nodes employ wireless communication in order to exchange data with their peers. This is based mostly on radio communication, but there are some designs that employ alternatives, such as optical communication.

Initially, sensor networks were conceived for military applications, for example to track moving objects in inaccessible terrain. Due to their small size and low cost, they would allow for concealed deployment in large quantities. Recently they also have aroused interest in civilian applications, for example as monitoring tools in health applications, to collect environmental data, or to improve agriculture.

In many application scenarios, security is a critical property of sensor networks, for example in the healthcare domain where sensitive data is handled. Without adequate security, the acceptance of sensor networks as a tool will likely be limited. It is therefore important to understand what the risks are when operating a sensor network in a sensitive environment, and how these risks can be countered.

Sensor networks are vulnerable against attacks from the outside as they are using an openly accessible medium for communication. This allows an attacker to intercept messages, or interfere with transmissions, for example by jamming the radio channel or replaying intercepted messages. Insider attacks are possible if the attacker manages to inject his own sensor nodes into the network

or take control of existing nodes. Insider attacks are very powerful as they are hard to detect and allow the adversary to gain access to cryptographic key information stored on nodes, which would not be possible through an outside attack. Having control over a part of the network also allows the adversary to forge messages and sensor data, and thus directly influence the operation of the network.

In order to make a sensor network secure, mechanisms are required that make it hard for the adversary to exploit these vulnerabilities. There is a trade-off between the overhead of such mechanisms and the expected loss due to certain threats, and the operator of a sensor network is expected to choose those mechanisms that minimize both the risks and the costs. It is the task of researchers and engineers to provide such cost-effective mechanisms.

1.2 Problem Statement

A basic mechanism in a sensor network is the transmission of messages between sensor nodes. Due to the large size of a sensor network and the limited transmission range of a single node, messages that are being exchanged between distant nodes typically have to be transmitted over multiple hops. Secure message transmission requires either a security association between the communication endpoints, or the collaboration of the involved nodes.

Secure end-to-end relationships do not scale well in large sensor networks. For each relationship, a secret key must be stored. However, a single node typically only communicates with a small fraction of all nodes during the network's lifetime. This is highly inefficient since a lot of memory is blocked for storing keys that will never be used. A viable alternative is to dynamically establish end-to-end relationships only when they are needed. However, performing a key agreement over multiple hops is an expensive operation.

In light of these drawbacks, collaborative approaches seem to be more appropriate. In such approaches, nodes may rely on a small number of "friends" with whom they share a secret key. They may also enter new security relationships with nearby nodes that are dynamically joining the network, so key agreements are restricted to k-hop communications (with a small k). In order to transmit messages over multiple hops, "friendly" nodes will help each other to forward the messages and to ensure that they will not be tampered with.

Compared to end-to-end security schemes, the security guarantees of collaborative approaches are lower. Using proven cryptographic algorithms and tools, end-to-end schemes can be made *computationally* secure, i.e. no existing computing resources are sufficient to break the scheme. Collaborative approaches

on the other hand provide either *threshold* (i.e. the scheme is secure against attackers that are bounded by some constant) or *probabilistic* (i.e. given an attacker, a communication relationship is secure with a certain probability) security. It is therefore important to be able to quantify the security properties of a sensor network.

1.3 Contributions

The main contribution of this thesis is a novel protocol for protecting the integrity of messages that are transmitted over multiple hops. This protocol emerged from the observation that end-to-end message authentication is neither scalable in sensor networks nor strictly required. By restricting the security mechanisms to the preservation of the integrity of messages, good scalability is achieved while the most important properties of authentication are preserved. In fact, we provide designers with the possibility to choose the most appropriate compromise between scalability and an approximation to end-to-end security properties, with regard to the application scenario at hand and the implied threat model. The provided security guarantees are of probabilistic nature and are quantified with regard to fundamental network properties, namely the probability of message compromise and the fraction of nodes being able to participate in a consensus protocol.

The proposed protocol has similarities with multipath message transmission, an approach known to provide threshold security. We provide a characterisation of both approaches in order to clearly separate them from each other. We also propose a new technique for setting up multiple, node-disjoint paths in a sensor network that avoids the deficiencies of existing approaches. It is independent of the topology of the network and provides a good physical separation of paths, which is important with regard to the assumed adversary model.

A prerequisite of many security mechanisms is the ability of nodes to agree on a common secret key. Multiple schemes have been proposed that are based on probabilistically distributed shares of a common key pool. Two nodes agree on a common key by determining their common subset of their shares. It also allows nodes to prove to each other that they are legitimate members of the network and thereby effectively defeating outsider attacks. We describe a generalization of these schemes that allows to increase their resilience significantly against insider attacks without requiring any additional storage and with only minimal computational overhead.

1.4 Thesis Outline

Chapter 2 introduces the architecture of wireless sensor networks and their fundamental properties that are important with regard to security. The main vehicle is a graph-based model of sensor networks that will be the basis for the evaluation of the proposed security protocols.

In chapter 3, we describe attacks on sensor networks and define and motivate an adversary model that serves as the context for the proposed security mechanisms. The main element of this adversary model is *node capture*, i.e. the adversary takes complete control of a number of sensor nodes. As end-to-end security constitutes the most powerful security model for communication, we describe the possibilities to achieve this in sensor networks, and the associated costs. We then describe an approximative model and methods for its evaluation.

Chapter 4 describes key agreement in sensor networks, a prerequisite for the following security schemes. The focus is on probabilistic key pre-distribution, for which we introduce a generalization that increases the resilience of these schemes with regard to the previously defined adversary model. The proposed technique is based on hash functions and is thus efficiently implementable on sensor nodes.

In the following chapter 5, secure multipath communication is introduced and a novel technique for the efficient construction of multiple, node-disjoint paths between arbitrary nodes is presented. This technique addresses the requirements and constraints of sensor networks and is adapted to the assumed adversary model.

In chapter 6, a novel interleaved message authentication scheme is introduced, which is based on shared secret keys between neighbouring sensor nodes. It provides integrity preservation for messages on multihop paths. Starting with a basic scheme, several extensions are described that compensate for the deficiencies of the basic scheme.

Chapter 7 evaluates the achieved results, discusses potential application areas for the proposed techniques, and proposes directions for future research.

In connection with the course of the work on the presented thesis, the following student theses have been conducted:

- Patrick Moor, Mario Strasser. Schlüsselvereinbarung in Sensornetzen (Semester thesis)
- Claudio Munari. Simulation und Visualisierung eines sicheren Kommunikationsprotokolls (Semester thesis)

- Mario Strasser. Intrusion Detection and Failure Recovery in Sensor Networks (Master thesis)
- Claudio Munari. Multipfad-Protokolle zur sicheren Kommunikation in Ad-hoc und Sensornetzen (Master thesis)

The following publications emerged from this work:

- Harald Vogt. Integrity Preservation for Communication in Sensor Networks, Technical Report 434, ETH Zürich, Institute for Pervasive Computing, 2004
- Harald Vogt. Exploring Message Authentication in Sensor Networks. In *Proceedings of the 1st European Workshop on Security in Ad-hoc and Sensor Networks (ESAS 2004)*, LNCS 3313, Springer-Verlag, 2005
- Harald Vogt. Increasing Attack Resiliency of Wireless Ad Hoc and Sensor Networks. In *25th IEEE International Conference on Distributed Computing Systems, Workshops*, IEEE, 2005
- Harald Vogt. Small Worlds and the Security of Ubiquitous Computing. In *First International Workshop on Trust, Security and Privacy for Ubiquitous Computing (TSPUC): Proceedings of 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, IEEE, 2005
- Harald Vogt, Matthias Ringwald, Mario Strasser. Intrusion Detection and Failure Recovery in Sensor Nodes. In *Tagungsband INFORMATIK 2005, Workshop Proceedings*, volume P-68 of *Lecture Notes in Informatics*, Gesellschaft für Informatik, 2005

Chapter 2

Wireless Sensor Networks and Their Security

Recent advancements in the miniaturization and commoditization of electronics, computing and communication technology have created a trend towards *ubiquitous computing*. It is anticipated both in academic research and industry that computational devices with communication and sensing capabilities will soon be intertwined with many products and integrated into many industrial and business processes. This would allow the creation of new business models as well as new and improved services and products. This development has already started to become reality with the increasing deployment of radio frequency identification (RFID) technology in many areas.

Driven by the same technological forces, *wireless sensor networks* (WSN) have emerged as a new type of network architecture. Such networks are comprised of small and inexpensive devices with the capability to interact with their environment, mostly through passive means allowing them to sense certain environmental parameters, but sometimes also actively by triggering actuators. The resources of these devices are quite limited due to the requirements of small size and low cost. The major restricting factor is their energy supply. In most cases, their batteries cannot be recharged since the devices are inaccessible after deployment. External power sources, such as solar panels, can compensate for this deficiency only partially. It is therefore paramount for designers to build sensor networks in a way that minimizes their energy consumption.

As a consequence of this principle, the computational and communication capabilities of sensor devices are rather limited. Current prototypes are based on microcontrollers typically used in embedded systems. These processors draw much less power than those used in desktop and server systems, but they also lack many of their desirable features, such as sophisticated memory management, parallel instruction execution, and high clock frequency. For communication purposes, low-power radio communication is typically used with a

transmission range between some centimeters up to one hundred meters.

Since single sensor devices are so much restricted, the usefulness of a sensor network only materializes through the cooperation of a large number of these devices. A large number allows covering a large geographical area, making up for the limited sensing range of a single device. It also allows to compensate for device failures (which are likely to occur during the lifetime of the overall network) by another device stepping into the position of a failed one. Through cooperation, they can forward each other's messages to remote destinations, and combine the sensor data they have captured in order to provide higher-order information.

Sensor networks are distributed systems, comprised of either homogeneous devices in the simple case, or various kinds of devices with different capabilities in more complex scenarios. As distributed, networked systems, they are prone to many kinds of failures just as other distributed systems, such as link and host failures that hamper the reliability of these systems, and malicious intruders that try to inflict harm or take illegitimate advantage of them. In conventional distributed systems, the ultimate protection against intrusions is the physical separation of a system from its environment, providing only a narrow, well-defined interface for users to interact with the system. Access to this interface is under tight control and only made available to authorized parties. This approach is unrealistic for sensor networks, since they use wireless communication, which exposes their communication interface, and they are often operated in publicly accessible spaces, thereby enabling physical access to the sensor nodes.

The applicability of conventional system security mechanisms to sensor networks is limited. Public key cryptography, intrusion detection, or sophisticated access control often exceed the available resources. As sensor devices are placed into environments that are open to physical access by potential adversaries, a sensor network cannot be isolated completely. Full tamper resistance of sensor nodes is not affordable in most cases, which in principle leaves them open to direct physical manipulations. (However, their large number would make it costly for an attacker to corrupt *all* of them.)

Effective security mechanisms thus have to take into account the large number of nodes in sensor networks, their probabilistic nature due to failed or compromised components, and their inherent limitations regarding physical protection and tight resource restrictions. Establishing and maintaining the security of a wireless sensor network is challenging due to several reasons:

- The use of a wireless communication medium makes covert access to the transmitted data possible. Data injection or manipulation is also possible

without tampering with any physical infrastructure.

- Sensor networks are often operated in open, publicly accessible space. This enables physical access to sensor nodes.

- The implementation of security mechanisms, such as cryptography or intrusion detection, must respect the limited computational and energy resources, and the limited communication bandwidth.

2.1 Applications of Wireless Sensor Networks

Sensor networks are tools to bridge the gap between the physical and the virtual world. They allow to automatically collect information about physical phenomena, immediately process this information and transfer the results into background information systems. This processing delivers high-level information according to the application's requirements. Sensor nodes organize themselves autonomously, work in a collaborative manner, and are designed for energy-efficiency. This allows it to monitor large geographical areas or inaccessible spaces over long periods of time without the need of human intervention. A comprehensive survey of sensor network architectures and applications can be found in [2].

2.1.1 Surveillance

An important application class for sensor networks is their use for the protection of assets or people. By definition, sensor networks are highly suitable for data collection. This capability can be exploited for surveillance purposes, most importantly for the detection of intruders [62] or physical security breaches, or more generally for "perimeter protection" [8], where an area around a threatened entity (which is possibly moving) is under surveillance. Today, there already exists an industry for surveillance tools such as video cameras, motion detectors, burglar alarms, etc. Sensor networks add two new qualities to such systems, first the large number of tiny devices that can be deployed in an ad hoc manner, and second the self-organization of these devices for communication and configuration. Thus, with the wide-spread availability of wireless sensor network technology, it will become possible to put areas under surveillance that would not be economically feasible today. Besides the potential gains to (both public and private) security, this will have severe impact on people's privacy and may also have severe consequences for politics and society [26].

2.1.2 Context Awareness

Pervasive and ubiquitous computing concepts assume a tight relationship between computing devices and human users. Since the behaviour of humans is closely related to the *context* in which current activities take place, the notion of this concept has gained much attention in this research area. It is assumed that by understanding context, applications can adapt their behaviour to the specific needs of the human user and his environment. A prominent example is the mobile phone that autonomously recognizes situations where an audible ringtone is inappropriate, for example a work team meeting or sitting in a cinema.

Sensor networks can be used as a tool for deriving contextual information. For example, office rooms could be equipped with sensor networks that automatically determine whether a meeting is in progress. Another example would be body-area sensor networks that determine in which activity the person is currently involved. Other devices in the vicinity could then make use of this information and adapt their behaviour accordingly.

It is the subject of ongoing research how to derive higher-level, contextual information from basic data such as lightning conditions, biological parameters, temperature, time, or acceleration, for which sensors exist. There is no universally valid definition of what exactly should be included in the notion of context (there are at least nine diverse definitions for context cited in [52], many of them based on examples). Usually, context subsumes – at least – these (quite broad) parameters: time, location, activity and identity. In its most general form, it is not compuationally tractable, which can be concluded from the discussion of context given in [59], where it is argued that context is primarily of phenomenological nature. However, for many applications, useful statements about the current context in which a process is executed can be derived. We will now give some examples where context is used to enhance the human experience of applications.

The research area of context derivation and context adaptation is quite broad. We will give a brief overview here. In [160], a context-aware PDA is described that changes the orientation of its user interface between landscape and portrait according to how the PDA is held by the user. A general framework for designing context-aware applications and processing contextual data is presented in [52]. Some application examples are given, including a “conference assistant” that supports the user by presenting information that is especially important at the user’s current location and guiding him to interesting presentations based on his own interests.

Another example for context derivation is given in [113]. Devices are en-

abled to tell if they are close to other devices, especially if they are carried by the same person. Simple accelerometers are used to accomplish this. Also with accelerometers, different user activities can be recognized as described in [92]. The authors attached sensors to major joints of the user's body, including knees, elbows and shoulders. The results collected from experiments indicate that activities like walking or writing on a whiteboard can be recognized with high accuracy with such rather simple sensoric equipment.

2.1.3 Other Applications

Several sensor networks have been prototypically deployed for scientific (e.g. [199]), military and other purposes. A survey of projects can be found, e.g., in [47] and [157]. Examples include

- environmental monitoring, e.g. in wineyards, forests, and glaciers;
- self-repairing minefields[1];
- improved care for the elderly through activity monitoring;
- equipment monitoring in industrial installations;
- sniper detection.

To our knowledge, security – in the sense of protecting the sensor network against intrusion or data manipulation – has not been an active focus of practical deployments, which is likely due to their prototypical nature and the emphasis on technical and algorithmic problems that have to be solved for putting such a network into operation. However, as the significance of sensor networks increases in critical military, social, and industrial applications, security issues will become more important. The example of the Internet shows that its potential in business applications today is often constrained by security-relevant threats. Thus, it is crucial to anticipate possible threats to sensor networks and be prepared to provide suitable countermeasures in order to support a smooth integration of sensor networks into applications.

2.1.4 Security Concerns

Although prototypical deployments of sensor networks have not been equipped with security measures until now, it is foreseeable that adequate security is a

[1] Such military applications raise important ethical questions that have hardly been addressed in the literature. They go beyond the scope of this work, but readers are encouraged to develop awareness of these issues.

prerequisite for the success of sensor networks in practice. The main reason is that sensor networks will play a critical role in monitoring and protecting valuable assets and people. Their open architecture makes sensor networks highly vulnerable. If they could be easily deactivated or manipulated by competitors or criminals, the consequence could be serious financial damage or even threats to human lives. The following scenarios are intended to illustrate these dangers.

Structural integrity of buildings Wireless sensor networks could replace currently used wired infrastructures for monitoring the structural integrity of buildings and bridges. Initial studies in this domain have been made [159, 136]. The collected data is used for maintenance planning, but is also important for the safety of the users of such a structure. Wireless sensor networks have a potential cost advantage over wired sensing equipment.

Sabotage is a realistic and significant threat to such a surveillance infrastructure. The easy accessibility of the communication medium makes WSNs vulnerable to message injection and manipulation attacks. Systematic reduction of data quality could lead to devastating effects in the long run on a large number of buildings and bridges.

Traffic assistance Cars and other vehicles become increasingly equipped with electronic assistance systems that are intended to increase the comfort and the safety of their passengers. Inter-vehicular communication and communication between vehicles and roadside infrastructure may even further improve on today's technologies. Wireless sensor networks play an important role for monitoring traffic and weather conditions and communicating their findings to vehicles.

It is conceivable that malicious users could exploit vulnerabilities of these communication systems for their own advantage, for example by signalling traffic lights or keeping parking lots free. There is even a danger of congestions or accidents caused by manipulating such an environment.

Healthcare Body-area sensor networks are a tool for monitoring vital signs of athletes and patients (e.g., see [177, 105, 75]). A multitude of body sensors can be combined, either as wearable nodes or implanted. The findings are communicated either to a personal device carried by the person herself, or they can be directly fed to a remote telemedicine center. Over time, valuable data can be collected for in-depth analyses. In case of a medical emergency, an alarm would be immediately raised, thereby leading to a sharply reduced reaction time.

Personal health data is of very sensitive nature, which most people would refuse to give away freely. It could be used to assess the lifestyle of a person or get hints on former and present diseases. Such data must therefore be protected against unauthorized access.

Military surveillance Military applications of wireless sensor networks include battlefield surveillance, where large numbers of nodes are deployed over territory through which enemy forces may move, treaty monitoring, and others [144]. The sensor network could autonomously differentiate between friendly and enemy vehicles, and report the number of vehicles detected as well as their speed and direction.

Security in the military domain is of paramount importance, since accurate information is key to effective actions and for avoiding own losses. If it were possible to manipulate the reports of a sensor network, forces may be misguided and could give the enemy an advantage.

Logistics Wireless sensor networks can be used in logistics, for example for monitoring the transport and storage conditions of goods. A prominent example is the cold chain for food [153, 43]. Being able to acquire more accurate and more timely data may not only increase consumer safety, but also reduce costs for product recalls.

The highly fragmented, highly competitive markets in the food industry seem to encourage sloppy handling of products, and sometimes even criminal actions take place. WSNs have the potential to deter such actions, if it can be ensured that the reported data is not falsified, which requires adequate security measures in all involved information systems, including sensor networks.

2.2 Sensor Node Characteristics

In this thesis, we aim at providing security mechanisms for sensor networks that are applicable as widely as possible. We therefore make as few assumptions as possible about the supported types of networks. The security mechanisms should be effective even in the most basic configuration.

Adopting the classification scheme of [157], our model can be characterized with the parameters shown in table 2.1. The unlisted categories (deployment, mobility, energy, modality, coverage, lifetime) have no direct impact on the security mechanisms that are proposed later, although they could be relevant with regard to other security mechanisms. The listed parameters are understood as

minimal requirements. The security mechanisms still work if they are exceeded by the actual characteristics of a network. However, in such a case other mechanisms could be more appropriate, i.e. more cost-effective or providing better security guarantees. For example, if a base station is available, it could act as a trusted third party and be used for node authentication and key establishment between nodes. This would allow end-to-end secure communication, which would be infeasible otherwise.

Computational Resources	matchbox to dust
Composition	homogeneous
Infrastructure	ad hoc
Topology	multi-hop
Connectivity	connected
Size	unlimited
Tamper resistance	non-neglibile

Table 2.1: General constraints on considered wireless sensor networks

We have extended the classification scheme by adding a category *tamper resistance*, which is highly relevant for the security of a sensor network. If nodes would be completely open to arbitrary manipulations, speaking of a secure sensor network would have little meaning. We therefore require a minimum level of tamper resistance that imposes at least a small, constant cost on an attacker before he is able to take control of a node. This could mean as little as that sensor nodes are hidden between rocks on an open field, or as much as them having a shielding that is hard to penetrate.

In the remainder of this section, we discuss the properties of this model in detail. For each category, we discuss the impact of possible configurations on the security of the network. In some cases, using a seemingly more powerful configuration introduces security risks that are avoided with a simple one. Of course, one doesn't always have the choice as the architecture of a sensor network is governed by its application context. However, one should be aware of the risks that a certain architecture implies. In many cases, it may be advantageous to prefer a simple design.

2.2.1 Computational Resources

Sensor networks are demanding with regard to protocol and algorithm design due to size and energy restrictions. Only a small amount of memory is available for program code and data, the energy source is limited, and the communication range is rather small. These factors indicate that algorithms for sensor networks should be as localized as possible in order to avoid long-distance interaction as

much as possible [65]. This principle of localization should also be applied to security mechanisms. In consequence, this means that end-to-end security may be too expensive in many cases.

For providing security services, we require that sensor nodes are able to execute basic cryptographic functions and store a moderate amount of key material. These requirements will be made more exact in the following chapters. Most current sensor node prototypes fulfill these requirements. Some very small designs with similar computational power as RFID controllers, like Smart Dust [189], may not be able to comply with these requirements.

Generally, increasing the avaialable resources on a sensor nodes helps the security of the node itself and the network as a whole. Some possible improvements are the following:

- By increasing the **memory size**, sensor nodes can store more key material. This saves bandwidth for key agreement or key transfer. More keys also result in a greater supported network size. Cost is increased if key memory is located within the tamper resistant module of the node, where space is expensive.

- A higher **processor speed** decreases latency for cryptographic operations, which in turn allows larger keys and makes asymmetric cryptography more attractive.

- A **cryptographic coprocessor** can be added. With hardware support, a speed-up of approximately 2 orders of magnitude can be achieved computing cryptographic functions (cf. Table 2.2 and Table 3.1). The coprocessor requires additional space within the tamper resistant module.

- By increasing the available amount of **energy**, more of it can be spent on security mechanisms, e.g. using larger key sizes leading to longer messages.

2.2.2 Composition

We assume a homogeneous network, i.e. all nodes possess equal capabilities. In particular, we do not assume that some nodes are better equipped than others (e.g. higher communication range), or that base stations exist. This allows large networks to be easily deployed, for example by dropping nodes out of a plane. It requires that the nodes organize themselves autonomously. As all nodes have equal capabilities, failed nodes can be substituted for by others.

One could add "super-nodes" for better security. These special nodes could have a high level of tamper resistance, additional computational or energy resources etc. without increasing the overall cost too greatly. They could then act as trusted parties for key exchange, for example. Thus, as long as super-nodes are not compromised, end-to-end security guarantees between nodes can be implemented.

However, such super-nodes come with a disadvantage. They pull traffic towards them, which not only depletes their neighbouring nodes more than others. It also makes the super-nodes themselves as well as their neighbours more valuable for an attacker. For illustration, consider the following example. Assume that an attacker manages to compromise one super-node (which requires a significant effort). Then, he disables the ordinary sensor nodes surround the other super-nodes (which is cheap). This forces all nodes to use the only reachable super-node, the compromised one, for security relevant activities. Thus, despite the high effort for compromising one super-node, the attacker gains control over the entire network at relatively low cost.

2.2.3 Infrastructure

There are two fundamentally different types of communication networks. The first one relies on a supporting physical infrastructure that provides the necessary services for clients to communicate, such as name resolution, routing, or persistence. This class comprises networks such as the traditional telephone system, mobile phone networks, the Internet, the (paper-based) mail system, and the television system (which is one-way broadcast only). Here, the operation of the network is taken care of by dedicated entities that are clearly separated from the network's clients who use terminal devices to use the network's services. A sensor network based on this paradigm employs *base stations* which offer an infrastructure for the sensor nodes to use. A base station is a device equipped with more resources and a greater radio range than an ordinary sensor node. Each node usually has a direct link with a base station and exclusively communicates with that base station. Direct communication between peer nodes does not take place. A possible exception is message relaying on behalf of nodes which happen to be out of range of the base station. Careful planning is required in order to minimize the number of base stations while achieving full coverage.

The second network type does work completely without a supporting infrastructure. There are no dedicated devices or installations that support the network's clients. Instead, all services are provided on a peer-to-peer basis by

the clients themselves. Cooperation between the clients is required in order to assure fair use of each other's resources. Examples of such networks are overlay peer-to-peer networks, wireless ad hoc networks, and amateur radio networks (ham radio). Sensor networks based on this paradigm are simpler to set up than those of the previous type. Often, it is possible to simply deploy the nodes randomly within an area. However, the algorithms and protocols used in such networks are usually more complicated since the nodes have to collaborate in providing network services and there is no global view of the network's state.

In practice, most networks will probably be a mixture of these two extreme incarnations. Both designs may exist in parallel in the same network: inaccessible areas may be covered by randomly deployed, self-organizing nodes, while populated areas may be covered with the use of base stations. Or, a network may be basically self-organizing with few base stations spread randomly throughout the network area, providing access points for external clients, for example, but without full coverage. A network may also exist temporarily in environments where no base station is available, for example a group of mobile sensors during transportation.

Base stations can provide certain security services, such as authenticated broadcast (cf. the μTESLA protocol [142]), for example for distributing code updates, or acting as trusted third parties for establishing secure links between nodes as each node maintains a trust relationship to a base station. Generally, they are not as restricted as sensor nodes and thus the extensive use of public key cryptography is possible.

For the purpose of our considerations regarding security, we do not rely on base stations or any supporting network infrastructure for communication within the sensor network. The deployment of an infrastructure is costly and not always possible, therefore we want to avoid relying on it for security purposes. In fact, base stations do not only offer new opportunities for security services, but also introduce risks similar to those in heterogeneous networks (see the previous subsection).

The lack of an infrastructure means that all security-relevant decisions have to be made autonomously by the network, for example whether a query is authorized to access certain information. Such access control decisions should be made by collaborations of sensor nodes [17].

2.2.4 Topology

We do not restrict ourselves to a specific network topology of the connectivity graph. We generally assume a complex, multi-hop topology. Generally, we assume a multi-hop topology which is necessary due to the large number of nodes, their limited radio range, and the lack of a base station covering the whole network. This means that messages usually have to travel over multiple intermediary nodes before they reach their target. This is in contrast to a star-like topology, for instance. In such a topology, all messages would be exchanged between the nodes and a central component, and every node would have a direct connection to this component. We do not assume a certain communication pattern. Instead, we allow arbitrary point-to-point message exchanges, i.e. a node is allowed to send messages to any other node in the network.

One of the problems in routing a message over multiple hops is addressing. Usually, each host in a network is assigned a unique identifier which serves as its network address, such as an IP-address on the Internet. Such addresses may be hierarchichally organized in order to allow routers to save space in their routing tables. Using network addresses and address-based routing algorithms is problematic in sensor networks:

- Sensor nodes may fail prematurely due to several reasons, which has to be considered a common case in a sensor network. As the network reorganizes after such failures, services are migrated to new nodes. These services should therefore not be addressed through the identifiers of the nodes they are running on.

- In a sensor network, as few messages as possible should be exchanged in order to save energy. This especially applies to message exchanges between distant nodes, as the energy resources of all intermediary nodes are affected as well. Therefore, route detection and maintenance are costly operations that should be avoided.

Due to these reasons, we consider geographic routing [91] and area-constrained broadcasts as being more suitable for providing routing stability and efficiency. These mechanisms do not require the nodes to keep any state information in order to route messages. Routes are established on demand, thus compensating for node and link failures, and even node mobility.

2.2.5 Connectivity

We assume the connectedness of the network, more exactly we assume that most nodes are connected in a large network component. Single, disconnected nodes should not affect the overall operation of the network. Nodes that are sporadically connected to their neighbours, for example due to fluctuations in link quality or mobility, are supported. If the network is partitioned into two or more large components, the security mechanisms should still be effective within these components. After the network becomes connected again, no change in the security infrastructure should be required.

2.2.6 Network Size

We want to support networks of virtually arbitrary size. This is necessary since sensor networks are most useful when they cover an area as large as possible. The small size of sensor nodes and their limited communication range require that a very large number of them is deployed in order to achieve large coverage. Assuming that sensor nodes will be manufactured at low cost with the size of a grain or even smaller, it will become feasible to deploy them in extremely large numbers, i.e. on the order of hundreds of thousands and even more.

We also would like to support the extension of an existing sensor network by any number of nodes. Security mechanisms should not make it necessary to fix a maximumum supported network size when a network is initially deployed. This would limit a network operator's flexibility to adapt the network to the changing needs of its application context. Rather, the security mechanisms should support a dynamic network structure, especially the introduction of freshly deployed nodes into the existing network, the replacement of exhausted nodes, and the withdrawal of nodes.

A larger network means that an adversary has a greater selection of nodes he could attack. This should not lead to a greater vulnerability of the overall network. Ideally, nodes that are located in one area should not be affected when nodes in a distant area are being compromised. Such a clear isolation of intrusions may not always be possible. For example, it may be necessary that messages originating from an uncompromised node are being routed through a compromised one, which could then selectively suppress or alter these messages. The security mechanisms we propose are able to mitigate the impact of compromised nodes in such cases.

2.2.7 Tamper Resistance

Sensor networks are often deployed in publicly accessible or even hostile environments. An adversary can thus approach the nodes, establish a wireless communicatin link, and even physically access them. Without any means to prevent an attacker from accessing the node's hardware, it is easy for him, for example, to inject his own code or to extract key material.

A minimal level of tamper resistance is able to deter a large class of potential adversaries. Encountering resistance, an occasional attacker with a low level of sophistication and few resources is likely to look for easier targets. Even merely hiding the sensor nodes may be sufficient in certain application scenarios. In order to deter a determined and resourceful attacker, a higher level of tamper resistance is necessary, for example by incorporating techniques that are common for building security modules [63], such as secure packaging, hardware locks that control access to memory areas, and tamper response mechanisms (e.g. zeroization of sensitive memory areas upon detection of unauthorized movement). Ideally, there should be a mechanism for sensor nodes to recover from corruption.

It is not necessary to make the whole of a sensor node tamper resistant. It is sufficient to have a small module with these properties on a node. This module contains the sensitive parts, such as the key storage and the processor itself. Sensors, for example, need not be part of this module as it is likely just as easy to manipulate their readings through outside means as it would be by directly accessing the sensor hardware.

The level of tamper resistance that is required for an application scenario can be determined through risk analysis. Important questions are, who is going to attack the sensor network, how large is the affected area, what is the potential damage of an attack, how likely is an attack. If the cost for implementing the required tamper resistance mechanisms is higher than the expected damage, deployment would be uneconomical. As one of the advantages of sensor networks is their low cost compared to alternative techniques, it is likely that in most cases, tamper resistance capabilities will be at a moderate level. However, some applications justify a high investment in tamper restance mechanisms, such as surveillance of military facilities to enforce an international treaty, for example.

We assume throughout this work that an attacker needs to invest at least some fixed effort to compromise a single node. This rules out software-only attacks which require an initial effort to find a way to exploit a vulnerability, but can be applied to arbitrarily many nodes with minimal, i.e. approaching

zero, cost. This gives rise to an adversary model that determines the strength of the adversary based on the number of compromised nodes.

2.2.8 Further Assumptions

Mobility

We make no assumptions about the mobility of sensor nodes. Geographic routing is largely invariant to mobility as long as the mobility rate does not affect local message forwarding (which is the case as long as transmitting a message is much faster than the movement of nodes, which we can safely assume for radio communication). The schemes of chapter 4 provide the means for key agreement between neighbouring nodes. A high mobility rate could impose a significant overhead as the neighbourhood of a node frequently changes. However, this does not affect the security of key agreement or the proposed security schemes. A high mobility rate can be compensated for by making all negotiations between nodes reactive, i.e. they occur only when a higher-layer protocol requires it. This avoids unnecessary message exchanges but is likely to increase response times.

Evolving Hard- and Software

Currently, sensor nodes are being built mostly as prototypes in small quantities that are used in education and prototypical deployments in research projects. There are several lines along which the technology is being improved, such as component integration for optimizing size, or provision of an enduring power source. The software layer is being developed mostly using platforms such as the BTnode [20]. The physical size of such platforms is largely dominated by the batteries being used. For deployments in real-world environments, usually a special casing is required to protect the sensor nodes against external forces.

A size shrink can be expected in the future due to two lines of progress: batteries with increased lifetime at lower size, and decreased energy consumption by electronic components, protocols, and software that makes more efficient use of the available power source.

2.3 Related Network Types

Miniaturization in electronics components has made the commoditization of digital wireless communication possible. This allowed not only the proliferation of mobile phone networks, which rely on a fixed infrastructure, but also

the development of wireless communication for hand-held and portable computers. These interfaces (e.g. IEEE 802.11, Bluetooth, ZigBee) also support ad hoc networking, i.e. devices are able to connect to each other without a supporting infrastructure. Although there are sensor network designs that rely on base station support, in this work we concentrate on infrastructure-less architectures. Here, we review the most important related network types. Peer-to-peer overlay networks are included in the presentation since they follow the same principle as wireless ad hoc networks and provide their own supporting services, although they are implemented on top of the Internet infrastructure.

2.3.1 Ad Hoc Networks

An important class of networks that share many similarities with sensor networks are subsumed under the notion of *ad hoc* networks. As sensor networks, ad hoc networks usually operate without infrastructural support. This means that routing, security, and other services must be provided collaboratively. However, there are significant differences between the network types since their application cases are quite different. In contrast to sensor networks, ad hoc networks are usually comprised of heterogeneous devices which are specialized to different tasks and are under the control of different users. Also, from a user's point of view, they operate on a different level. Ad hoc networks often fulfill user-controlled tasks, while sensor networks operate more autonomously. However, sensor networks incorporate features of ad hoc networks and vice versa, and the boundaries between both network types are blurry. The main differences with regard to security are summarized as follows:

- **Fairness** Ad hoc networks are not operated nor owned by a single entity. Usually, every participating device is associated with a different user on whose behalf it operates. As a consequence, fairness is not easily maintained when part of the users act selfishly, trying to exploit other users' resources without offering their own. Also, it is not easy to decide whether it is advantageous for the network to admit a new device. These problems are part of normal operation in ad hoc networks. In sensor networks, similar problems occur only when a network is under attack.
- **Group communication** Often, some devices organized in an ad hoc network need to form a closed group. This usually means they have to negotiate a common group key in order to exclude occasional (or malicious) passers-by from their communication. This is not common in sensor networks, as a sensor network as a whole can be considered a closed group.

- **Physical characteristics** Nodes in ad hoc networks are usually more resourceful than sensor nodes. While sensor nodes are designed to be as small as possible, devices in ad hoc networking are usually handled by a human user. Therefore they cannot be arbitrarily small (which allows their microprocessors to be more powerful), and they must be equipped with a minimal user interface. Besides, their human users take care of recharging their batteries. In contrast, sensor nodes are often not accessible after deployment. They have to operate autonomously and manage with their limited energy supply.

- **Failure modes** A device in an ad hoc network that stops working is likely to be replaced (or recharged) by its user, while a failed sensor node will hardly be noticed. A user will keep her devices under guard most of the time, and protect them against external threats to her best abilities. Sensor nodes are susceptible to external influences and may often be disabled by environmental conditions.

  User devices such as PDAs or mobile phones are often vulnerable to network-level attacks such as malicious code [102]. Although sensor nodes are not immune per se against such threats, code injection attacks require special equipment and are only possible in physical proximity to the device. Code injection attacks are further impeded by the Harvard computer architecture that is often used in embedded microcontrollers. This architecture physically separates memory sections for code and data. It is thus possible to prevent unintended code updates.

- **User vs location** As devices in ad hoc networks are associated with human users, an attack on a device usually means that its owner, or rather personal information she possesses, is the actual target of the attack. In a sensor network, nodes are associated with locations rather than users. An attack on a sensor node may provide the adversary with some control over information associated with the node's location. This may allow the adversary to observe an object (such as a human), but will not directly reveal sensitive information about it.

- **Network scale** In a sensor network, the number of nodes is usually very high, and so is the density, leading to some inherent redundancy. Taking control over a single node will not even allow the adversary to completely control the data flow in some area. Also, only the small fraction of the total network traffic passing through that one node will be revealed to the adversary. Ad hoc networks are usually smaller, and the significance of

a single device is much higher. For example, if the adversary has taken control over one device, it may allow him to compromise all the sensitive communication that takes place within a closed group.

We conclude that the motivation for an attack on a device in both network types is rather different: In an ad hoc network, the target of an attack is the sensitive information accessible through a personal device. Additionally, a single attacked device may reveal information about the whole network of which the device is a member. In a sensor network, control over one node provides access to information associated with a location, but only limited information about the overall network. Thus, a substantial attack on a sensor network will usually not be restricted to a single node.

2.3.2 Personal Area Networks

A personal area network (PAN) is comprised of devices that are associated with the same (human) user. They are specialized for different tasks and often work collaboratively to provide some service to their user. A simple example would be a mobile phone and a headset that are linked through a wireless Bluetooth connection. This simple network may become more complex if a PDA is added that is equipped with both Bluetooth and WLAN. The headset could then be used not only for GSM calls with the phone, but also for Voice over IP calls via the PDA. Also, the user may update her addressbook in either the phone or the PDA, as the devices are able to synchronize their databases.

Other device types are available or may become open to personal area networking in the future, such as medical sensors for monitoring health parameters, the numerous sensors and actuators in a car or home, and all sorts of wearable devices that are equipped with computing and communication capabilities, such as glasses and clothes [118].

From a security point of view, such PANs should be largely closed to the outside world. There may exist well-defined interfaces to other networks, such as the mobile phone or the Internet connectivity of a PDA. But, for example, a heart rate sensor should not be globally accessible through its own IP address. This kind of isolation is manifest in the Resurrecting Duckling security policy [171], where devices are coupled and configured through a powerful master device (the “mother duck”). Access to a device is possible only if allowed by the master device (and possibly after some user interaction).

Sensor networks and PANs have some commonalities, and a (small-scale) sensor network may even be part of a PAN, for example as sensors built into clothing. A sensor network may be considered a closed group similar to a

PAN, both being controlled through a master device (base station). However, they also show some differences in detail. For example, network organization is static in a PAN, governed by the capabilities of the devices, which are heterogeneous in nature, while the homogeneity and failure rate in a sensor network demands a flexible and dynamic organization. Generally, PANs are similar to ad hoc networks (with a single user), thus the issues discussed in the previous subsection apply to them as well.

2.3.3 Peer-to-Peer Networks

Peer-to-peer overlay networks have become a convenient infrastructure for constructing application-level networks based on the Internet infrastructure. They are being used for file storage and retrieval, media distribution, and information exchange [134]. Generally, all participants are regarded as being equal. The major threats in these networks originate at the social level, similarly to user-level ad hoc networks. An additional difficulty comes from the fact that the participants in a peer-to-peer network are largely anonymous. This allows misbehaviour on all technical and social levels without the danger of retribution. To overcome the implications of this, reputation systems have been introduced [55, 121].

Although peer-to-peer networks are built on top of the Internet, which allows arbitrary connections between nodes, they introduce a separate address space for the management of meta-information. By binding information to these addresses, for example using distributed hash tables, a node storing a specific piece of information can be found more easily by navigating within this address space (see [173]). Large amounts of data are, however, transferred through Internet-level point-to-point connections.

Peer-to-peer networks are often considered as social networks, as the resources for their operation are provided by users with similar interests. The self-organization of these networks is considered their great strength, as censorship is virtually impossible and large numbers of faulty nodes can be tolerated. However, as there are no means for quality assurance and user authentication (unless some repudiation system is introduced, which puts newcomers at a disadvantage), peer-to-peer networks may be overloaded with meaningless information and exploited by “leechers”, i.e. users who draw large profit from the network but do not contribute own resources.

Sensor networks are not prone to these socially motivated threats, as there is a single operator controlling all network hosts. Access to the network’s resources is subject to access control. New nodes are only deployed by the same

operator (or with her permission). While peer-to-peer nodes draw most of their utility from being open to anybody, sensor networks are under centralized control.

The application areas for both network types are quite different. Peer-to-peer networks manage arbitrary data, while sensor networks are tightly bound to their deployment area. The architecture of peer-to-peer networks allows them to introduce distributed redundancy for storing data. For example, CAN [150] assigns multiple nodes to a piece of information. In contrast, sensor nodes are closely bound to their geographical location. Obtaining current information about a specific area is only possible through nodes that are located in that area. An adversary who wants to disable access to a specific piece of information in a peer-to-peer network would have to either disable multiple, randomly distributed Internet hosts (which can be considered hard) or pose as multiple identities, thereby drawing all replicas of the data item to himself (which may be easily possible). In a sensor network, he would have to take control of a number of sensor nodes within the respective area.

Despite their differences, both network types share a number of properties, such as self-organization, homogeneity, and distribution. As noted in [157], there seems to be some potential for many techniques being applicable in both architectures. As an example, we refer to the discussion of interleaved authentication in chapter 6, which will be considered in both contexts.

2.4 Related Device Types

Small, embedded computers are commodity products today. According to [76], 95% of all microprocessors being sold are embedded microprocessors. The difference in performance between them is huge as they range from 4-bit to 32-bit processors. There is a wide range of specialized designs already available, and often microcontrollers are designed and manufactured for specific purposes. Today's prototypical sensor node designs are based on general-purpose controllers and usually placed somewhere in the middle of the performance range. In this section, we give a brief overview of commonly used embedded microprocessors and compare them with those used in sensor networks.

2.4.1 Smartcards

Smartcards are security tokens commonly used for authentication, e.g. as Secure Identity Modules (SIM) in mobile phone networks [184], and for creating digital signatures [162]. They have no own power source but can only be used

in conjunction with a reader device, which often also provides a user interface (keypad). The most important feature of smartcards is their tamper-resilience. This allows to store secret information, such as a cryptographic key, within the smartcard's memory with a minimal risk of disclosure even if an adversary obtains the smartcard. A smartcard has a well-defined interface through which its functionality, and therefore the stored secret, is accessible. This interface requires proper authorization by the user, such as entering the correct password (PIN).

Tamper resistance against all classes of attackers is impossible to achieve. If one is willing to invest enough resources, extracting the secret from a smartcard is possible as has been shown through successful attacks [6]. The goal therefore is risk minimization when deploying smartcards. For current applications, the risks seem acceptable, as the widespread use of smartcards shows. Other applications, however, seem to be hampered by security issues. The problem here is not the smartcard itself but its connection to a backend system, which boils down to the question, how can the user be sure to be connected to the right system, and who else has access to this connection? This problem is usually addressed by trying to establish a *trusted path* between the user and the backend system. This is a hard problem as was recently shown practically by exploiting the fact that smartcards often accept the PIN only in cleartext, which makes it possible to intercept the PIN through a specially crafted smartcard reader [4]. This violates the trusted path between the user and the smartcard.

Sensor nodes are hard to protect against tampering, due to their deployment in openly accessible locations and tight cost constraints. Thus, the security of a sensor network should not depend on the integrity of single nodes. Even if a network is under (not too heavy) attack, the risk of using it should be acceptable. Of course, if too many nodes are compromised and the attacker gains control over large parts of the network, relying on the results delivered by the network would become dangerous.

In contrast to sensor nodes, smartcards are dedicated devices with specialized functionality. Their programmability is limited. A smartcard "application" is usually defined by a set of files accessible from an application running on a host to which the smartcard reader is connected to. The SIM Application Toolkit is an exception in that it also allows the execution of code on the smartcard itself [1].

It is obvious that the usage patterns of sensor nodes and smartcards are fundamentally different. While a smartcard is associated with a single user, performing security-relevant actions on behalf of this user, sensor nodes operate autonomously in large clusters, obtaining sensoric input from their environ-

ment and performing arbitrary, distributed computations. Although their computational power is on a similar level, both device types are used for different tasks.

A combination of both concepts could have some potential. A tamper-resistant component could be included in a sensor node design. This component would be responsible, e.g., for storing secret keys and performing cryptographic operations. It could also act as a "supervisor" for the programmable part of the node, thus helping to detect intrusions and to recover from them [174].

2.4.2 RFID

Radio Frequency Identification (RFID) is a technology for object tracking. An RFID tag is a small label made of a logic circuit and an antenna. The complexity of such logic circuits ranges from very simple, such as a 1-bit storage, to quite complex, for example implementing the functionality of a smartcard. Reader devices communicate with these tags wirelessly. These readers also transfer energy to the tags through an electrical field. This allows the tag to perform complex calculations, such as executing a challenge/response protocol for authentication, or verifying a password. The tags are completely passive and can only operate when triggered by a reader device.

By placing reader devices at important checkpoints, tagged objects can be tracked during transportation, or their presence can be verified, for example in a warehouse. As RFID tags become cheaper, they become more available to mass deployment. It is anticipated that they will eventually replace the bar codes that are found on most items today.

The most important difference to sensor networks is the fact that RFID tags can not communicate with each other but only through a dedicated infrastructure. Thus, RFID tags do not operate autonomously and in a self-organizing manner. However, from an application's point of view, sensor nodes and RFID tags may be treated in a similar way. Both device types may be temporarily disconnected, they both carry information about their environment (in the case of RFID tags, this would be comprised of the object the tag is attached to), and they both have limited computational capabilities.

RFID devices can be functionally similar to smartcards and are therefore suitable for many security purposes, for instance as access keys. They offer convenient use through their contactless interface. There also seems to be some potential for them in anti-counterfeiting systems [126, 169].

RFID technology has been heavily criticized for endangering people's privacy as an increasing number of consumer goods and other items such as pass-

ports are being tagged. Since RFID tags can be accessed through a concealed, wireless connection, people can be potentially tracked using the tags that are embedded in their clothing and other personal items. By associating the identifiers of RFID tags with the identity of a person, this person can be recognized whenever one of the tags is detected. This leads to even better tracking capabilities than with biometric recognition techniques, as not even physical contact or a good view on a person is necessary for recognition.

Consequently, there are different views on RFID security. The first is from the system operator's point of view, where RFID tags are used as authentication tokens. Here, security concerns exist regarding the clonability of tags, or other manipulations that would yield illegitimate authorization. Also, the channel between the reader device and the tag is a potential target for attacks. These security issues are equivalent to those concerning the use of smart cards. As the circuitry of an RFID tag is often much simpler than that of a smart card, novel techniques are researched that offer certain security guarantees.

The second view on RFID security regards the tag itself as a risk as it allows identification, and thus poses a potential threat to privacy. It is therefore crucial to restrict access to the tag in such a way that the user has to agree on its use before it can be accessed. For passports, this may be achieved by using a metallic cover that shields the tag as long as the passport is closed. Another approach is the use of a key that is required to access the data stored on the RFID tag. This key can be determined from the optically readable data that is printed in the passport (personal data of the passport holder, passport number). Thus, a reader device with optical access to the passport can also read the data from the RFID tag [67]. Whenever the passport is handed to somebody, for example a customs officer, the right to access the RFID tag is implicitly granted.

2.4.3 Embedded Computers

As mentioned above, embedded microprocessors make up for the majority of all microprocessor sales. It is estimated that on average a household in the U.S. owns approximately 60 microcontrollers [76]. Most of these computers are not networked in any way. They simply function as controllers for their containing device, for instance a car engine or a washing machine. Making such devices connected to their environment to other devices has just been started. One such example is the registration of trucks on highways in Germany for road billing, where so-called "on-board-units" that are installed in trucks communicate with by-road stations for automated billing and can also be queried by mobile checking teams [29].

It is expected that with the emergence of ubiquitous computing, embedded systems will become more widespread as the costs for hardware are decreasing and integration becomes easier with novel materials. In order to make full use of such a large number of embedded computers, they will be able to connect to each other and their environment. It can be expected that through the increased utility they provide, these systems will then also be used in security-relevant applications. Thereby, these devices will carry sensitive information and thus must be protected against abuse. At least, user interactions with them are likely to be recorded and can be tracked, which may violate a user's privacy.

The main difference between embedded computers and sensor networks is that the integration of a sensor node to its environment is usually not very tight, while an embedded computer implements a significant part of the functionality of the object in which it is embedded. An embedded system does not have to operate autonomously but may make use of the resources provided by the containing object. For example, the above mentioned on-board-units act on behalf of trucks in the context of road pricing, and are also powered by these trucks. In contrast, a sensor node should function independently from its environment, since its main task is to monitor the environment without being intrusive. However, the borders between these two concepts are becoming blurry in applications such as "smart tagging" (cf. [167]) where sensor nodes are being attached to physical items. The task of a node here is to monitor the context of a specific item, thus both entities are becoming closely coupled. As the underlying technologies are similar in both cases, sensor networks eventually have to be considered as a special case of networked embedded systems.

Maintaining the security of an embedded system is a challenging task since such systems incorporate features from two worlds, namely those of powerful computers being responsible for critical functionalities, and those of autonomously operating devices with limited application scope. The first characteristic requires the application of security mechanisms that are also being applied in desktop or server-oriented computing, which are resource-intensive in terms of computing power as well as development and maintenance effort. On the other hand, the second characteristic requires freedom of administrative overhead and cost minimization. Achieving these opposing goals is obviously not trivial. Additional security requirements of embedded systems are discussed in [97], which include their energy constraints and the context of developing embedded systems. Digital Rights Management is also an issue for embedded systems handling protected content [135].

The application space of sensor networks is very constrained, so many security issues for general embedded systems are mostly irrelevant. However, in

both cases devices are potentially exposed to attackers. Tamper resistance (or tamper evidence, if sufficient) is therefore a major requirement for their security [95, 151]. The research focus in sensor networks, including security mechanisms, is mainly governed by the assumption that resources are very tightly contrained and replenishing the energy source is often not possible. In contrast, embedded computers are usually adapted to and integrated with their application environment, which is often of a technical nature where a power source is available and where regular maintenance is performed. Therefore, existing security mechanisms can often be adapted to the special requirements of embedded systems, while novel mechanisms are needed for sensor networks.

2.5 Sensor Network Models

2.5.1 The Geometric Model of Sensor Networks

It is common in the literature to represent the topology of a sensor network as a graph in the plane, where vertices represent sensor nodes, each at its distinct location, and an edge exists between two vertices if the respective nodes can communicate over a (wireless) link. For simplicity, it is often assumed that these links are symmetric, and that the communication range is equal for all nodes. Usually, the deployment model for sensor networks is assumed to be *random*, i.e. nodes are randomly distributed on a plane within a constrained area, e.g. a square or a circle. Most often, a *uniform* random distribution is assumed. This distribution makes the least assumptions about the real-world deployment and thus one can hope that this model provides useful information about a large class of real-world networks. Although these assumptions are stretching the practical properties of sensor networks, they provide a useful abstraction for performing calculations and simulations on such networks.

For this *random geometric graph* model, there exists a large body of theoretical work, which is comprehensively presented by Penrose [140]. One of the most important aspects is the connectivity of such graphs. In a wireless network, it is desirable that all nodes are contained in one large connected component of the network graph. This means that there exists a path between every pair of these nodes. Isolated nodes, or small connected components, are undesirable since these nodes are not able to collaborate with other nodes, and it may not be possible for them to communicate with a base station. Bettstetter [18] evaluates the conditions under which a wireless network is connected with high probability. In particular, the required transmission range is derived for a given density (number of nodes per area unit), and, vice versa, if a transmission range

is given, the required density can be derived.

In this work, we assume a pragmatic view of wireless networks. Avoiding isolated nodes is not a prerequisite. In sensor networks, we can assume a very large number of nodes but also a significant failure rate. This means that with high probability, sooner or later there will be node failures resulting in the isolation of other nodes. To avoid this, or at least prolong the connectedness, more nodes have to be deployed on the same area. However, the protocols we are presenting still work within a connected component even if that component comprises only a small fraction of all network nodes.

2.5.2 Small-World Networks

General random graphs [27], which are not restricted to a two-dimensional plane, are often used for modelling relationships between real-world entities, for example in "small-world" social networks [191]. The latter networks are characterized by a strong connectivity within a neighbourhood of nodes and a sparse overall connectivity, but nevertheless short paths between any pair of nodes. The strong connectivity within a neighbourhood is represented by a large clustering coefficient, which is defined as the probability that two nodes having a common neighbour are connected themselves. Few long-range connections between nodes not having a common neighbour are sufficient to guarantee short path lengths between any pair of nodes. In practice, these short paths are often hard to exploit efficiently, as the long-range connections may not be known.

Small-world networks cannot be directly applied for describing the communication relationships in common wireless networks. Neighbourhood in such networks is prescribed by the spatial proximity of nodes. Long-range connections cannot be established due to the limited range of wireless communication. However, there are two approaches of introducing small-world characteristics into wireless networks. The first one is to establish long-range communication links by introducing additional communication paths, for example by introducing a few pairs of nodes with a wired connection between them [41] or by attaching wires to the sink [166]. This approach is often prevented in practice by the deployment model, such as aerial deployment. It also introduces heterogeneity as wired nodes have to deal with more load than other nodes and therefore require additional energy sources. From a security perspective, such nodes offer a more valuable target for attacks than ordinary nodes.

The second approach to introduce small-world characteristics does not affect the physical level but is only carried out in a virtual manner. Node pairs

are associated logically, for example on the application level. This association involves either simply introducing the nodes' identities to each other or establishing a pairwise key, for example. Shortcuts of this kind can be helpful in reducing the communication path lengths, and therefore cost, as shown in [79], where service discovery is proposed as an application area. Gathering information through shortcuts is more efficient as fewer nodes are involved in the process. This "wiring" on the logical level has the advantage that no physical changes to the network are necessary, and therefore much more shortcuts can be introduced at minimal costs.

2.5.3 Fundamental Properties of Sensor Network Graphs

We give a brief overview over some fundamental concepts in wireless sensor networks. These concepts will be used when security properties of networks are analyzed, and they serve as a guidance for generating simulated network models. They are useful, for example, to guarantee that a simulated (i.e., randomly generated) network is connected with high probability.

We assume that the sensor nodes are uniformly distributed on a plane surface. The communication radius is constant. It is assumed to be independent of the location or the neighbours of a node, nor are environmental interferences taken into consideration. Also, nodes do not adapt their transmission power.

We generally disregard so-called "border effects" as they do not significantly affect our results. Every realistic deployment area is constrained. Nodes that are located close to the border of the deployment area have, for example, fewer neighbours than nodes that are further away from the border. Thus, there will be a certain error if such a formula is applied within a realistic context. An analysis of these effects can be found in [19]. In simulations, we are generally able to neutralize border effects by considering nodes that are located far enough from the border only. For a square deployment area with side length w, for example, we might consider only nodes that are located within the square defined by (R,R) (top-left corner) and $(w-R,w-R)$ (bottom-right corner) where R is the communication radius of the nodes.

Nodes may be regarded to have different "value" depending on whether they are inner nodes or outer nodes. From an application's perspective, nodes close to the border are generally less valuable to an application than inner nodes. This assessment is based on the observation that outer nodes are less densely connected and are therefore involved in fewer conversations than inner nodes. From a security point of view, outer nodes may be very important since they may be the first ones to observe an attack. If the WSN deployment area is

inaccessible, the outer nodes might also serve as communication entry points into the network. Thus, it might be worthwhile to provide them with extra protection.

Density

Let A be the size of the deployment area, and N be the number of nodes, i.e. the size of the network. The density is defined as number of nodes per unit area:

$$\rho = \frac{N}{A}$$

Neighbourhood

Let R be the uniform communication radius of nodes. The expected number of directly reachable (or 1-hop) neighbours is

$$d = d_1 = \frac{\pi R^2 N}{A} .$$

This is, exactly speaking, only true for nodes that are located far enough from the border of the deployment area as mentioned above. The same is true for the expected number of neighbours reachable within k hops, which is:

$$d_k = \frac{\pi (kR)^2 N}{A} .$$

Path Lengh

On a unit square, the longest path can be expected to be approximately

$$\frac{\sqrt{2}}{R}$$

if the nodes are sufficiently densely distributed as that many hops are required to span the distance from one corner to the opposite corner. In a less dense graph, we can expect that the longest path will not be as straight and therefore be slightly longer.

A disk of size A has a diameter of $2\sqrt{A/(2\pi)}$. Therefore, the longest path can be expected to contain

$$L_{\max} = \frac{2\sqrt{A/(2\pi)}}{R}$$

hops.

In real-world deployments, the covered area is unlikely to be a regular shape. Deployment planning must take into consideration several parameters, such as the connectivity of the network and coverage. There is no general upper limit on the path length in practice, but for a given application, the maximum path length is usually known or can be approximated.

Connectivity

According to [18], for a network with $n \gg 1$ nodes and a homogeneous node density ρ (nodes per unit area), all nodes are connected to the network with probability p, if the radio range r_0 of all nodes is

$$r_0 \geq \sqrt{\frac{-\ln(1-p^{1/n})}{\rho\pi}} \,. \tag{2.1}$$

If a minimum node degree d_min of at least n_0 is desired, the following equation is applicable:

$$P(d_{min} \geq n_0) = \left(1 - \sum_{N=0}^{n_0-1} \frac{(\rho\pi r_0^2)^N}{N!} e^{-\rho\pi r_0^2}\right)^n \tag{2.2}$$

It gives the probability with which random variable d_{min} is above the threshold n_0.

Example Topologies

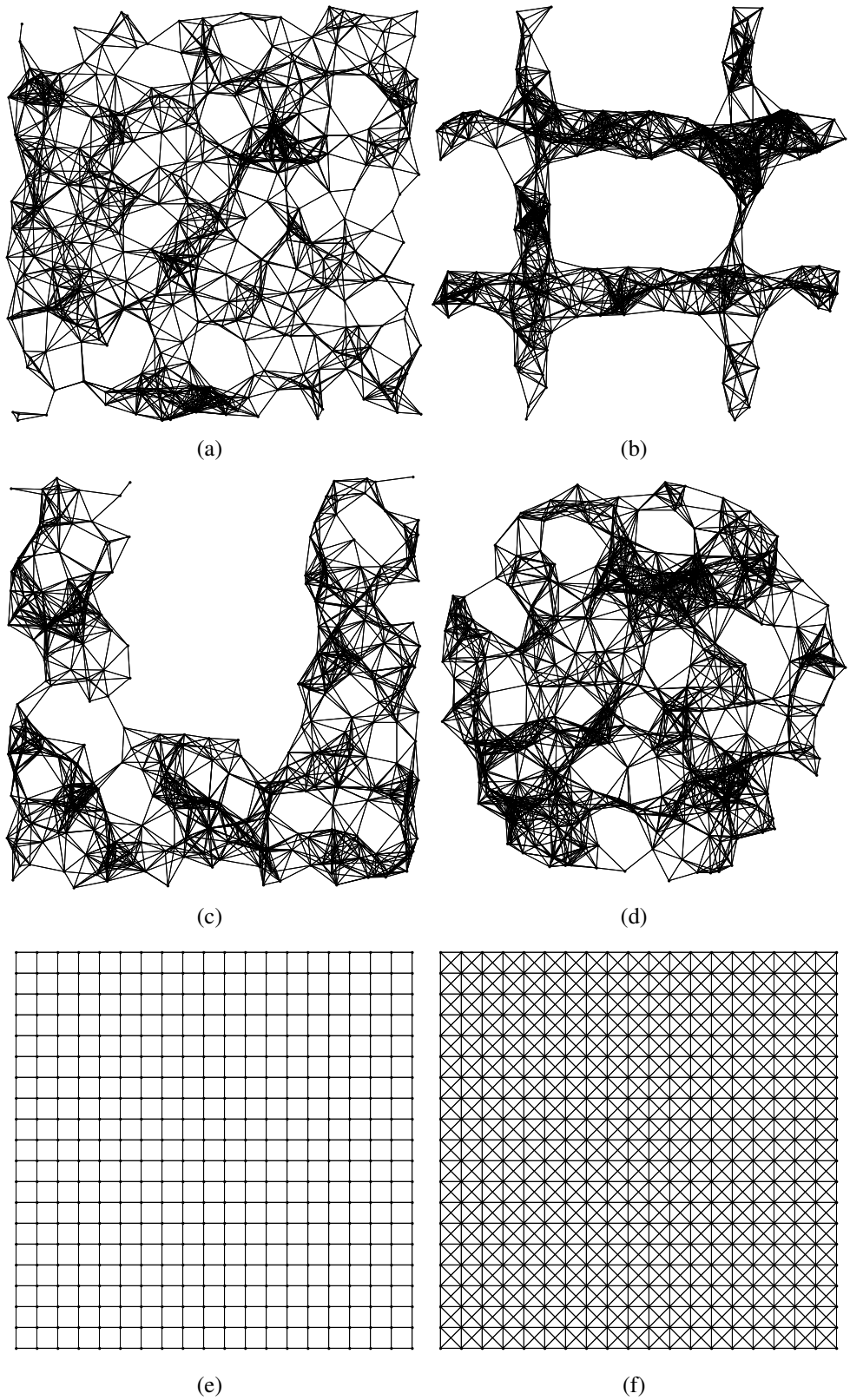
Figure 2.1 shows examples of sensor network topologies. These are simple prototypes of real-world deployments. Most of them are regular shapes that are unlikely to occur in practice, but are useful for quantitative evaluations. Figure 2.1(b) is intended to resemble the streets around a house block, which is a possible application area for sensor networks. The U-shape of figure 2.1(c) might serve as a model for a floor in an office building.

It has to be noted that the fundamental qualitative results in this work are not affected by the actual topology being used. However, for simulation-based, qualitative evaluations, certain topologies have to be selected in order to provide a variety of results that provide a foundation for further assessments.

2.6 Simulation of Sensor Networks

2.6.1 Applications of Simulation

Testing an hypothesis about the properties of a system is ultimately only possible by examining the real system itself. However, relying on this method has



(a) (b) (c) (d) (e) (f)

Figure 2.1: Sample network topologies

severe drawbacks:

- For an artificial system under consideration, this system has to be built first before it can be studied, which is costly and time-consuming. Furthermore, testing an hypothesis may require to destroy the system, so it has to be built several times in order to fully analyze it.
- For natural systems, there exist similar problems, which are even more severe. There may exist only one instance of the system under consideration (e.g., the Amazonian rainforest, or the global climate), so testing certain hypotheses, which may have harmful or catastrophic consequences, is impossible.
- The act of observation often influences the results of an experiment.

For these reasons, certain properties are best studied by examining an abstract model of a system, not the real system itself. The two major model-based approaches are analytical methods, where results are formally derived from initial assumptions, and simulation. A simulation is a trace of a system's behaviour in terms of an abstract model of this system.

Simulation as become an important tool when building a new type of system, such as a new type of car, airplane, or computer system. Before an actual instance is being built, several models are constructed first, which reflect certain properties of the final product. At the same time, other properties are not reflected in the model, which are considered irrelevant for the specific hypothesis at hand. Thus, a model is an abstraction of the real system. For example, in order to study the user acceptance of a new software system, it is important to provide the "look and feel" of the proposed system together with some means for interaction, but it is not required to implement all underlying databases, networks, storage systems etc. The critical point is to make sure that observations on the model can be carried over to the real system. Only then it is possible to obtain knowledge about the real system by simulating (certain aspects of) its behaviour using the model. This is a fundamental limitation when working with an abstract model: only those properties can be examined that are mapped in the model.

Simulation, and in general also testing, is an imperfect tool for obtaining knowledge, since there is always the possibility that some aspect of reality is not reflected in the model. This could lead to a case where all simulations and tests are successful, but the real system fails. Of course, this is also true for analytical approaches when some critical aspect is not reflected in the initial assumptions. Therefore, all these model-based approaches can only increase

the confidence in the design of a system, but the ultimate test for success can only be provided by reality.

Building a test environment for sensor network protocols and algorithms is a tedious task. Sensor networks are a new technology, there are no off-the-shelf solutions for any application available today since almost all issues regarding usability, management, robustness, security, life cycle, and profitability are yet unsolved. Commercially available products do not seem to incorporate important design requirements such as self-organization or small size. Thus, creating a sensor network installation basically requires designing the system from scratch, especially when new algorithms and protocols are to be studied. Additionally, observing the behaviour of all nodes is difficult in a distributed environment. Often, nodes have insufficient resources for logging all events over a long period of time by themselves. An additional, complex infrastructure would be required to obtain the full data set.

Thus, simulation is an important tool for designing systems based on sensor networks. This comprises the full range from systems in which sensor networks are used as external tools, for example for providing contextual information, to software running inside sensor nodes themselves. Simulating a sensor network, instead of really building it, makes it possible to study its behaviour independently of events occurring in the real world. It saves the effort of building, deploying, and managing sensor nodes. It makes it possible to closely observe the behaviour of every single node during a run. Finally, simulation can reveal the network's behaviour under a large variety of circumstances, which would be too numerous to consider in reality.

Depending on the aspect to be studied through simulation, a certain model of the sensor network has to be constructed. There are two fundamental approaches to simulating sensor networks. The first is simulating single sensor nodes as closely to reality as possible, which means that all hardware and software components are reflected in the simulation model, often including a detailed model of the communication channel. The second approach abstracts from the individual behaviour of single nodes, rather considering the network's emergent behaviour. Both approaches are considered in more detail in the following paragraphs.

2.6.2 Node-based Simulation

In order to study the design of a new sensor node architecture, a model of the design is constructed in software only. This allows it to run a large set of simulated nodes, including their communications, on a single (workstation-

class) host, which makes it easy to reconfigure the nodes and run a wide range of tests in a completely automated manner with minimal effort.

An important component in a sensor network is the wireless communication medium being used. The performance and energy demands of a protocol directly depend on the characteristics of the medium and the communication interface. It is therefore important to be able to test new protocol designs and evaluate their costs before they are deployed on actual hardware platforms. Consequently, there exist sophisticated models of this low level layer that include characteristics such as radio propagation and interference. The goal is to reflect the properties of the real communication medium as closely as possible. For example, simulation environments like OMNeT++ [120], TOSSIM [115], or ns2 [130] allow for the specification of a model of the wireless channel. Some predefined specifications are already provided in most packages, which can be further refined as required. However, all of these models provide only an imperfect embodiment of reality, so in most works, the results obtained through simulation are validated by tests on real hardware, usually on a smaller scale than the simulation.

Node-level simulations encompassing the network stack are useful for evaluating the performance and energy characteristics of an implementation. They can also be helpful for testing whether potential security-relevant flaws exist in an implementation. Standard security evaluation techniques like penetration testing [7] and fuzzing [131] are usually applied to real-world implementations. However, they can also be helpful in uncovering vulnerabilities in the design of network protocols or system designs when they are applied to formal system models [168] or during simulations. In these cases, it is important that all relevant properties of all components of a network node are reflected in the underlying abstract model.

Security evaluations in this fashion study the security properties of single nodes. They assume an external attacker, i.e. some entity that is able to send messages through the communication channel. In reality, this could be some existing node that is compromised by the attacker, or some additional device through which the attacker is able to simulate a legitimate party. Depending on its actual manifestation, the attacker may have certain abilities that exceed those of the legitimate participants, such as excessive computational or transmission power. Usually, the attacker model proposed by Dolev and Yao [57] is assumed. This model represents an external attacker that has full access to the communication channel being used by legitimate participants. This means that the attacker can read all exchanged messages and inject own messages, as well as intercept messages and drop them. The attacker's computational abilities are

assumed to be superior. This allows it to evaluate the security of an entity (or a set of communicating entities) facing a powerful attacker. If no vulnerabilities can be found during a test of an entity, or even a proof of their absence can be found, this provides strong evidence of the entitiy's security.

2.6.3 Network-based Simulation

A network-based simulation is completely independent from implementation details. Nodes are modelled declaratively, i.e. their behaviour is described as a relationship between incoming and outgoing messages. Messages are never lost and are delivered instantly. The only characteristic of the communication medium that is carried over to the model is the limited range of a wireless connection, which determines the connectivity graph. Fluctuations in connectivity are disregarded, though. These simplifications lead to a static graph model of the sensor network, where the vertices represent the nodes, and the edges of the graph mirror the communication links between neighbouring nodes.

A simulation based on this model is executed as a message exchange between nodes. Some nodes create messages "spontaneously", meaning we silently assume that there is a good reason for some node to create a message. This very reason is unimportant for our results. A message is transmitted over the (wireless) communication medium. In most cases, we can assume a broadcast medium, such as electro-magnetic waves. But a directed medium is possible, such as light. A node that receives a message will either consume the message or transform it and then relay it to other nodes. The exact operation depends on the current state of the node. Possible states are, abstractly, *correct, malicious, failed*. The fundamental attacker model includes the attacker's capability to take full control over all nodes where the attack succeeds (this is what is accomplished by a "root kit" for computers connected to the Internet). Alternatives are possible, for example partial control over a node. Depending on its state, a receiving node transforms the message and relays it further to one or more of its own neighbours. This process continues until some node consumes the message without relaying it. The state of a node influences the transformation of messages that are going through a node. In our case, the state mainly reflects whether a node is controlled by the adversary or not. The state remains fixed during a simulation run.

The major advantage of network-based simulations over node-level simulations is scalability. Network-based simulations allow the simulation of much larger sets of nodes, since most node-internal details are disregarded. Thus, the state information being kept for each node is minimal. Also, possible in-

teractions between nodes, for example radio interference through concurrent medium access, are not considered. This simplifies the simulation of node processes, which can be executed strictly sequentially as interleavings between concurrent processes do not have to be considered. This allows the simulation of sensor networks on a much larger scale. A direct comparison between different simulation environments shows that network-based simulation is able to handle at least two orders of magnitude more nodes than node-level simulation [98].

In contrast to node-level simulation, simulation on the network level is more appropriate for evaluating and testing algorithms and protocols on a higher, more abstract level. This shifts the focus from implementation details to a global view, which is more appropriate for studying aspects such as collaboration, accuracy, correctness, failure tolerance, and security on a network-wide level. For example, Michiardi and Molva [123] study, by network-level simulation, the impact of malicious node behaviour on routing quality in mobile ad hoc networks. Their metrics include the fraction of dropped packets as well as delays.

In summary, we regard network-level simulation as an important tool in the following areas:

- Testing algorithms and protocols in large-scale networks during the design phase. Algorithms can be specified on an abstract level without the obligation of giving an implementation for a concrete platform. The environment is idealized, but the effects of natural disruptions, such as connectivity fluctuations, can be modelled if necessary.

- Studying emergent properties of large-scale networks. Certain properties emerge only from the interactions of a large number of nodes. They are not necessarily anticipated during the design phase, but could be uncovered by simulation done in an early phase.

- Validation of security assessments. The security of single nodes does not necessarily imply the security of the overall network. It is therefore important to provide means to establish security properties on a network-wide level. This is the most relevant point throughout this work.

2.7 Security Requirements

A number of general security vulnerabilities of sensor networks can be identified, such as in the transmission of messages, on the routing layer, or regarding

physical node capture. Depending on the anticipated attacks, these vulnerabilities may become security threats. It is therefore important to formulate the requirements that can help to avoid these (potential) threats. Here, we give an overview of the building blocks of sensor network security.

2.7.1 Message Transmission

A message sent from one node to another one should not be susceptible to manipulation or eavesdropping by an attacker. On the link layer, this can be achieved by encrypting and authenticating messages in transit. The necessary keys can be agreed upon when the wireless link is established. However, at that stage this procedure is vulnerable against man-in-the-middle attacks in the following way: As soon as the adversary detects a communication between two nodes, he intercepts the initiating message being sent by one node while at the same time jamming the other node's radio interface. He then sends his own initiating message to the second node. This results in both legitimate nodes having a "secure" link with the adversary.

To avoid this attack, some kind of entity authentication is required. This problem is prevalent in all wireless networks. Bluetooth, for example, requires a pre-authentication stage, called "pairing", where devices are introduced to each other through a common password. Here, authentication is mutual, as both devices are "convinced" that they are connecting to an authorized peer. The IEEE 802.11i [141] standard for WLAN security prescribes client authentication towards an authentication server before network access is allowed. Here, neither the network access point nor the authentication server need not authenticate themselves to the client, which leaves the opportunity for network spoofing.

Sensor nodes need to be sure that other nodes they are communicating with are legitimate participants in the network. This requires not necessarily the verification of another node's ID, but merely its membership in the group of legitimate nodes. It should be ensured through a light-weight process, preferably without the help of a base station acting as a trusted third party.

One solution is provided by the key predistribution schemes discussed in chapter 4. Each node is provided with a set of keys selected from a key pool. A node can verify the legitimacy of another node by challenging one of these keys that both nodes have in common. If public-key cryptography is available, signatures can be utilized for authenticated Diffie-Hellman key exchange.

Yet another solution may be possible in certain deployment scenarios. A valid assumption could be that for a short period after deployment, the sensor

network is not under attack. Thus, there exists a short time window during which key material can be exchanged as plaintext. This essentially replaces authentication by a (temporarily) secure environment.

Link-level security is, however, only effective against outsider attacks. A message that is transmitted over multiple hops is easily compromised if its path includes a malicious node. Therefore, a mechanism to set up end-to-end secure connections is desirable. However, this requires a means to establish a shared key between the endpoints of a connection, which turns out to be challenging in sensor networks. This problem is further discussed in section 3.4.

Sometimes, messages need to be delivered network-wide, for example code or configuration updates. Such messages originate preferably at a base station that is trusted by all nodes, and must be authenticated. The μTESLA protocol [142] has been designed for authenticated broadcast and is suitable for this use case.

2.7.2 Routing

Multi-hop routing is a fundamental service in large-scale sensor networks. There are protocols that are specifically designed to address the needs of sensor network applications. Usually, they do not rely on up-to-date routing tables, which would be too complex to maintain. Rather, they forward a message based on features of the message itself, or they set up paths on demand.

One important class of such protocols provide data-centric routing. They are either demand-driven, where a node that is interested in a certain type of events announces its interest and thereby pulls messages towards itself [86], or event-driven, where a message source announces the availability of messages of a certain type [31]. A path is then established between source and receiver. Another important class are geometric routing protocols [91, 100, 149, 201]. They rely on nodes knowing their and their neighbours' position in a real-world or virtual coordinate space. Messages are routed according to a target location without the need of setting up a path.

A routing protocol ensures that messages reach their target. Attacks on the network layer, where routing functionality is located, aim at diverting or suppressing messages. This can lead to unauthorized data disclosure, missing critical events, energy exhaustion, or event triggering at undesired locations. A secure routing protocol must be resilient against such attacks.

Karlof and Wagner [90] have identified a number of ways to attack routing protocols designed for sensor networks, and propose countermeasures. These include link-layer encryption and authentication, multipath routing, identity

verification, and broadcast authentication, which are able to protect against most attacks. Sinkhole and wormhole attacks are hard to defend against. A sinkhole attracts messages and prevents them from reaching their target. A wormhole is a shortcut through the network (an external low-latency link) that can be used to mount sinkhole, selective forwarding, or eavesdropping attacks. Data-centric routing protocols are vulnerable against these attacks since a path is established between the source and the receiver of a message. By offering superior routing properties (low latency, high energy resources), a node under the adversary's control can influence the establishment of these paths. Geographic routing protocols are less vulnerable to these attacks. For example, a wormhole could deliver a message to its intended target prematurely, which is hardly a violation of security. Assuming that the target address is included in a message and cannot be changed by the attacker, a node that receives that message through a wormhole but is not located at the target address itself would simply forward the message towards its actual target location. This changes the path the message travels, which may or may not be a security violation depending on the application context.

2.7.3 Access Control

Many sensor network applications deal with sensitive or commercially valuable data; queries are disseminated through the network, triggering nodes to activate their sensors and transmit data; actuators are triggered by control commands. All these actions are significant with regards to the (commercial) operation of the sensor network, and its interaction with its environment. Illegitimate use may have harmful consequences. Therefore, access to a sensor network should be restricted to authorized parties.

Access control plays an important role in ensuring the confidentiality and integrity of sensor network data, as well as the safe operation of a sensor network. Thus, an effective access control mechanism is required, preferably implemented in a distributed manner in order to allow arbitrary entry points into the network. This avoids the use of a centralized entitiy that acts as a single entry point, thereby constituting a single point of failure, and a performance bottleneck.

The access control mechanism should still be effective when the network is being attacked and some nodes have been compromised. Such a robust framework for access control in sensor networks is described in [17], where a certain minimum number of nodes have to agree in order to authenticate a principal requesting access. This avoids that a single, compromised node is able to grant

access to an illegitimate user.

A simpler approach to access control would be the use of a globally shared key for encrypting all traffic, which is regularly updated in order to provide backward secrecy. Such an approach is described in [13]. It protects against eavesdropping by outsiders, but would not help in case there is at least one node being compromised by the adversary. The underlying adversary model is fundamentally different from the one assumed in the previously described framework, but it is much easier to implement. Of course, there is no universal mechanism that can be applied efficiently in all cases. Eventually, the application scenario determines the appropriate mechanisms required for providing access control.

2.7.4 Data Aggregation

One of the main tasks of sensor networks is data aggregation, i.e. the combination of data gathered by various sensors into a single value that is meaningful within the application context and represents the monitored state as accurately as possible. This process must not only be robust against random errors, which are likely to occur, but also against malicious nodes reporting intentionally falsified sensor data.

Generally, it is not feasible to detect whether a sensor reports the data it has obtained from its sensors correctly, as this would require a second sensor node that performs the same sensor readings. Due to the high redundancy in a sensor network, it would seem likely that in most cases, there would be sufficiently many nodes close to any other node. However, there are two reasons that argue against such a solution. The first ist cost. Keeping all nodes actively monitoring their environment all the time depletes the energy sources of all these nodes. It would be more desirable to exploit the redundancy for extending the lifetime of the network, replacing depleted nodes with others that have saved their energy. The second reason is that the adversary who has managed to compromise one node is likely to be able to compromise, or at least disable, the nearby nodes as well. These would then rather support the falsified readings of the first node than dispute them.

If some correlation can be assumed between the sensor readings of nodes within a certain area that goes beyond the immediate reach of the adversary, nodes may still monitor each other's readings and report gross aberrations. If a node constantly reports data that is inconsistent with its neighbours' readings, it might be expelled from the network. However, such a mechanism must be careful not to miss important events whose patterns may be misjudged as

falsified data (false positives in the sense of an intrusion detection system).

Existing approaches for securing the process of data aggregation [146, 186] aim at minimizing the error that may be introduced by a fraction of malicious nodes. Additionally, the data reported by aggregating nodes may be rejected if it is discovered that the underlying raw data is inconsistent with the aggregaded value. This verification can be performed by sampling a small portion of the raw data.

2.7.5 Location Verification

In most sensor network applications, it is of importance not only *what* phenomenon, but also *where* it has been detected. When the location is reported incorrectly, responsive actions may be misguided. This not only wastes resources but also leaves the location where the phenomenon actually occurred unattended. A malicious report with a falsified location can thus inflict heavy damage.

A solution to this problem is the verification of the location of the reporting node. If the reporting node has to convince other nodes that it is indeed located at the reported location, it is much less likely that a falsely reported location is being accepted.

A technique for location verification is proposed in [158]. It assumes that the location verifier and location prover can communicate via a radio interface. Additionally, the prover has to generate an ultrasound signal which the verifier receives. First, the prover announces to the verifier through the radio interface its distance from the verifier. The verifier then sends a nonce to the prover (also through radio) which is immediately reflected by the prover on the ultrasound channel. If the roundtrip time of the nonce is within appropriate limits, the verifier can safely assume that the prover is within the the announced range. Through triangulation using multiple cooperating verifiers, the exact position of the prover can be determined.

A different approach for distance bounding, which is based on a single communication channel that could be radio-frequency or optical, is described in [77]. It demonstrates that distance bounding is achievable at quite low cost. Its usage in wireless senos networks thus seems feasible. However, the approach as described relies on an asymmetric architecture as it is intended for RFID tags or contactless smart cards. The verifying node (RFID reader) carries a much larger burden than the node (RFID tag) proving its location. In addition, several constraints, such as ultra-wideband, are imposed on the radio channel.

2.7.6 Intrusion Detection

The impact of an attack can be greatly reduced if it is detected as early as possible. Intrusion detection systems (IDS) have therefore grown to major security tools in Internet networking. Their purpose is to detect patterns in system behaviour that indicate malicious activities by both outsiders and insiders. There are two basic types of intrusion detection systems. The first one is host-based, i.e. activities on a host are monitored for anomalous patterns (work in this area goes back to [50]), e.g. extensive resource allocation or suspicious system call sequences. The second type is network-based, i.e. network traffic is monitored for attempts to exploit weaknesses in the network stack (a popular system doing that is Snort [156]).

In a sensor network, an intrusion can occur at two levels. Either one or more sensor nodes have been taken control of by the adversary, or the adversary is disrupting the sensor network's operation through external means. The latter type of attacks can be more easily detected and defended against. If the adversary manages to take control of sensor nodes, however, he gains access to sensitive information stored on these nodes, e.g. cryptographic keys, and is able to participate in the network's operation. If the compromised nodes do not show obviously aberrant behaviour, they may go undetected. The adversary can then use them for eavesdropping or subverting the network by injecting false messages in such a way that these actions will not be appearant to legitimate nodes.

Host-based intrusion detection is not directly applicable in sensor networks, as it must be assumed that once the adversary has gained control of a sensor node, he completely controls all processes running on that node. This would allow him to disable any intrusion monitoring process. The reason is that an embedded computing platform, such as a sensor node, does not provide the required level of memory protection and process separation that is available on a high-end platform with the appropriate operating system and hardware support.

It seems that in a sensor network, one has to concentrate on the behaviour of nodes that is observable by other nodes in order to detect malicious nodes. Since the functionality of sensor nodes is typically very restricted due to their computational constraints, it seems feasible to specify legitimate observable behaviour of sensor nodes on a detailed level. This behaviour includes monitoring frequencies, sleep schedules, and communication patterns. An intruder who wants to make use of the nodes he has compromised would be required to deviate from these specifications in one way or another, which could then be

detected by neighbouring nodes.

Legitimate nodes should cooperate in identifying and classifying abnormal behaviour of their fellow nodes. However, the attacker may not change the behaviour of the nodes significantly in the beginning, such that neighbouring nodes cannot detect the intrusion by simply observing the compromised nodes. This gives the adversary time for compromising even more nodes and then start a large-scale attack involving many nodes distributed over a large area at the same time.

In order to detect such intrusions, an active approach is needed such as described in [174, 183]. Changing the behaviour of a sensor node requires changing its program code. As there is only limited memory on a sensor node it is likely that in order to apply the changes, the old (legitimate) code has to be erased. Neighbouring nodes can detect these changes by putting out randomized challenges that require the challenged node to prove that it possesses the current legitimate program code. If a node fails to prove it, it could be expelled from the network, or its program code can be updated by its neighbours. Such an update mechanism requires that the bootloader part of the sensor node is kept in a tamper-proof module.

There are similarities between sensor networks and ad hoc networks that are relevant to intrusion detection techniques [202]. One such similarity is the lack of a communication infrastructure. As there are no central routers, each node has to rely on its own audit traces (the network traffic it has monitored over time) in order to make decisions about possible intrusions. These traces are limited to the vicinity of a node and therefore provide only partial information. The limited storage capacity restricts the amount of potential evidence that can be stored, and the computational power avaialable for evaluating this data is limited. On the other hand, intrusion detection seems to be easier in sensor networks as the behaviour of nodes is much more restricted and homogeneous than in general ad hoc networks.

2.7.7 Intrusion Tolerance

One problem with intrusion *detection* is that the adversary may be able to adapt the behaviour of compromised nodes in such a way that their aberrant behaviour is not classified as such. Thus, these nodes operate seemingly normally and are able to influence the overall operation of the network.

In order to introduce some level of intrusion *tolerance*, it is advisable not to rely on the reports from a single node but instead require some agreement by a number of nodes before a report is accepted and further processed. A number

of techniques can be used to achieve this. For example, multiple sensor nodes may transmit their readings independently to the query issuer, or a consensus procotol is executed within a cluster of nodes before a report is generated.

Threshold schemes [122] have been proposed [203] in the context of ad hoc networks, but they are similarly applicable in sensor networks. A (k,n) threshold scheme ($k \leq n$) lets any subset of k out of n principals compute a function F. Such a scheme has the following general features:

- k elements are sufficient to compute F;
- no subset of less than k elements can compute F.

This would allow a set of k nodes to collaboratively create a signature of a query result that can be verified by the query issuer. The issuer would then be sure that at least k nodes agreed on the result. If there are less than k compromised nodes, the result has been approved by at least one legitimate node. On the other hand, as long as there are k non-compromised nodes, the signature can be generated, providing robustness against denial-of-service attacks.

One problem with threshold schemes is their applicability in sensor networks. For the combination of partial results, the individual shares have to be revealed to a combining service. In a sensor network, this service would have to be provided by a sensor node. If this node is compromised, the adversary will learn k shares of legitimate nodes and will thus be able to compute signatures on his own in the future.

To avoid this, new shares may be distributed after each signature generation, which is very costly in terms of message complexity. Alternatively, the combination of partial results may be done only by a (trusted) base station, which requires an additional component in the network architecture.

A better approach may be to introduce epochs, i.e. time intervals during which the shares are valid. After each epoch, new shares are distributed to all nodes, and the combination service is migrated to a different node. This limits not only the complexity for distributing new shares but also the ability of the adversary to exploit the shares he may have learned through a compromised node running the service.

2.7.8 Availability

Several factors can reduce the lifetime of a sensor network, for example naturally occurring phenomena such as extreme temperature variations exhausting the batteries. An attacker could force sensor nodes into frequent retransmission of messages by selectively inducing errors. Over time, this leads to the failure

of nodes and eventually in the inoperability of the overall network. Such disturbances may not be easily identified as malicious interferences, and therefore no effective countermeasures exist. An attentive observer may notice a higher failure rate than expected, but still the cause may remain obscure.

While battery exhaustion attacks are effective, see [170], they require some sophistication on behalf of the adversary and are time-consuming. For example, the attacker has to match the activity cycle of the victim device. For an immediate effect, more radical approaches are required, such as the physical destruction of nodes or jamming the communication channels. However, most of these attacks can be easily detected by those nodes being unaffected.

For example, if the attacker is jamming a region of the network, nodes within this region cannot receive messages anymore. However, they will notice that they are being jammed and may be able to issue messages reporting the attack. Nodes at the border of the jammed region pick these messages up. They can then further report the attack to the operator, and they can set up paths for routing messages around the jammed region such that the operation of the rest of the network is not affected. If jammed nodes are not able to send anything, the nodes at the border will have to assume that some of their neighbours have failed. A geographic routing mechanism will then automatically start to route messages around that dead area [201, 198].

Jamming and physical destruction of nodes are simple forms of denial-of-service attacks that can be compensated for if they appear only on a small scale, i.e. if only a small area is affected. Sometimes, the effects of a denial-of-service attack can be mitigated if nodes are able to extend their sleep cycle when they notice that an attack is going on, such that they conserve as much power as possible. This would make the sensor network inoperable during the attack, but at least it can continue operation once the attack has ceased. Of course, if an exhaustion attack is successfully executed on a large scale, affecting large areas, there is no possiblity to recover other than by deploying new nodes after destruction.

2.8 Cryptography for Sensor Networks

Cryptography plays an important role in securing networked computer systems. It provides the basic functionality for protecting the confidentiality, integrity, and authenticity of messages and data. Here, we present the cryptographic building blocks that will be important in later chapters. Their main applications will be key agreement and message authentication. We include a separate section where general issues in key management are discussed, which will not

be intensified later.

2.8.1 Hash Functions

A hash function is a mapping from a set of *documents* of arbitrary length to a set of *hash values*, which have a fixed, small size. A cryptographic hash function $h : A \rightarrow B$ has the following properties ([122], ch. 9):

1. For any $x \in A$, the hash value $h(x)$ is easy to compute (in linear time).
2. For any hash value $y \in B$, it is computationally infeasible to find a $x \in A$ for which $h(x) = y$ (one-way property/preimage resistance).
3. For a given $x \in A$, it is computationally infeasible to find a x' for which $x' \neq x$ and $h(x') = h(x)$ (weak collision resistance/second preimage resistance).
4. It is computationally infeasible to find any two distinct $x, x' \in A$ for which $h(x) = h(x')$ (strong collision resistance/collision resistance).

A hash function produces a small representative (also called fingerprint or message digest) of an input document of arbitrary size. As the cardinality of the domain A is greater than that of the range B, the existence of collisions is unavoidable. However, for practically useful hash functions, it is computationally infeasible to find such collisions. This allows it, for practical purposes, to identify the output of a hash function with its original input and use it as a substitute, for example in the process of creating a digital signature of the original input.

To an observer who only sees the result y of a computation $h(x)$, without knowing x, the value y seems "random" in the sense that it is unknown how the value has been created, and it could as well be drawn randomly from B. This is a consequence of properties 2 and 3: it is practically infeasible to either find the original value x or any other x' with which y could be reconstructed. Ideally, each value from B appears with equal probability and thus a uniform distribution can be assumed.

Practical infeasibility refers to the computational power that would be required in order to find a preimage or a collision. The number of computational steps required directly corresponds to the length n of the hash function's output. Without additional knowledge, finding a preimage or a weak collision would require 2^n steps, while finding a strong collision would require $2^{n/2}$ steps (cf. [122]). With a sufficiently large n, often 2^{80} steps are considered sufficiently hard as of today's available technology, it is impossible to break the security of a hash function within reasonable time.

Hash functions are useful in many contexts such as pseudo random number generation (see the previous argument), key generation, or as a building block in cryptographic protocols. Some use cases are collected and described in [14]. We will use them primarily for key agreement and message integrity protection.

2.8.2 Message Authentication Codes

A well-known construction for authentication codes is HMAC (hashed message authentication code, see e.g. [21, 133]). This construction is based on a keyed hash function, allowing it to authenticate the hash value of a message using a cryptographic key. Assuming that the key being used is only known to the sender and the receiver, and that both the sender and the receiver are trusted, the receiver of a message is able to derive its authenticity. Since knowledge of the key is necessary for constructing the MAC, the receiver can be sure that nobody else than the purported sender could have constructed the message. This type of authentication does not provide the ability to prove to a third party the authenticity of the message, since the receiver could have constructed the MAC itself.

The formal definiton of an HMAC is given in [16]. Simplified, it is given as

$$\text{HMAC-}h(k,m) = h(k' \| h(k'' \| m))$$

where k' and k'' are padded versions of k, h is an arbitrary hash function, and m is the input message. In order to verify the authenticity of a message using an HMAC, the receiver has to compute the HMAC value in the same way as the sender. In particular, this means that the message has to be available in plaintext to the verifier.

This is in contrast to digital signature schemes which provide message recovery, where the successful verification of a signature yields the original message. For example, RSA can be used as such a signature scheme for small messages, i.e. messages of length up to half the length of the modulus (see Chapter 11 of [122]). For larger messages, however, the message is being hashed before encryption and thus message recovery is not possible.

The HMAC construction has the disadvantage that if multiple MACs have to be created for the same message but using different keys, the complete message has to be rehashed for each MAC. As the key k is prepended to the message, this effectively modifies the message each time. In this case, using the secret suffix method for creating a MAC is more efficient. Here a MAC is created as

$$\text{MAC-}h(k,m) = h(m \| k) \ .$$

Using this construction, the message has to be hashed only once. Arbitrarily many MACs can be produced simply by hashing the result together with the respective keys.

This construction has the disadvantage that an attacker can perform an off-line attack trying to find a message m' that has the same MAC as m. Since the key is involved only in the last step of the application of h, any message m' that yields the same hash output as m will lead to the same MAC value. Thus the task of the attacker is to find m' for which $h(m') = h(m)$. No knowledge of k is required for this. The security of this construction therefore depends on the collision resistance of the underlying hash function.

2.8.3 Symmetric Ciphers

Encryption is an important tool in security as it guarantees the confidentiality of data. It makes it possible to store or transmit sensitive data such that it is not susceptible to disclosure to unauthorized parties even if such parties gain physical access to the encrypted data. A *cipher* is a method for turning a message into a secret code. For digital data processing, there exist two distinct approaches, one called *stream cipher* and *block cipher*

In sensor networks, we are mostly interested in protecting discrete messages that are about to be transmitted over a wireless channel, so we are mainly interested in *block ciphers*. These operate on a block of data for a single encryption or decryption operation. If a message is larger than the block size of the used cipher, it is split into multiple pieces that are then operated on independently.

The most recent standard algorithm for symmetric encryption is AES [132], which can also be efficiently implemented on current WSN platforms.

2.8.4 Implementation

There exist a large number of practically usable hash functions, of which the best-known are the SHA function family, MD5, and RIPEMD. These can be implemented on BTnode sensor nodes, usually without modifications to available ANSI-C code as they require mainly integer and bitwise operations only. Table 2.2 shows the runtime of some of these algorithms (together with the AES block cipher) on the BTnode platform. Note that the implementations have not been optimized for that platform but have been used as provided in their respective sources.

Although a hash function can take input of arbitrary length, it internally processes data blocks of a specific length. In order to process input of arbitrary length, it splits the input in blocks of fixed size and iterates over them,

Algorithm	Block size	Digest size	Block operation	Bits/s	Code source
AES-128	128 bit	n/a	2.13 ms	60,093	[70]
AES-256	256 bit	n/a	2.95 ms	86,779	[70]
MD5	512 bit	128 bit	2.83 ms	180,918	[155]
SHA-1	512 bit	160 bit	6.99 ms	73,247	[51]
SHA-256	512 bit	256 bit	11.7 ms	43,760	[127]

Table 2.2: Execution time of cryptographic primitives on the BTnode platform (own measurements)

combining the result of the previous interation with the current block using a compression function.

2.8.5 Bandwidth Overhead

Messages in sensor networks are potentially very small. For example, if only current sensor readings are transmitted, a message may contain only a few (less than ten) bytes of payload. Sensor network protocols on the medium access layer therefore often provide the means for transmitting short messages without inducing unacceptably high overhead for synchronization and error control.

Adding a message authentication code (MAC) to such a small sensor message adds a significant overhead. For example, the SHA-256 hash function produces digests of 256 bit (32 byte) length, which may exceed the size of a sensor message by far. This would mean that a significant amount of the transmission energy and time had to be spent on authentication information, which is hardly acceptable in a resource-constrained environment.

One solution would be to increase the size of messages in order to reduce the relative portion of authentication data. This would be possible for messages containing aggregated data or combining the readings from various sensors. However, this would introduce a latency for the delivery of sensor data. In application-driven scenarios, where the timely availability of data is crucial, this is not a feasible option.

An alternative is to transmit the message authentication code only partially. Instead of attaching all 256 bits to a message, for instance only the first m (e.g. $m = 32$) bits are used. The receiver is still able to verify the message's authenticity by computing the (full) authentication code of the message and comparing its first m bits with the received data.

Partial MACs offer less security than full ones, but their level of security is adequate for practical purposes. The output of a hash function can be regarded as random data. Constructing a valid HMAC without knowledge of the key is possible only by a brute-force approach and requires 2^n steps, where n is the

output length of the hash function. If this output is truncated to, for example, its m-bit prefix, where m is large enough (depending on the anticipated attacker strength), this would still provide a sufficiently high security level.

It is important to distinguish between offline and online attacks on a MAC. If only online attacks are possible, for example if HMAC is being used for creating MACs, a relatively small MAC size would be adequate. As an example, assume a sensor node requires a time of $t = 10$ ms for receiving a message and verifying its authentication code. With a bitlength $m = 32$ of the authentication code, around 2^{32} attempts are necessary before the sensor node accepts the message as being valid. Thus, the node would accept the message only after approximately 5965 hours (248 days). Obviously, running under constant load, a battery-powered sensor node would run out of energy long before that time has passed.

The same MAC length would be insufficient to prevent offline attacks, however. In offline attack, the computations are performed without interaction with the attack target and can be massively parallelized. To prevent such attacks, at least 80 bits are required, which would still lead to a 10 byte overhead per MAC.

2.8.6 Key Management

Cryptographic keys are necessary to establish secure channels between communicating nodes. The main tasks of key management are the distribution, refreshment, and revocation of keys. In centralized architectures, one can use key distribution centers (KDC) to which nodes securely connect (by means of a predefined secret between the KDC and each node) in order to receive updates on their keys or to retrieve new keys for pairwise communication among nodes.

Generally, sensor networks are highly distributed, multi-hop systems in which communication to a central entity is expensive. Transient disconnectedness of parts of the network would even make it impossible to reach a KDC and thus prevent nodes from exchanging messages. Therefore, a distributed solution to key management in sensor networks is preferable.

One viable approach is to pre-load a set of keys onto each sensor node before deployment. This can be done in a secure environment that has no tight energy restrictions. These keys can serve various purposes after deployment:

- Secure interaction with the base station – Each node is assigned a secret key that is only known to the node itself and the base station.
- Key agreement between nodes – A key agreement scheme based on (random) subsets of a large key pool such as proposed in [64] can be used to

establish pairwise secret keys between arbitrary pairs of nodes.

- Entity authentication – By means of a challenge/response protocol, a key can prove its knowledge of a certain secret key and thus demonstrate its identity and liveness.

Once a key has been established between two communicating entities (either two sensor nodes or a node and the base station), this key can serve as a master key for deriving actual communication keys that are only used for a certain time span. This makes cryptographic attacks on the communication harder, since only a limited amount of data is available for cryptanalysis. Also, if a communication key happens to be exposed to the attacker, only a limited amount of data is compromised. In sensor networks, long-term relationships between pairs of nodes exist mainly between neighbouring nodes. Generating a fresh communication key for their communication is desirable. Long-range communication between nodes is usually sparse and not bound to distinct nodes but happens rather between node clusters or groups. Frequent updates to these communication keys may not be necessary in many cases.

Key revocation in sensor networks has the goal of excluding specific nodes from future communication after it has been detected that the key material of these nodes has been exposed to the attacker, for example after a node capture. Such nodes must not be allowed to further participate in the operation of the network. Additionally, the key material shared between compromised nodes and others should not be used anymore and keys that have been established based on this material may have to be renewed. This is especially important in cases where it has to be assumed that the attacker has recorded all previous traffic.

Revoking keys is an expensive operation in a sensor network as it cannot be expected that nodes regularly check a central repository of revoked keys. Thus, revocations have to be actively distributed throughout the network to be effective. These messages may become large if a large amount of key material is affected such as in pool-based schemes. The follow-up key re-negotiations put further load on the nodes.

Another problem is the detection of compromised nodes, which is necessary to initiate a key recovation procedure. In general, this is only possible through aberrant behaviour of nodes or through external means such as surveillance. A sophisticated attacker might avoid detection by not substantially altering the behaviour of captured nodes, and surveillance may not be possible or too expensive.

Yet another problem of revocation is possible abuse by an attacker. It may

be possible that by capturing a small number of nodes, the attacker is able to make secure communication completely impossible by initiating extensive key revocation. Thus, it is important to address the correctness and robustness of key revocation mechanisms. Formal definitions for these concepts are provided in [36].

2.9 Existing Approaches to Wireless Sensor Network Security

The emergence of wireless sensor networks as an object of academic research as well as practical engineering has initiated the development of such networks towards a tool that can be applied in many environmental, industrial, and social environments. This development has triggered an intereset in the security of such networks, as they are vulnerable due to characteristics such as deployment in open environments, wireless communication, and the absence of close administrative surveillance. Frequent node failures, changing topologies, and resource restrictions make the design of algorithms and protocols challenging, also including security mechanisms. Usually, techniques used in "traditional" computer networks are not directly applicable. This section discusses approaches to security that have been especially developed for wireless sensor networks.

Security for wireless sensor networks is of paramount importance because they are closely interlinked with the physical world, in which goods and people are moving and interacting. Data collected by a sensor network may therefore reveal sensitive data about personal behaviour, lifestyle, or business secrets. Although such data may be obtained through other means as well, sensor networks could potentially provide it to remotely located parties in an automated manner, which makes the protection of such information essential for guaranteeing the security and privacy of people [37].

Denial of service attacks are a serious threat in computer networks, as they disrupt work flows, cause annoyance, and may even be harmful to human lives, e.g. if emergency services become unreachable. Sensor networks are susceptible to denial of service attacks as well; their vulnerabilities on all protocol layers are discussed in [198]. Due to their nature, certain attacks on wireless sensor networks on the physical layer cannot be prevented, such as the jamming of the wireless communication medium. Higher layers are more approachable to countermeasures, as will be further discussed in this section.

2.9.1 Key Distribution and Agreement

Before the invention of public-key cryptography by Diffie and Hellman [54], secret keys used for enciphering and authenticating messages had to be distributed over secure channels, such as by courier. Then, it became possible to send messages that could only be read by the intended recipient without having to exchange a secret key beforehand. It became also possible to sign messages such that their authenticity could be verified by anybody. This is possible by the use of a key pair instead of a single key. One key is kept secret by the owner of the key pair, while the other key is published. Because of the convenience they provide, public-key cryptosystems, such as RSA [154], are widely applied today, for example in smart cards [78] and for the TLS protocol [53] used in the world wide web.

Public-key cryptography is widely considered too expensive for resource-constrained sensor nodes, since the involved operations are time- and energy-consuming [142, 64]. On a typical, contemporary sensor node hardware platform, RSA private-key operations consume 10 seconds and more (depending on key size), and cryptographic operations based on elliptic curves in the order of seconds [74, 22, 190]. Energy-wise, a single key exchange based on RSA is reported to consume as much energy as the encryption (and decryption by the recipient) with AES of approximately 77 kbyte data [187]. Whether these resource requirements make the use of public-key cryptography generally infeasible in sensor networks is debatable and will in practice depend on the used hardware platform, the frequency of public-key operations, and the available energy resources. The availability of public-key operations may even be dangerous to the sensor network. It has been noted that the possibility for a node to create signatures could be exploited by an attacker for draining the batteries of sensor nodes [5].

In order to be prepared for the case where insufficient resources are available for public-key cryptography, it is reasonable to come up with alternatives. The most important use of public-key cryptography is for handshake, i.e. the exchange of a secret key. This key, which is shared only by the two parties involved in the handshake, can be used for encrypting and authenticating messages. Therefore, techniques for establishing secret keys without using public-key cryptography are desirable.

A simple solution to the problem is pairwise key predistribution. Here, every node receives in advance one secret key for every other node in the network. This approach is not only limited by the memory capacity of sensor nodes, but is also inflexible regarding the extension of the network during runtime.

A more flexible technique has been proposed first by Eschenauer and Gligor [64]. It is based on random key predistribution and provides stochastic security against node capture attacks, i.e. if only a small number of nodes are captured, there is a high probability that a key exchanged between two nodes remains secret. There have been several extensions and improvements proposed to that basic technique [39, 143, 204, 38].

The earlier proposals for group key management by Blom [24] and Blundo et al. [25] were the foundation for other key agreement schemes [117, 60, 161]. These exhibit a threshold property: unless a certain number of nodes have been captured, all pairwise keys remain secure with high probability; if the threshold is exceeded, all pairwise keys are almost immediately compromised.

Other variations are based on combinatorial designs [110] (where the assignment of keys is deterministic, but the resilience against capture is nevertheless stochastic), or assume a slightly different attacker model [5]. In the latter case, it is assumed that the attacker can monitor only a small fraction of message exchanges during an initial time frame after deployment. It is shown that although keys are transmitted in clear text during that phase, only a small fraction of keys is actually compromised.

2.9.2 Secure Communication

Key agreement as described in the previous subsection is a prerequisite for general secure node-to-node communication, either on the link level between neighbouring nodes, or between remote nodes that are separated by multiple hops. For more constrained communication patterns, more economical techniques are conceivable.

One of the most light-weight protocols is μTESLA [142]. It is intended to be used in base station-centric networks, where the most prevalent communication patterns are point-to-point between a base station and a node (e.g. for queries and reporting sensor readings), and broadcasts from the base station to all nodes (e.g. for queries or reprogramming the entire network). It only assumes that every node shares a unique key with the base station (for point-to-point messages between the base station and the node). Using this key, the broadcast authentication mechanism can be bootstrapped.

This broadcast authentication mechanism is based on hash chains and loose time synchronization. For bootstrapping the mechanism, the base station sends an authenticated (using the pairwise key) element of the hash chain to a node, together with timing information. The hash chain element is a symmetric key that is valid only during a certain period in time. During this period, it is used

to authenticate messages from the base station, but it remains unknown to the receiving nodes. This means that nodes must buffer these messages first. Only after a certain delay after the end of its validity period, it is disclosed by openly broadcasting it. At that moment, nodes can (1) verify the authenticity of the key (by computing the according step in the hash chain) and (2) authenticate the messages that have been sent by the base station.

The deferred disclosure is necessary in order to avoid abuse of the key by malicious nodes or an external attacker. If the key would be disclosed before the validity period ends, it could be used by an attacker to inject authenticated messages.

The key is disclosed only after the end of its validity period, plus the disclosure delay. This is required in order to compensate for slight misalignments in the nodes' clocks. Thus, tight time synchronization between all nodes and the base station is not required.

μTESLA is extremely efficient to implement, as it requires each node to store only two keys, and the only cryptographic operation is the computation of hash functions (message authentication is based on the same hash function). The overhead per message is caused by the message authentication code, which is 8 byte using RC5 as the hash function, which was used in [142]. The key disclosure messages incur little overhead since they can be piggy-backed on other messages from the base station. Only if no other traffic is generated, extra messages for key disclosure have to be sent.

The reverse communication pattern to broadcast is aggregation. Here, data collected by sensor nodes is transmitted towards the base station (or another data sink). The individual data packets are not simply concatenated and passed on by intermediary nodes, but rather the data is combined and compacted. This minimizes the size of transmitted data but preserves its usefulness for the application. For example, the average, minimum, and maximum temperature in a room could be determined in this way without having to preserve every single data item.

The threat to data aggregation is that malicious nodes could falsify the aggregation result by either injecting false sensor data of their own, or by passing on falsified aggregation information. This cannot be completely prevented, as false sensor data may be indistinguishable to a faulty sensor. Also, as sensor data from different sources is integrated, authentication information of individual data packets is lost.

The work described in [146] and [186] approaches this problem from a statistical perspective. The objective is to reduce the possible error introduced by malicious nodes below a certain threshold. Data aggregation is also discussed

in 2.7.4.

2.9.3 Secure Routing

Cryptographic keys, being established through techniques discussed in the previous subsection, help to ensure secure communication with regard to the confidentiality and the authenticity of messages. It has to made sure, however, that messages indeed arrive at their destinations without being misrouted or dropped.

Ariadne [82] is a route discovery protocol that is able to find routes from a source to a destination in a multi-hop network and pass them back to the sender. It is secure in the sense that (1) the hosts authenticate themselves and (2) each host certifies the previous piece of the path from the source to itself. (3) All host identities are linked through a hash chain. (4) A basic assumption is an end-to-end secure link between the source and the destination. (This could be provided by a pre-arranged secret key between both hosts.)

It is assured that malicious hosts cannot cut nodes off the path or insert new ones. h_0 is the initial value, the authentication code of the initial request message. This is equivalent to a digitally signed request. Each following host identity X is appended to the hash chain as $h_{i+1} = H[X, h_i]$ where H is a one-way function. Destination D can easily verify whether the hash chain matches the path, since it can reconstruct h_0. Requests with mismatching items will be dropped. Intermediate hosts cannot do this verification step and will forward also bogus messages.

Ariadne does not prevent a malicious host from getting on the selected path if it follows the protocol. For example, it could then later drop messages being sent along the path. On the network layer, this will be noticed as a path failure. A new path discovery request would be issued by the source. Unless the malicious node creates a bottleneck between source and destination, a path will be eventually found that passes by the malicious host.

This protocol is not well-suited for wireless sensor networks for several reasons. (1) The route request is flooded through the network. This is efficient only in highly dynamic networks. (2) Source routing, for which the protocol provides the basis, requires node identifiers in the path to be sent along the node each time the route is used. This is undesirable overhead. The alternative, caching routes, would require additional memory in hosts. (3) A secure channel between source and destination hosts is a prerequisite. This implies that the identity of the destination is known to the source in advance. In sensor networks, such close links between hosts (sensor nodes) are not a common

communication pattern.

The most important attacks against sensor networks on the routing layer are identified in [90]. They comprise:

- Bogus routing information
- Selective forwarding
- Sinkholes
- Wormholes
- HELLO floods
- Sybil attack
- Acknowledgement spoofing

All these attacks have the goal of misdirecting or suppressing message traffic, or tamper with the topology of the network (HELLO floods and Sybil attacks, where node identities are announced in a multitude of locations, respectively new identities are created). If the attack is successful, the routing layer of the sensor network will not be able to deliver messages to their destinations.

Countermeasures against these attacks are also discussed in [90]. Geographic routing, for example, is a means to prevent wormhole and sinkhole attacks. These attacks rely on the fact that a node selects the next hop based on some metrics that can be faked by the attacker. Geographic routing relies on location information only, so a sinkhole cannot be easily created since its attraction does not extend to remote locations. Wormhole links, which span large distances, raise suspicion because alleged "neighbours" are located far beyond radio range.

Selective forwarding may be thwarted by multipath routing. If on one path, a message is dropped by the attacker, it may still get to the target through another one. However, using multiple disjoint paths implies a significant overhead and they may be hard to construct. A novel method for constructing them will be presented in chapter 5.

Mutual authentication between nodes, possibly involving a base station, can help against HELLO floods and Sybil attacks, since these attacks are based on creating non-existant node identities all over the network. If every node has to authenticate itself and is showing up in different places or has an unrealistic high number of neighbours, an alarm may be raised.

Ariadne is immune to some of the described attacks, but not to all of them. For example, Sybil attacks are prevented by node authentication, but a sinkhole

attack could be executed by an hybrid attacker who has obtained key material from a compromised node and has a strong transmitter at his disposal. This is based on the fact that in *Ariadne*, the hop from which a message is received first will determine the resulting path.

There is a large body of work that approaches routing security for ad hoc networks, part of which is applicable to sensor networks. It includes defenses against wormhole attacks [83], multiple paths for tolerating malicious nodes [49, 68], and cooperation and incentives in ad hoc networks [32, 34].

2.9.4 Available Implementations

Sensor network systems are still evolving, both in terms of hardware platforms, operating systems, and support libraries. This diversity is reminiscent of that existing in general embedded systems [66] and is in sharp contrast to desktop- or server-oriented computing, where the market is dominated by a small number of operating systems. There are important differences between the requirements on applications for embedded systems and for desktop/server systems. One is the fact that the hardware platforms for embedded systems vary widely (e.g., from 8 bit to 32 bit CPUs, or regarding their I/O interfaces), while the desktop and server markets rely on products that are highly standardized. Another difference are the markets for both kinds of systems. The market for embedded systems applications is highly fragmented. There is a multitude of devices in which embedded systems are integrated, all with different target groups and greatly varying lifecycles (e.g. short-lived consumer products vs. industrial machinery). Only time will tell whether sensor networks will be based on standardized platforms in the future, or the diversity in this domain will prevail.

One of the currently most widely used operating system for sensor networks is TinyOS [176], which supports a variety of hardware platforms. This system is also a popular research platform for security techniques in sensor networks. A library for link-level security based on symmetric cryptography, TinySec [89], has been incorporated into TinyOS. Another library providing Elliptic Curve Cryptography operations is also available [116]. To our knowledge, no other operating system for sensor networks provides cryptographic functionality by default, although standard algorithms like AES or SHA can be easily ported to sensor platforms (portability has been among the design criteria for most of these algorithms). Also, there are no standard implementations for end-to-end key agreement, secure routing, and other high-level security mechanisms.

2.10 Summary

This chapter highlighted several aspects of wireless sensor networks, thus establishing the technical context of this work. By considering the capabilities and limitations of WSNs, we are able to formulate high-level security requirements. The cryptographic foundations for meeting these requirements have been laid out.

We have also reviewed the existing relevant approaches to WSN security. Building on this knowledge, we proceed to describe a security model for WSNs, which will be the basis for the following security protocols we propose.

Chapter 3

A Security Model for Wireless Sensor Networks

Wireless sensor networks inherit security vulnerabilities from three classes of computer systems: (1) As classical computers, they are vulnerable to malicious code such as viruses. (2) Since they use wireless communications, their communication links are susceptible to eavesdropping, jamming, or message injection attacks; they share this characteristic with mobile computers. (3) Sensor networks often operate in unsupervised, open environments, where they can be physically accessed by potential adversaries; in this regard, they resemble embedded systems.

In order to counter the threats stemming from these vulnerabilities, well-known mechanisms can be employed, including code authentication, message encryption and authentication, and tamper proofing (cf. [3]). However, the specific constraints of sensor networks may be prohibitive to lifting them to a security level as high as traditional computer systems. Instead, we must explore solutions that exploit inherent characteristics of sensor networks, such as the large number of nodes, redundant deployment, and sensoric input, to provide adequate security guarantees.

In this chapter, we lay the foundations for such approaches by analyzing the vulnerabilities of WSNs. We concentrate on a type of attack on wireless sensor networks, which is both specific to and relevant in this domain. The security mechanisms presented in following chapters will be especially effective with regard to this threat model. This model assumes an adversary who is able, at a certain cost, to capture and take control of individual sensor nodes. This enables the adversary to exploit the capabilities of the captured nodes and participate in the operation of the sensor network. Depending on the objectives of the adversary, this may lead to data corruption, denial of service, or data disclosure, if no countermeasures are utilized.

The strength of an attack is not only determined by the number of cap-

tured nodes, but also by the attacker's ability to interfere with the operations of uncompromised nodes. Such interference may comprise the manipulation of sensor readings, which in most cases requires physical access to the nodes or the ability to change their environment, which may involve the use of powerful equipment. Usually it is easier to tamper with the operations of uncompromised nodes from within the network. Since sensor nodes act as message forwarders on behalf of other nodes, it is easy for them to drop messages or manipulate their contents. Thus, the selection of compromised nodes will be important for determining the strength of an attack as some nodes are more valuable to an attacker than others.

The best known protection against attacks on communications are end-to-end security associations such as shared keys or those provided by a public key infrastructure. However, such associations can be prohibitively costly in wireless sensor networks. Therefore, we explore alternative approaches that provide approximations to end-to-end security while still achieving an adequate level of protection.

3.1 Attack Paths

Today's research prototypes of sensor nodes employ virtually no protection mechanisms at all. Re-programming is easily possible with little technical requirements [15]. This is, of course, due to the early technical stage of development, and the research focus on functional and operational aspects of sensor networking. Also, the real-world deployment of sensor networks is in its infancy, so the demand for secure sensor devices has not yet emerged. Should the demand arise, it is probably possible to develop devices with a certain level of hardware protection, profiting from experience in other areas. With appropriate financial investment, it seems perfectly feasible to build sensor devices that are physically isolated from their environment except for dedicated communication interfaces. These interfaces remain to be secured through cryptographic means and secure protocol engineering.

There are three levels on which vulnerabilities may exist despite of careful design and engineering. The first is the physical level, i.e. vulnerabilities of the hardware. These are hard to protect against, since sensor networks are often deployed in open environments, and reliable physical protection is expensive. The second level is comprised of the interfaces that are offered by a sensor node. Some interfaces are indispensable, since communication with other nodes is required, and connections to sensors have to exist. The third level is the software running on sensor nodes. Secure software engineering is a quickly developing

field (cf. [179]), but eliminating vulnerabilities on this level requires stringent development practices, which often conflict with tight schedules and resource limitations. Therefore, is is likely that potential vulnerabilities remain in every wireless sensor network system. In the following, we discuss their relevance.

3.1.1 Physical Attacks

In a physical attack, the attacker gains direct access to the computing device hardware. This makes a denial-of-service attack easily possible: the attacker can simply destroy the sensor nodes. Physical access also allows him to access a node's components without any software layer involved. This is in contrast to a remote attack, where the attacked computer is accessed through some protocol or application layer, which gives it the possibility (at least, in principle) to detect the attack and react accordingly. In a physical attack, this sort of "self-surveillance" is not available to the device under attack and would only be possible by additional measures, such as external surveillance.

This makes physical attacks extremely powerful. They have a number of potential advantages over remote attacks:

- The attacker has (almost) certain knowledge about *which* device he is actually attacking. Network traffic, which is the medium for remote attacks, can be misdirected easily, and verifying the identity of a remote entity is hard (cf. honey-pots, which are dedicated computing environments that are designed to attract attacks for the purpose of identifying them and creating countermeasures [145]). Physical attacks happen with direct access to the computer equipment, which usually gives enough information for reliably identifying the equipment itself and its owner. Once the attacker has gotten so close, it might be impossible to divert his efforts to a less sensitive target.

- Network traffic is often secured by cryptographic means, for example by employing SSL. This makes eavesdropping or message injection practically impossible. On computers, data can be stored in encrypted form as well, but this is often refrained from due to usability and availability issues (e.g. the danger of lost keys). Therefore, physical access to a computer system usually yields full access to the stored data herein, including the ability for manipulations.

- The closer one gets to a computer system, the higher becomes the available bandwidth. Remote attacks are constrained by network interfaces. A long-distance connection typically yields between 64 kbit/s and 2Mbit/s

(ISDN, ADSL). Wireless connections usually yield between 128 kbit/s (ZigBee) and 54 Mbit/s (IEEE 802.11g). An attack occurring within a LAN yields up to 1 Gbit/s. Direct wired interfaces allow similar data rates, for example Firewire (the IEEE 1394b standard yields up to 800Mbit/s) or serial ATA (300 Mbyte/s = 2.4 Gbit/s).

- Sensitive information, which would not be accessible otherwise, can be acquired through special equipment that is secretly attached to a computer, e.g. a key logger for recording passwords.
- Physical evidence can be collected during an attack, which is not possible with remote attacks. Physical evidence could support non-repudiable attribution of data to a person or organization, thereby facilitating extortion. Examples of such evidence include hard disks, possibly with fingerprints on them, or printouts (that can be attributed to a certain printer).

On the other hand, physical attacks are usually riskier than remote attacks, since the attacker himself enters the domain of his opponent. Some risks are:

- Leaving traces that could lead to the identification of the attacker.
- Physical effort is required to break into the area where the computers are kept (e.g. a server room), which is susceptible to detection my surveillance mechanisms.

Physically attacking a sensor network avoids most of the risks usually associated with physical attacks. Sensor nodes are usually placed outside the close domain of their owners, for example in public spaces. Surveillance systems may be hard to operate in such areas. Once the attacker has physical access to the sensor nodes, it is easy for him to extract information from them if no further precautions are taken.

One possible measure is *tamper-proofing*. Here, sensor nodes are shielded by a barrier that is hard to penetrate and thereby prevents direct access to memory or the CPU. Similarly to smart cards or trusted platform modules (TPM), the core computational unit concerned with the handling of secret keys could be made *tamper-resistant*. This may deter an occasional attacker, but a determined and resourceful attacker is likely to break any existing shielding or scrambling mechanism as research in smart cards and other hardware platforms (such as the X-Box gaming console) has shown [6, 96, 84].

Active countermeasures can further raise the bar for the attacker, for example by incorporating means for detecting a physical breach, temperature extremes, voltage variation, and radiation, which is common in high-end security

modules [63]. As soon as a potentially threatening event is detected, the memory holding secret keys is zeroed. Such measures are costly, and an acceptable trade-off must be found that takes the actual risk of such attacks into account. Sensor devices, which have to be available in large quantities at low cost, are unlikely to incorporate such means. However, a certain level of self-protection may be possible. The sensors that are already attached to a sensor device may be useful for detecting certain events, for example sudden movements, which may be sufficient for many practical applications.

If tamper resistance is considered too costly, at least some level of *tamper evidence* may be provided. Upon inspection, this would make the fact that an attack has occurred obvious. Natural characteristics of the deployment area may also support the protection of a sensor network. For example, the terrain where the nodes are placed may be inaccessible, or sensor nodes may be concealed between other objects, making them harder to detect. All these measures lead to a certain level of *tamper resilience*, which increases the cost for a successful attack, for example by delaying the attacker or requiring him to acquire specially crafted equipment.

The risk of a physical attack depends on the environment and the context in which the sensor network is deployed. Questions to consider in order to assess the risks are: Who would be interested in disabling the network? Where and when is the network deployed, and how high is the exposure to potential attackers? What is the potential impact of a disabled or manipulated sensor network? In many cases, one might be satisfied with the risk being reduced by inherent properties of sensor networks, i.e. the small size and high redundancy of sensor devices.

3.1.2 Interface Attacks

Interface attacks exploit vulnerabilities of the interfaces a device provides in order to allow access to its own services or to access external services. For wireless communication interfaces, there are obvious attacks such as eavesdropping, jamming, traffic analysis, and message injection among others. They are facilitated by the broadcast nature of wireless communication, and the fact that access is easily possible without the risk of detection. An overview can be found, e.g., in [129]. Interface attacks can also be executed on the level of a service API, for example those of security processors [28]. Here, valid commands are executed in unusual sequence, thereby provoking unintended behaviour in favour of the attacker. To our knowledge, the service (message) interfaces of sensor networks have not been investigated with regard to security vulnerabili-

ties. Instead, most work has been done to secure the wireless interface.

Attacks on the wireless interface of sensor nodes are easy to execute as they require only a wireless transceiver. Either an external device could be used, or captured nodes of the sensor network itself, after a successful physical attack on some of the nodes. Here, the difference is in the coverage of the deployment area: a high-powered external device may enable the attacker to reach all nodes at the same time, while single sensor nodes have a much more limited radio range.

Some attacks on the transport layer can be thwarted easily. Link-level encryption is already sufficient to prevent eavesdropping, for example. Message injection through an external device (without knowledge of keys) is also thereby prevented. A possible (costly) countermeasure against traffic analysis is high message redundancy. Other attacks are almost impossible to prevent, such as jamming. Some mitigation techniques are applicable, though. If only a limited region is affected, it may be possible to route around it. In hybrid networks [166], which employ additional wired connections, a jammed node could raise an alert outside the jammed region. A possibility for preventing jamming would be the use of directed optical instead of radio links, but those are much harder to deploy.

The risk of an attack occurring on the wireless communication of a sensor network is quite high, since it is relatively easy to mount. The impact of such an attack can be mitigated by measures such as message encryption and authentication, and by reporting jamming attacks. Experience teaches that careful design and implementation of cryptographic mechanisms is necessary to ensure that the security goals are achieved. The vulnerabilities of link layer encryption in the IEEE 802.11 standard is a popular example [30]. Much of the research work on the security in sensor networks, as described in the previous chapter, is concerned with the design of such mechanisms that are suitable for sensor networks.

3.1.3 Software-Level Attacks

A powerful attack is the injection of code into an execution environment, since this yields potentially full control over this environment. Such attacks are common in the Internet world, where poorly administrated hosts are susceptible to adversarial remote control. One of the reasons for this is code mobility, i.e. code is often downloaded from remote sites and locally executed. Even if mechanisms for code certification exist, these are often circumvented by social engineering or user inattentiveness. Sensor networks are comparatively more

closed environments, but code updating is a common feature and introduces similar vulnerabilities.

Software for wireless sensor networks is often developed using low-level programming languages like C. This facilitates the introduction of vulnerabilities such as buffer overflows [109]. Fortunately, microcontrollers (which are the basis for sensor nodes) are often based on the Harvard computer architecture, which physically separates program and data memory. In such an architecture, buffer overflows usually don't lead to unwanted program execution, since most programs don't write into program memory directly. However, moving to processors that are based on the von Neumann architecture, or using virtual machines (such as Maté [114]), exposes sensor networks to the risks of such vulnerabilities.

The attractiveness (to an attacker) of software-level attacks lies in the fact that such attacks are "class-type" attacks, which means that once it is known how an attack can be successfully mounted, this attack can be applied over and over again with minimal additional cost since all systems of the same class are vulnerable to it. This means on one hand that if a software vulnerability can be exploited on one sensor node, all other nodes in the same network are most likely also subject to this attack. On the other hand, this can mean that if such an attack can be successfully applied in one sensor network, other networks that are built from the same underlying platform and system software may also be affected.

Custom software development can reduce the risk of software-level attacks, since the exploitation of vulnerabilities in such systems is more costly to an attacker than in standardized systems. Also, the absence of software lifecycle management mechanisms allows it to build such restricted interfaces that further reduce the risk of vulnerabilities. However, both approaches put harsh restrictions on the flexibility and the cost-effectiveness of such systems. It can therefore be safely assumed that a more open approach will be usually used in sensor networks in the future.

A mechanism for code updates is multi-hop over the air programming [172], where new software versions are distributed to all nodes in a network in a cooperative manner. A risk in this approach is that code updates are injected by an attacker who might thereby be able to exploit the inherent update mechanism of the network for gaining control over all nodes in the network. It must be noted that even if the update mechanism is protected cryptographically, it may be possible for an attacker to learn the required keys through out-of-band mechanisms such as "social engineering" (e.g. blackmail or bribery).

Virtual machines [114] execute programs that are encoded as ordinary data,

from the point of view of the microcontroller. This circumvents the inherent separation of code and data found in such platforms based on the Harvard architecture. In principle, this makes it possible to inject code through the manipulation of user data. Code being executed by a virtual machine can be restricted to certain data in a "sandbox", which limits the reach of malicious code. Nevertheless, the behaviour of a sensor node could be altered in a way that affects the overall result of the sensor network's operation.

A mechanism to prevent unauthorized code from being executed is remote software attestation [164]. This enables the verification of code running on remote devices, without having physical access to their memory. Sensor nodes are a good candidate for this technique, since they are operated within a single administrative domain, and their software configuration is known at any time. Of course, the verification of every single node in a large network does not scale economically, and even the verification of a single node could be expensive if done over multiple hops. However, this technique might be sufficient to deter some of the most harmful class-type attacks.

3.2 Attack Objectives

In general, one can differentiate between *primary* and *secondary* objectives that an attacker pursues. The primary objectives concern the informational resources the attacker wants to gain control of. His goal may be to acquire some secret information, or disrupt a service, or falsify some data in order to hide the the presence of facts, just to mention some examples. The secondary objectives are concerned with the circumstances of an attack. For example, the attack might have to be carried out within a certain time frame, or it might be crucial that the attack is not detected. In this section, we discuss possible objectives when attacking a sensor network.

3.2.1 Properties of Resources

In a sensor network, the valuable informational resources that require protection are the sensoric input, the aggregated data stored in the nodes, and the exchanged messages. Several aspects to these resources are important and may be subject to an attack:

- Confidentiality
- Integrity
- Availability

- Timeliness
- Origin

The first three of these points refer to the "classical" security properties that have to be protected in virtually any security sensitive application. We consider timeliness as an additional essential security goal in sensor networks. It may be possible to subsume this goal under availability, but in many applications, timing is crucial and seemingly insignificant delays could have severe impact.

When referring to the *origin* of some piece of data (or a message), we assume that a party obtaining the data (for example, by reading from a sensor or receiving a message) also obtains some statement about the association of the data with its origin, i.e. the source from which it has been obtained. This statement could be backed up by a digital signature, or it might be implicit as when reading from a sensor. If the evidence supporting the statement is sufficiently strong, the party may decide to attests its finding, thus further supporting the statement. It is therefore crucial that a statement about the origin of a piece of data cannot be forged or tampered with by an attacker.

Usually, the origin of a message refers to the entity that has generated the message. Origin authentication is based on some feature of the source, such as a public/private key pair, a common secret key, location, or a biometric attribute (in real life, voice is often used).

Integrity and origin authentication are often achieved through the same mechanisms. An important mechanism is the concept of message authentication code (MAC). Such a MAC provides a means for the receiver of a message to verify the message's origin and its integrity at the same time. Both concepts are closely connected, as when the integrity of a message is violated, it is essentially transformed into a different message, which has a different origin as well. Vice versa, if the origin of a message cannot be verified, it does not necessarily follow that the integrity of the message has been violated. Stated differently, even if we don't know where a message comes from, it could still be a valid message. Thus, it is valid to say that origin authentication implies integrity, but not the other way around.

3.2.2 Resource Types

It very much depends on the level of abstraction what components of a networked computer system are regarded as valuable resources that require protection. In a transaction-oriented system, the database would be the most valuable resource. On a home desktop PC, personal information such as credit card

details, passwords (saved in files) or e-mail accounts, are subject to attacks from the Internet, and therefore need protection. In process automation control systems, the physical actors could do severe damage, and must therefore be isolated from external manipulation. For a web service (such as Amazon, which has been a victim of denial-of-service attacks), the available bandwidth is crucial for business.

The most valuable resources, as regarded from a general, application-neutral point of view, are described in the following.

Sensor input The origin and timing of sensoric input are immediately available to a sensor node as the sensors are directly attached to the node and the sampling is controlled by the node itself. The attacker might replace the sensor attached to a node with some device that pretends to be an ordinary sensor but acquires its input from some distant place, or simply delays its output. It is not possible for a node to distinguish such a device from a real sensor. From the node's point of view, this attack is equivalent to a simulated environment.

The availability and integrity of sensor data can be affected without directly manipulating the hardware of the node. The environment, from which the sensor acquires its input, could be deliberately changed. For example, if the purpose of the sensor node is to measure the level of brightness, an artificial light source may provide the node with false data. Thus, access to the real data is denied. Some sensors may be dependent on certain environmental conditions in order to function correctly. For example, the accuracy of a sensor may deteriorate if the temperature exceeds certain bounds. An attacker might be able to exploit this by placing a heat source near the sensor node.

Confidentiality of sensor data is usually not a great concern if the monitored area is publicly accessible. Even in restricted areas, the data acquired by single nodes may not be meaningful as they reflect only punctual measurements. On the other hand, if acquiring the data is associated with significant costs, for example if an expensive sensor is required, the collected data is an asset in itself and it might be necessary to protect it against unauthorized read access.

Aggregated data The value of pure sensor data depends on contextual information (i.e. the circumstances under which the data has been acquired) and often gains significance only in combination with the findings of other sensor nodes, or when it is aggregated over longer periods of time. Such aggregated data contains sufficient information to be interpreted by the operator in a meaningful way.

Critical decisions are often based on aggregated data, it is therefore essential that the data is "correct" in the sense that even if some input on which the aggregation is based is corrupt, the deviation from the "true" result (the result that is obtained when no maliciously or otherwise induced errors are involved) is minimal. A viable attack goal is therefore the manipulation of data such that a false report is being accepted by the issuer of a query. Without any precautions, even an attacker restricted to manipulating very few input values could significantly influence the outcome of an aggregation. Consequently, techniques for detecting or at least mitigating the effects of faulty input are required.

Perfect resilience against manipulated input data is impossible to achieve if the manipulated parts cannot be identified (which is usually the case). The secure aggregations schemes by Wagner [186] and Przydatek et al. [146] therefore aim at *approximate integrity*, where the result y^* of an aggregation with partially corrupt input deviates from the result y that would have been obtained in absence of an attacker only by a small value ε, i.e. $|y^* - y| < \varepsilon$. We note that some slight deviation from the "true" value must be dealt with even when no attacker is active, since sensor data is inherently subject to noise.

Since some cost is involved in aggregation, aggregated data should be considered more valuable than raw sensor data. Therefore, it is often appropriate to restrict read access to it and allow only authorized parties to obtain such data even if access to raw sensor data is unrestricted.

The availability of aggregated data is endangered when the aggregating node is compromised by the attacker. This would allow the attacker to delay or suppress a report, at least temporarily. As "aggregator" is likely to be implemented as a *role* in a sensor network, there is nothing to prevent another node from assuming this role. If the answer to a query is not delivered in time, a role switch could be triggered, the new aggregating node would repeat the aggregation process and finally deliver the report. The suppression of data has the disadvantage that other nodes can detect the malfunctioning, which could be used by an intrusion detection system to mark the respective node as being "suspect" and eventually isolate them.

The "origin" of aggregated, higher-level data is determined by the origin of the raw data that serves as input to the aggregation process. Thus, the contextual information that is attributed to the raw data, such as location and time, may be maintained in the aggregated data. However, the origin of aggregated (i.e., processed) data becomes blurry and often such information will be discarded for efficiency reasons.

Messages Wireless radio communication is based on a shared medium, which makes it easy for a passive attacker to overhear the traffic in a network. Sometimes, directed radio links or optical communication make eavesdropping harder, but a determined attacker is likely to be able to record all traffic. Ultimately, the confidentiality of messages can only be preserved through encryption. Setting up keys for securing point-to-point links is a well-understood problem. In sensor networks, key agreement schemes as described in chapter 4 can be used. This renders pure passive attacks more or less useless. Still, it might be possible to extract useful information from the traffic patterns that occur in the network.

Of course, link-level encryption is not able to effectively hide the content of messages from an active attacker. All messages received by compromised nodes can be read by the attacker. In a strong sense, the only way to preserve confidentiality is end-to-end encryption. The same applies to message authentication (and thus integrity): only if there is a common cryptographic context between the sender and the receiver, the origin of a message can be verified in a strong sense. However, due to the resource restrictions that apply to sensor networks, end-to-end security may not be a practical approach. Other approaches that approximate the properties of end-to-end security are the main topic of this thesis.

In a large sensor network, messages often have to be transmitted over several links until they arrive at their destination. Generally, we can not assume a common cryptographic context between the sender and the destination. It is therefore possible that a compromised node changes the contents of a message it is relaying. Such changes could be detected by nodes that overhear the incoming and outgoing messages. However, these potential guards may be asleep at that time. Link-level encryption makes overhearing ineffective as well. We propose interleaved authentication (see chapter 6) to this end.

The injection of fabricated messages is another way of manipulating the operation of a sensor network. If a node emits messages in its own name, it may not be possible for other nodes to decide whether these messages are the product of correct operation, or if they are forged. Certain intrusion detection techniques may be able to isolate such nodes if their behaviour deviates significantly from ordinary operation. Alternatively, nodes may forge messages and attach a different ID as their origin to them. This attack is commonly called *spoofing*. If the used ID does not exist, this behaviour is subject to detection if the ID is challenged. If the ID exists and another, non-compromised node with the same ID exists in the network, it may raise an alarm if it detects a message that was sent by the malicious node.

Actors A complementary extension to sensor networks is the ability to act upon their environment. For this, some device is required, such as a capsule that can release a substance, or an electric motor that can open and close a door. These devices are called *actors* and they are controlled by well-defined – usually electric – signals.

One line of attack against an actor, which is attached to a sensor node, is to take control of the sensor, which controls the actor. This would give the attacker at least the level of control that the sensor had. Additionally, it may give the attacker the opportunity to transmit his own signals to the actor, leading to unpleasant consequences.

The complex interactions between the software and hardware of the sensor node and the actor device, and the communication protocols, could lead to vulnerabilities. By sending commands in a certain, perfectly legal, order, it may be possible to trigger a certain sequence of actions that were not foreseen by the system's designers. Such vulnerabilities are common software errors in complex systems.

3.2.3 Detection Evasion

The effectiveness of many attacks on a computer system depends on the fact that the presence of an attacker is not detected. For example, if an attacker wants to read sensitive data his victim's machine, he has to make sure that the victim does not learn about the presence of the attacker. Otherwise, if the victim learns that an attacker has access to sensitive data on a specific machine, he would stop using that machine, which renders the attack ineffective. Of course, for some types of attacks, it is unavoidable that they will be detected sooner or later. This includes, for example, denial of service attacks or the destruction of nodes.

Analogously, as soon as the operator of a sensor network learns that certain nodes have been compromised, he stops accepting data from these nodes. Other nodes will not forward messages from those nodes any more, and avoid them for routing own messages. Their keys are invalidated, and they lose their access rights to information within the network. Thus, they become useless to the attacker.

Considering the fact that compromising nodes requires a significant investment, it is in the interest of the attacker to avoid that his activities are detected. Usually, he wants to be able to use the nodes under his control for as long as possible in order to amortize the costs that were necessary for taking control over the nodes. The attacker will therefore adapt the behaviour of the compro-

mised nodes according to the intrusion detection mechanisms that are utilized by the network.

3.3 Adversary Characteristics

In section 3.1, we have already discussed several types of attacks on sensor networks and how they can be defended against. We have seen that there are efficient and effective mechanisms against outsider attacks, which allows us to largely exclude such attacks from further consideration. The major threat to sensor networks thus comes from the inside, i.e. legitimate nodes that have been compromised by the adversary and are now operating under his control. This section discusses the potential and the limitations of such attacks.

3.3.1 Basic Assumptions

We assume that compromised nodes can exchange messages among themselves without being noticed by the legitimate rest of the network. In practice, this may not be easily possible, but it is prudent to assume that a sophisticated attacker will find ways for compromised nodes to collaborate in secret. There are several ways in which this may be possible. For example, the adversary could install his own base station through which the compromised nodes communicate, or such traffic may be encrypted and encapsulated in ordinary messages.

A second assumption we make is that of a "practical" attacker, i.e. one that has only limited resources. Additionally, the attacker does only have insider knowledge about the WSN that is obtained during the course of an attack.

3.3.2 Attack Costs

The cost for a successful attack on a sensor network is the sum of the time and the expenditure that must be spent to take control of a sufficiently large number of nodes. This cost may almost be zero, for example due to a software glitch that can be easily exploited. In order to make a WSN secure, the cost should at least be linear in the number of nodes or even higher, for example by installing surveillance means that make detecting an attack more likely if the attack is taking more time.

It is likely that a WSN offers the capability of reprogramming sensor nodes after deployment. This functionality should only be available to the legitimate operator of the network and must be secured appropriately, for example through access keys. However, due to a software glitch, or by exposing the required

access keys, this capability could be available to the adversary as well. In such a case, the attacker may be able to impersonate the operator and most other security measures would be rendered ineffective. The adversary's cost for executing such an attack would be close to zero. We will further disregard this type of attacks as they are beyond the scope of this work.

We assume that breaking into nodes and taking control over them is a non-trivial task. Although with current prototypes, the barriers are rather low [15], it is valid to assume that, should the need arise, more resilient sensor nodes can and will be built. An adversary will therefore have to make some investment for each sensor node he attacks. This investment is composed of the cost for the required equipment, the effort of finding and accessing a node, and the time required for this process. Thus, an attack on a well-designed WSN requires effort that is at least linear in the number of nodes, assuming that the cost for attacking a single sensor node has a lower bound. Due to this assumption and the assumption of limited resources, an attacker will be able to take over only a limited number of nodes. A network that exceeds this size will have legitimate, working nodes left even after a successful attack.

3.3.3 Avoiding Intrusion Detection

We assume that a mechanism exists that can detect dropped messages. Many communications between nodes may include only a single message. We assume that the transport layer either delivers a message or notifies the sender when delivery is not possible. However, we do not assume a secure transport layer that ensures the delivery of messages. Instead, we assume a mechanism that will detect, possibly at a later time, if a message has not been delivered and no notification has been received by the sender.

3.3.4 Insider vs. Outsider

An important distinction of attacker classes is based on their knowledge of cryptographic keys.

An insider has access to keys, either on the API level or in cleartext (in strong security designs, the access level is usually restricted to the "needed" operations). To be conservative, we must assume that such an attacker can perform any cryptographic operation he likes using the keys. How he obtains access to these keys is not important. Bribery, extortion, break-in, any form of deception or "social engineering" may be successful. Ultimately, with sufficient resources, keys can be extracted from the physical devices that make up the network nodes.

In contrast, outsiders have to rely on externally observable information only. Eavesdropping may be as little as information on the message traffic, if messages are encrypted. Sometimes, cryptographic knowledge can be acquired through observant means only (cf. the attack on early WLAN encryption [175]).

3.3.5 Technical Capabilities

Another characterising feature are the capabilities of the devices that are used by the attacker. A "mote-class" (terminology from [90]) attacker has the capability to participate in the network on the same level as the sensor nodes. He achieves this by either placing his own nodes, or by seizing sensor nodes and taking control of them. In the latter case, we usually have to assume that he has access to secret keys at least on API-level, i.e. he can sign messages, generate authentication codes, etc.

A "laptop-class" adversary operates externally to the sensor network. By using powerful equipment, he is able to cover a large area and is able to communicate with several nodes at the same time. He can overhear communication, intercept messages (i.e., read them and at the same time prevent their delivery within the sensor network), inject messages, use shortcut routes (corresponding to a wormhole attack, e.g., by using a wired connection), or jam the network.

In our model, the attacker relies only on the capabilities of the compromised sensor nodes. This corresponds to a "mote-class" attacker. As described above, we assume an attacker that tries to avoid detection. The use of powerful equipment is therefore highly constrained.

3.3.6 Location-Constrained Attacks

We assume that it is not possible for an adversary to inject own nodes into the network, due to the cryptographic measures utilized. In order to participate in the operation of the network, it is necessary that the adversary takes control of some existing nodes. This allows him to read all message passing through these nodes and to inject messages. It also allows the attacker to make use of the keys of the nodes he controls, possibly even retrieving the key material. Clearly, the more nodes the adversary takes control of, the greater is the impact he has on the operation of the overall network.

The distribution of compromised nodes greatly affects the effectiveness of an attack. When the attacker controls nodes that are randomly spread across the entire network area, he cannot prevent data reports from any area, but he can influence the reports from all areas. The error in all reports is thus potentially increased. On the other hand, if the attacker controls only nodes that are

concentrated in a certain area, he fully controls all reports from that area, but he has no influence on the reports from other areas.

In general, the effectiveness of an attack highly varies with the geographical distribution of the compromised nodes. We describe this in the following through the *principle of locality*. We then go on discussing the effects of several distributions of compromised nodes.

Principle of Locality

The influence on reports, which a compromised node can exercise, highly depends on the location of the node. The most powerful position is at the source of the report. If the compromised node itself creates a report, or a significant part of it, it can make up the report with arbitrarily generated data. The power of other nodes to verify such a report are limited and depend on application semantics (e.g., to check the plausibility of reported data) and sensor range. Similarly, if the receiver of a report is compromised, it may deliver arbitrary data to the querying entity.

The second most powerful location where a node can exercise influence on reports is either close to the receiver or close to the sender of a report. Here, the probability that messages travel through a compromised node is high and thus the node might have the opportunity to change the contents of a report by manipulating these messages.

The threat of compromised nodes being located close to the receiver or being the receiver itself can be overcome by obtaining a report in a redundant manner, thus multiple receivers are established, which makes manipulations more difficult. In most cases, redundancy cannot be applied to the source of a report as easily.

Locations that are far away from either the sender or the receiver have a much lower probability of influencing a report through message manipulation as the likeliness that messages travel through specific nodes is low in densely populated networks. Only if few alternate routes exist, nodes become bottlenecks and draw traffic to them. This may be exploited by an attacker through simulating congestion in certain areas, which may trigger the rerouting of messages. The use of multiple paths that are spatially separated (discussed later in Chapter 5) is a possible means to mitigate such threats.

It can be expected that the distribution of compromised nodes determines how an attacker can influence the operations of a wireless sensor network. In the following, we discuss some fundamental distribution patterns.

Random Spread Distribution

This model assumes that the adversary picks arbitrary nodes randomly from the network and takes control over them. This is probably a very unrealistic model if applied to an already deployed network. It requires that the adversary breaks into a single sensor node and then randomly moves to another node. If there are measures to track such movements, this attack bears a high risk of detection. Considering the large number of nodes in a sensor network, a single node is only of very limited value to the attacker. This value is probably exceeded by the cost of moving from one node to the next, which makes the attack uneconomical.

However, the following scenario may be more realistic. Assume that the adversary gains access to a set of nodes before deployment and manages to replace the program code on these nodes with his own. The nodes will then be randomly deployed on the network area, and the adversary ends up with a number of randomly distributed nodes he has control over.

The advantage of this attack is that the adversary has access to nodes distributed throughout the whole network area, which allows him to monitor a large portion of the message traffic with relatively few nodes. In this regard, the attack is efficient for eavesdropping and monitoring purposes. However, active attacks are not very effective, as the amount of data that can be injected by a few randomly distributed nodes is small compared to the total amount of data in the network. Also, a single compromised node surrounded by legitimate nodes is more likely to be expelled when showing abnormal behaviour or reporting data with a high divergence from its neighbours.

Concentrated Distribution

When an already deployed sensor network is being attacked, the attacker might start at a certain position and try to subvert as many nodes around that position as possible. This would allow him to control all message traffic that is going into or out of that area. The cost for moving around (and the involved risk of detection) is amortized over a larger number of compromised nodes, thus this attack mode is more efficient than randomly moving around and picking out single nodes.

As a simple formal model of this type of attacks, we assume a starting position and a function f that describes the probability with which a node in a certain distance from the starting position is being compromised. At the starting position itself, this probability is equal to a success probability p_0 with $0 \leq p_0 \leq 1$, i.e. $f(0) = p_0$. With increasing distance from the center, this

probability is likely to decrease, while the exact progression of f depends on the adversary's capabilities. As a very simple approximation, we consider a maximum radius r and a linearly decreasing success probability:

$$f(d) = p_0 \left(1 - \frac{d}{r}\right)$$

for $0 \leq d \leq r$ and $f(d) = 0$ for $d > r$. This means that up to a distance r, the success probability of the attacker decreases linearly. Beyond that distance, the attacker is inactive.

We may consider different f-functions applied at the starting point. These could vary the radius r, or the success probability might depend on the direction from the starting position or other parameters such as the environmental conditions of the deployment area.

Hitpoints Distribution

We can use multiple starting points to model an adversary that becomes active in several locations. A number of these "hitpoints" throughout the network are selected according to a certain (random) distribution. The nodes in the hitpoint areas could be targeted sequentially or in parallel. When we refer to this attack type, we use the same f-function for all hitpoints.

Partitioning Distribution

This attack mode has the goal of partitioning the network, leading to control over the message flow between the parts of the network. This is achieved by subverting nodes along a path that partitions the network. Depending on the topology of the network and the objectives of the adversary, certain areas are more vulnerable to this attack. A "bottleneck" in the topology of the network would provide a good location for mounting such an attack, as the number of nodes required for a partition is very small. In most cases, the attack path is probably determined by the objectives of the adversary. For example, the adversary may attempt to separate a certain area from the rest of the network in order to be able to perform certain actions in this area undetected.

If the attacker blocks the message flow out of some part of the network completely, this may be detected due to the lack of reports from that area. However, it may give the attacker enough time to perform his activities. When the block is canceled afterwards, new reports from that area are again unsuspicious. However, if there are end-to-end security mechanisms between some nodes within the blocked area and outside of it, this attack may still be detectable. We will present such a mechanism in chapter 6.

3.4 The Cost of End-to-End Security

By end-to-end security, it is usually understood that two principals communicate with each other in such a way that no outsider can interfere with this conversation. Two problems have to be addressed in order to achieve this:

1. It has to be determined *who* the communicating parties should be. In particular, if one party wants to establish a connection to a service, the address of the service provider has to be determined in a way that ensures that a correct and honest party is chosen.

2. It must not be possible for anyone except the legitimately participating principals to read, overwrite, or delete messages that are being exchanged, nor insert new messages.

To address the first problem, either an out-of-band mechanism is employed, such as manually entering the address of a web service, or some service directory is consulted, for which its address must also be known in advance. There is always the issue of bootstrapping this process, but usually a practical approach can be found.

The second problem is especially prevalent in open networks such as the Internet, where communication partners are often unknown, and wireless networks, where anybody can potentially interfere with the communication links. Therefore, additional protocol layers are deployed that provide end-to-end security guarantees based on an insecure network. In the Internet, examples of such layers are the Secure Shell (SSH) [200] and Secure Socket Layer/Transport Layer Security (SSL/TLS) [53] protocols.

When using SSL connections, packets are authenticated, encrypted, and serialized. This ensures that packets cannot be inserted or replaced, their cleartext cannot be accessed, and replayed or deleted packets can be detected. Such connections are used for transmitting sensitive data, such as passwords or bank account information. Another popular area where end-to-end security is desirable is telephony over the Internet (Voice over IP). Concerns about the security of that technology have led to the development of protocols that support the encryption and authentication of voice traffic between the communication endpoints (e.g. the Secure Real Time Protocol, see [99]). Deleted packets are usually noticed by the participants on the semantic level (here, by voice interruptions).

The mechanisms to address the above two problems require a certain effort, and this section investigates, which amount of resources must be devoted to their implementation in the context of wireless sensor networks.

3.4.1 Connection Establishment

Before an end-to-end connection between two nodes can be set up, the initiator must sort out *which* node should be on the other end. In a traditional network setting, this is often done through external means such as a service directory, from which the address of a service provider is obtained. Each change of the current service provider requires an update of the service directory. If such changes occur frequently, all service requests must be preceded by a new look-up in the directory in order to retrieve the current provider address. The Domain Name System in the Internet is an example of such a system. However, since changes in this system occur infrequently, most of the information can be cached in a hierarchy of directory servers.

In a WSN, it can be expected that the actual node providing a service changes quite often due to the dynamic nature of phenomena being observed by the network, and due to load balancing and failure recovery mechanisms. This transient existence of service providers potentially facilitates a form of man-in-the-middle attacks, i.e. malicious nodes posing as service providers and intercepting messages targeted at a legitimate service provider. The use and maintenance of a central service directory would be helpful, but also very costly due to frequent directory updates and look-ups. Additionally, a central component is a security risk since it would be a worthwhile target for an attacker.

Assuming that some mechanism for obtaining the identity and the address of a service provider exists, there is still the need to establish a secure connection between the initiator (client) and the provider (server), i.e. to engage in a key agreement protocol. The standard protocol for doing this on the Internet is SSL. There are three messages being exchanged before application data is being sent: one “hello” message for initiating the connection, and two messages whereby client and server exchange certificates and key information. This protocol can be simplified to a two-step version if certain parameters, which are usually communicated in the first message exchange, are fixed in advance. Thus, establishing a SSL connection requires at least one message in each direction.

In the context of the Internet, these two messages induce a negligible overhead. They use only a small fraction of the available bandwidth and the induced delay is insignificant compared to the duration of the following session. The key exchange protocol is based on public-key cryptography, which requires a significant amount of computational power. However, the involved computational overhead is an easy task for modern processors used in PDAs, workstations, and servers.

In a sensor network, however, the overhead induced by connection setup

Algorithm	Signature		Source
	Verification (public key op.)	Generation (private key op.)	
RSA 512 (16 MHz)	0.70 s	5.0 s	[127]
RSA 1024 (16 MHz)	2.30 s	33.9 s	[127]
RSA 2048 (16 MHz)	8.40 s	4 min 7.6 s	[127]
RSA 1024 (8 MHz)	0.43 s	10.99 s	[74]
RSA 2048 (8 MHz)	1.94 s	1 min 23.26 s	[74]
ECC secp160r1 (8 MHz)	0.81 s	n/a	[74]
ECC secp192r1 (8 MHz)	1.24 s	n/a	[74]
ECC secp224r1 (8 MHz)	2.19 s	n/a	[74]

Table 3.1: Execution time of public key operations on the BTnode platform

Algorithm			Source
	Sig. verification	Sig. generation	
RSA 2048	< 35 ms		[163]
ECC 192	40 ms	20 ms	[163]
	Block operation		
AES-128	11 μs		[163]
AES-256	13 μs		[163]

Table 3.2: Performance of cryptographic operations on a smartcard processor

may be significant. Messages can be required to travel over multiple hops, thus the burden for transmitting them is placed on multiple nodes. Also, there is a delay of a full round-trip time interval before the first application data message can be transmitted. This delay is further increased as cryptographic computations require a significant amount of time on sensor nodes. With only a small amount of application data to be transmitted, which is typical in sensor networks, such a security mechanism is very inefficient, since the overhead for setting up the connection significantly delays the transmission of data messages and cannot be amortized over many messages.

3.4.2 Public-Key Cryptography

Protocols like SSL rely on public-key cryptography, which is computationally very intensive. RSA operations for signature generation and verification take several orders of magnitude more time than symmetric key operations or hash functions. There exist alternative approaches, as those discussed in Chapter 4, which are based mainly on hash functions and require no public-key cryptography, so they perform much faster than RSA-based key exchange.

Table 3.1 illustrates the time consumption of RSA operations. These figures were empirically obtained with an implementation based on the PKCS

standard [104] executed on a BTnode (source: [127]). As public exponent, the value $F4_{(hex)}$ has been used. The numbers are averaged over 5 measurements performed for each key size. Gura et al. [74] report on the performance of an optimized implementation of RSA and an Elliptic Curve Cryptography (ECC) algorithm on the ATmega128 platform. For RSA, their implementation yields a 3- to 5-fold improved performance compared to the (more naive) implementation of [127]. However, the order of magnitude of these operations still prevents excessive usage of RSA. The ECC algorithm yields an improvement of one order of magnitude compared to RSA signature generation. This indicates that ECC seems to have quite some potential for sensor networks; a library for TinyOS is available [116].

For comparison, Table 2.2 shows figures for different cryptographic primitives, namely AES encryption and hashing. These numbers show that there is a vast difference between symmetric key cryptography and hashing, and public key cryptography. They clearly indicate that symmetric mechanisms have a significant advantage over public key cryptography from a performance perspective.

Another issue with RSA are the relatively large key lengths. At least the public key and the accompanying certificate (a signature) have to be exchanged for key agreement. For a key length of 1024 bit, this yields an additional overhead of 256 bytes in either direction. Compared to the typically very small size of data messages in a sensor network, this overhead is significant.

3.4.3 Pairwise Key Distribution

In a fully connected network of n nodes, each node maintains $n-1$ connections. With each connection, a data structure is associated that uses up some space, say m bytes. Therefore, each node has to store $m(n-1)$ bytes of state information. Given M bytes devoted to storing such kind of state information, the supported network size is determined by $n = \frac{M}{m} + 1$.

As an example, let's assume that with each node, a 32-bit (4 bytes) identifier (which may include a unique ID, location information, and possible other data) and a 128-bit (16 byte) key is associated. Thus, a node has to store $m = 20$ bytes for each other node in the network. If a sensor node provides $M = 100$ kbyte for security purposes, this allows a network size of $n = 5000$ nodes.

Currently, microcontrollers used in sensor node prototypes provide up to 512 Kbyte non-volatile data memory (Flash EEPROM), e.g. the MICA product line of Crossbow Technology Inc. Much of this space is used by code for the operating system, network stacks, and applications. We can assume that

large parts of the memory are used for storing application data such as sensor readings and aggregated data. In principle though, deploying more memory would be possible.

Full pairwise key distribution therefore seems to be a viable option for sensor networks. There are, however, limiting factors:

- A static distribution of keys is inflexible as it does not allow to add more nodes to the network later.
- Although a certain network size can be supported, the approach is nevertheless not scalable to larger networks, say in the order of magnitude of 10^6 or larger
- Sensor nodes should be as small as possible, thus the amount of memory that can be added is limited. Also, the operator of a sensor network probably prefers using the available memory for application purposes and is not willing to reserve a large part of the available resources for security purposes.
- Although keys are available for every pair of nodes, most of them will never be used, since a sensor node will interact with only a tiny fraction of the nodes in the network during its lifetime. Thus, most of the memory used for storing the keys is never used productively.

We conclude that up to a certain network size, full pairwise key distribution is feasible if one is willing to dedicate a significant amount of memory to storing keys. It is, however, not a generally applicable solution to the problem of secure communication. The deployment of additional nodes during the lifetime of the network is not well-supported. Most importantly, the approach does not make efficient use of the available resources. For the most attractive use cases of wireless sensor networks, where nodes are very small yet there is a large number of them, this approach does not work.

3.5 Approximating End-to-End Security

As elaborated in the previous section, End-to-end security mechanisms achieve two goals: (1) Secure matchmaking of communication partners and (2) secure message exchange. Essentially, they guarantee that dishonest parties cannot interefere with the communication of honest parties in any way.

One could relaxe these criteria to a certain degree and demand, for example, that, say, 90% of all matchmakings occur between legitimate parties; or, that

the adversary can only read messages but not interfere with them. By "approximation" to end-to-end security we understand, in an informal way, that there is an upper bound on the influence that malicious parties can exercise. This influence can be measured in several ways, for example as the fraction of messages in the whole system that are subject to manipulations, or the probability with which messages that are exchanged between a specific pair of nodes are subject to manipulations.

In this section, we first outline the threat model under which we will study the security mechanisms described in the following chapters. We sketch multipath communication as a generic model for these mechanisms, and conclude by defining standard measures for assessing their effectiveness.

3.5.1 Threat Model

We assume an adversary with the general objective of manipulating the outcome of the wireless sensor network under attack. As a secondary goal, the adversary tries to avoid detection. We assume that the adversary is restricted in his attack capabilities such that a node capture attack is the only feasible type of attack, and that capturing a node is associated with at least a certain fixed cost.

The strength of the adversary varies in two dimensions. First, the number of captured nodes is variable, and second, the adversary's mobility is geographically restricted.

The power of the adversary lies in his ability to make use of the captured nodes for manipulations on the messages being exchanged. Under an end-to-end security regime, his influence would be constrained to the captured nodes themselves. By definition he would not be able to interfere with the message exchange of other nodes. Under a more relaxed, but also more economical security regime, the influence he can exercise mostly depends on the number of captured nodes. Sometimes, the geographical location of a node makes it more valuable to the adversary if many messages have to pass through this node, for example.

3.5.2 Multipath Communication

In communication settings where no end-to-end secret keys are available but where it is possible to construct a set of disjoint paths between the message source and its destination, we can achieve a level of security that compensates the lack of authentication to a certain degree and is sufficient for many applications.

The simplest form of multipath communication is replication: the same message is sent over multiple paths in parallel. If at least one copy arrives at the destination, the message transmission has been successful, thus providing a good countermeasure against message dropping (denial-of-service) attacks. Attacks on the integrity of messages can, if not prevented, at least be detected if multiple copies of a message, but with different contents, arrive. Of course, this method does not provide any protection against eavesdropping. Nevertheless, it is suitable for a Diffie-Hellman key exchange [54] (without key authentication, but possibly with integrity protection).

Shamir's secret sharing scheme [165] allows the splitting of a message into n pieces, k of which are sufficient to reconstruct the original message. The scheme is secure against an adversary that manages to get hold of at most $k-1$ pieces of the message, which will not yield any information about the message at all. Of course, if the adversary obtains k pieces, the confidentiality of the message is broken. Apart from threshold confidentiality, Shamir's scheme also achieves robustness against the loss of pieces if $n > k$. However, it comes at the cost of an n-fold increase in size, since each piece is as large as the original message. Rabin [147] has devised a more efficient method that retains reliability, but lacks the feature of not revealing any information.

Having n disjoint paths available and using secret sharing, we can transmit each piece on an independent path. The adversary will not be able to reconstruct the message unless he can eavesdrop on at least k paths. When faced with such a constrained attacker, this method can preserve the confidentiality of a message.

One can distinguish between node disjointness and link disjointness. In wireless sensor networks, we can generally say that nodes are more important than links. Link-centricity should be replaced by node-centricity for the following reasons. (1) If one link suffers a failure because of external forces, it is likely that all other links in a certain area are exposed to the same external forces and thus suffer the same failure. Thus, disjointness of geographically close links does not help in this case. (2) The bandwidth of links is limited by the computational capacity of the nodes, and not by wireless technology (as in the conventional case). (3) There are many links compared to nodes, which alone makes nodes somewhat more significant; for each node, there is a potentially large number of links – between 1 and N (network size).

From a security perspective, disjointness of paths is desirable: two paths that share a common node are only as good as that common node allows. An example are public-key infrastructures without a central certification authority [152]. Here, trust in the authenticity of public keys is indirectly transferred over multiple hops and each certification path is only as good as its weakest

link, thus disjointness is required. For WSN, we should consider a different security model, which allows us to relax the disjointness condition. Considering a geometrically constrained attacker, the probability that a node is captured is high when one of its neighbours is captured. Therefore, if a node near one of the endpoints is captured, it is likely that the endpoint is captured as well. In this case, the connection is broken by definition. Thus, we can consider two paths as "disjoint" if all the shared nodes between them are close to the endpoints, thereby increasing the risk of security breaches only minimally.

The most important task in a disjoint path setting is to find or set up disjoint paths in the first place. We will present a method, which takes the previous considerations into account, in Chapter 5.

Remark Multipath communication is very common in the physical world, although multiple paths are used rather sequentially than in parallel. One popular example is the distribution of credit cards and their associated PIN numbers. Here, separate letters are used for sending the credit card itself and its PIN number, and one of them is sent with a delay of a few days. The underlying assumption is that under these conditions, it is unlikely that both letters can be intercepted by malicious parties. Another example is key distribution for e-banking. The keys are sent by paper mail, where endpoint verification is possible, while the actual banking statements etc. are sent via the Internet.

3.5.3 Assessing the Security Level

When devising techniques that provide a level of security that is not equivalent to end-to-end security, it is helpful to be able to somehow quantify how close they are able to approximate end-to-end security. For wireless sensor networks, such a quantification is canonically based on the number of nodes that are able to communicate (or, generally, act) securely. For wireless sensor networks, we propose two measures: the fraction of node pairs that are able to communicate securely, and the number of nodes that are able to participate in agreement schemes.

Secure Pairwise Communication

A fundamental requirement in a communication system is that two hosts can communicate securely (w.r.t. confidentiality or authenticity) with each other. The probability with which a connection provides the required security properties is an indication of the security level the network provides. Related to

this is the fraction of pairs of uncompromised nodes that can communicate securely. From the adversary's point of view, this means that the adversary is able to manipulate a certain fraction of the messages that are sent throughout the network.

Note the difference between link and path security. Link security prevents a passive adversary from reading messages that are sent between two adjacent nodes. In case of an active adversary that operates only at the link level, link security prevents the adversary from manipulating messages. If the active adversary is also able to compromise nodes, link security is not sufficient to keep the adversary from eavesdropping or manipulation. In a multi-hop environment and an adversary of the latter kind, link level security provides no protection against this adversary. In this case, it is necessary to protect communication paths instead of links only. A path provides a connection between two nodes that are non-adjacent. All nodes on that path cooperate relaying messages. There is a certain level of trust that must be put into them to provide the necessary protection.

A *live path* is an end-to-end connection in which both endpoints are uncompromised. This is independent of the fact whether these nodes can communicate securely with each other or not. If they have a unique secret shared key, secure communication is possible independent of the number of compromised nodes that relay the messages. The shared key guarantees that intermediate nodes cannot tamper with the message. If the nodes use disjoint multi-path routing, a shared secret key is not necessary, but there is a limit on the number of compromised paths that can be tolerated. Using a secret sharing scheme, t out of n available paths may be compromised without affecting the traffic.

A *functional path* is a path that provides a secure connection between two uncompromised endpoints. If end-to-end security means are available, such as a shared key, any path that is alive is considered functional. Only its ability to relay messages is required. Security is provided by the shared secret key. If no end-to-end security means are available, the path itself is responsible for providing security properties. For example, in a hop-to-hop authentication scheme all nodes are trusted to relay a message untampered.

For defining a measure for the level of security, we consider the set of live paths as the basic reference. This seems sensible as the integrity of a path is irrelevant if one of the endpoints is compromised. Formally, we define as a measure for the security of a sensor network the quotient

$$\frac{|\{\pi \in \Pi : \pi \text{ is functional}\}|}{|\{\pi \in \Pi : \pi \text{ is alive}\}|} \tag{3.1}$$

where Π is the set of all paths (i.e., pairwise multi-hop connections) in the net-

work. This measure is defined only if there is at least one live path in the network. It is consistent with end-to-end security techniques: for these, it always yields 1, independent of the number of compromised nodes in the network, since every live path is also functional.

Byzantine Agreement

In many applications of distributed systems, at some point a consensus problem has to be solved. For example, the hosts have to agree whether or not to perform a specific action, such as committing a database transaction. In sensor networks, nodes may have to agree on the value of aggregated sensor data before reporting it. Or a distributed intrusion detection system is concerned with the expulsion of a sensor node that is suspected to falsify sensor readings.

The problem of reaching consensus in the presence of malicious faults is called the Byzantine agreement problem. It is well-known that solutions to this problem exist only under specific conditions on the synchronization of hosts, the characteristics of the communication network, and the authentication of messages. We will not go into detailed descriptions of appropriate conditions. We are interested in evaluating the ability of a network to reach consensus when it is subject to an attack. This evaluation provides a metrics for the level of security that is delivered by the network.

Our model is a synchronous system with point-to-point connections. In a fully connected network, this would allow for Byzantine agreement in case there are $n > 3t$ nodes in the network. Since we are dealing with a sparsely connected, multi-hop network, message authentication is used to simulate full connectivity, i.e. provide resilient point-to-point connections. Digital signatures would allow tolerating arbitrary values of t, but we disregard this possibility here and concentrate on end-to-end security properties.

The synchrony assumption is a strong assumption to be made in a sensor network. This assumption demands that messages are reliably transmitted between nodes within a "round" of operation. This requires a reliable message transport service that retransmits lost messages. Transmission failures have to be detected. Such a service should be possible to implement in a sensor network, though it may be unusable in practice.

Protocols for distributed consensus are very complex. In order to tolerate t faulty nodes, they require at least $t + 1$ rounds of message exchanges between all node pairs. It is clearly unacceptable in a large-scale sensor network to include all nodes in such a protocol. Thus, we do not consider Byzantine agreement among all nodes in a sensor network to be of practical value. However,

we consider the ability of a network to reach consensus useful as a means for comparing authentication schemes. Authentication based on pairwise keys provides the highest security level in this framework, since it prevents the adversary from manipulating messages. This allows secure communication among all pairs of uncompromised nodes. Alternative authentication schemes, as described in chapters 6 and 5, provide secure communication only for a fraction of the node pairs. This reduces the ability of the uncompromised nodes to reach a network-wide consensus. The fraction of nodes still able to participate in this network-wide consensus yields a quantitative measure for the provided security.

Bibliographic notes The problem of Byzantine agreement has been defined by Lamport, Shostak, and Pease [138, 108]. Protocols and complexity bounds for distributed consensus are presented in an accessible way in the book by Nancy Lynch [119].

3.6 Related Work

The standard attacker model in cryptographic research has been defined by Dolev and Yao [57]. It assumes a distributed system in which hosts communicate by exchanging messages. It considers two (or more) honest parties that are trying to communicate, while the attacker tries to tamper with this communication. The attacker is assumed to be nearly omnipotent, having access to all communications and being able to suppress or fabricate messages. He is only limited by cryptography, which is assumed to be secure. This model has proven to be useful for the analysis of cryptographic protocols. However, as discussed in [45] within the context of ubiquitous computing, often other threat models are more useful for the analysis of security protocols. In such settings, additional security assumptions are being made that allow to relax the Dolev-Yao model by going beyond the availability of unbreakable cryptographic primitives.

One condition, which is often fulfilled in practice, is the existence of a low-bandwidth but secure channel that can be used for a short period of time. One example where this is exploited is the "pairing" of consumer Bluetooth devices. Here, the (human) user enters the same random code on both devices. This code is then used as a seed for creating a secret key [106] that allows future secure communication between these devices. The underlying assumption is that the attacker does not have access to the codes entered by the user.

Location-constrained channels [11] are a similar means for authenticating messages or entities. They can be used to restrict access to physically close users, thereby excluding an attacker that is too far away. However, setting up such a channel is non-trivial. Restricting the range of a wireless radio transmission is often not possible. For example, using special equipment it is possible to interact with a Bluetooth-enabled mobile phone from a distance as long as 1.78 km (in contrast to the advertised 10 m) [80]. Alternatives such as sound or light have been proposed, for which range limitation can be achieved more reliably.

Access to the communication medium can also be restricted in time, for example during the deployment phase of a WSN. Just after the nodes have been positioned, for example after being dropped from a plane, the adversary may not have access to the deployment area. This short time window can then be used to set up secret keys between the nodes by exchanging short plain-text messages [5]. The LEAP key distribution scheme [205] depends on a similar condition. Here, it is assumed that the time consumed for a node to detect its direct neighbours will be shorter than the time it takes for the attacker to compromise a node. This condition is necessary since each node initially uses a master key. This key is erased after neighbour discovery has taken place (and before the attacker is successful). Nevertheless, the use of a master key that is the same for all nodes is problematic. A single captured node with the master key still intact would suffice to take over the entire network. Therefore, it is required that all nodes erase the key reliably.

3.7 Summary

The topic of this chapter was the description of a security model for wireless sensor networks. The purpose of such a model is to provide a framework within which possible attacks on a sensor network can be considered, such that the most likely type of attacks can be determined for a certain deployment.

We first considered how a wireless sensor network could be attacked in general, namely by hardware or software manipulation, or through its communication interfaces. We then discussed the possible objectives of an attacker. They not only refer to the resources that a WSN provides but also include detection evasion. Following that, we described the “design space” for attackers, which determines the attacker’s characteristics and capabilities. Since our major concern is communication security in wireless sensor networks, we took a closer look at end-to-end security measures and their associated costs for implementation in WSNs.

As we concluded that end-to-end mechanisms would be either too costly or too constraining in many applications of wireless sensor networks, we considered the approximation of end-to-end security as an alternative approach that provides a level of security that is sufficient for many purposes and may deter potential attackers in many cases.

Chapter 4

Key Establishment

Shared secret keys are a prerequisite for communication that is secured by cryptographic means with regard to the three "classical" security properties: confidentiality, integrity, and authentication. In wireless sensor networks, confidentiality is important if, for example, sensor readings or aggregated data are regarded as secrets that have to be protected against unauthorized reading. Integrity and authentication are required for WSNs that operate in critical environments where the manipulation of data may have harmful consequences.

In communicating systems, cryptographic keys can be established in a variety of ways, which can be broadly categorized in two classes. The first is ususally described as *key exchange*: Two (or more) parties each contribute a partial key that are combined into the final key. A key exchange protocol solves the problem of how to efficiently convey the partial key to the other party without compromising the final key. The second class is usually called *key agreement*. Here, it is not necessary that both parties contribute key material. The final key can be chosen externally (and both parties simply agree to use it), or it can be assigned by one party to the other. Such a case is sometimes called *key transport*.

In identity-based key agreement protocols, the only information that may be exchanged are the identities of the involved parties. Identities are not equivalent to keys as they are (often) not randomly chosen, static, and public. Based on the identities, the shared key is determined. This often involves the use of additional key material, which can be either (pseudo-)randomly constructed or could be already present. The latter case is the result of key pre-distribution.

In this chapter, we describe identity-based key agreement protocols for wireless sensor networks. The properties of such protocols match the resource constraints of WSNs such that they are advantageous over alternative key agreement approaches.

4.1 Requirements for Key Agreement

The objective of key establishment protocols is to create a key that is known to, and only to, the legitimate nodes involved in the protocol, thereby creating a secure association between these nodes. That means, no information about the key whatsoever must leak to outsiders. In general, this is only achievable by using resource-intensive protocols based on public-key security, or by pairwise key distribution, which are both impractical for wireless sensor networks. Public-key cryptography is expensive and slow, and pairwise key distribution would require extensive memory capacity ($N-1$ keys have to be stored on each node for network size N). However, in sensor networks, it is often not necessary to guarantee perfect key confidentiality but a certain probabilistic security level is acceptable. This is an opportunity for the use of more economical key agreement schemes.

In general, a key agreement scheme for a WSN should be based on the exchange of only a very small amount of data and it should be sufficient to send one message in each direction. Additionally, the scheme should be based on cryptographic functions that are easy to compute.

From a cryptographic point of view, a key establishment protocol must be designed in a way to ensure that it is secure against an attacker as defined by Dolev and Yao, i.e. the protocol must not allow the rearrangement or manipulation of messages such that the attacker can eventually learn the key. It is valid, however, to assume that the cryptographic primitives are secure against cryptanalysis. In practice, this latter assumption means that only widely recognized and tested cryptographic algorithms must be used.

For specific application scenarios, usually further assumptions can be made that allow the relaxation of the threat model (as discussed in the previous chapter). In wireless sensor networks, it may be acceptable that a pairwise key used by two nodes is also known to a small fraction of the other nodes in the network. This *global trust assumption* means that a key agreement scheme has to provide key confidentiality only with a certain probability (which, of course, should be as high as reasonably possible).

4.2 Random Key Pre-Distribution

Random key pre-distribution for sensor networks has been first proposed by Eschenauer and Gligor [64]. The objective of a key pre-distribution scheme is to allow two sensor nodes establish a shared secret key that can be used for securing their bilateral communication, i.e. for encrypting or authenticating

messages. The EG scheme is based on a pool of keys of which a subset is known to every node. Two nodes can derive a pairwise key, a *link key*, from the intersection of their subsets. Note that here, *link* refers to a connection in the authentication graph of a network, which is not necessarily equivalent to a radio link.

Such a scheme does not provide "perfect" security since it cannot be guaranteed that a derived key is known exclusively to one pair of nodes. An attacker who captures a set of nodes acquires the key material known to these nodes and can, with a certain probability, derive from that the link key that has been established between two other, uncompromised nodes. Depending on the chosen parameters, the scheme provides a certain *resilience* against such attacks.

Due to its probabilistic nature, the scheme cannot guarantee that two nodes will be able to establish a link key at all, as it is possible that the intersection of their key material subsets is empty. The parameters can be chosen such that *connectivity* (i.e., the probability with which two nodes can establish a pairwise link key) will be high, but it will be usually below 1, and a high connectivity will lead to reduced resilience.

4.2.1 A Model for Key Pre-Distribution

The following elements are required for a random key pre-distribution scheme.

The *key space* defines the set of values that are eligible as keys. These values must be of sufficient length to provide computational security when being used as cryptographic keys. A typical length could be 128 bit.

For some pre-distribution schemes, for example full pairwise key distribution, keys are drawn from the complete key space. For random key pre-distribution, only a subset of the complete key space is used. This subset is randomly chosen and is called *key pool* $\mathcal{K}$.[1] The size of the key pool determines the connectivity and resilience of the scheme, as we will see later. We will denote the key pool size as S.

We assume that each of the N nodes has a unique identifier ID_u ($u \in \{1,\ldots,N\}$). To each node, a set of keys is assigned, which is called a *key ring*. The elements of a key ring are selected from $\mathcal{K}$ by using a selection function F, which will be defined shortly. First, we define the following elements:

- $\mathcal{K}$ is the key pool, i.e. an ordered set of keys.

[1] It should be noted that some cryptographic algorithms, such as DES or Blowfish, have "weak keys", i.e. keys with certain properties that lead to insecure results. Although weak keys are usually very rare, one might check if the key pool contains such keys and replace them, if a cryptographic algorithm with weak keys is being used. There are no known weak keys for the current cryptographic standard AES/Rijndael.

- $S = |\mathcal{K}|$ denotes the size of the key pool.
- m is a global parameter and denotes the key ring size, which is the same for all nodes.
- A (pseudo-) random number sequence generator Ψ^g where g acts as a seed.

Ψ takes four parameters: a node identifier, a lower bound on the output, an upper bound on the output, and the number of elements in the generated sequence. Ψ may generate "real" random numbers or pseudo-random numbers. In any case, we assume that Ψ is "stable", i.e. for the same input, it will produce the same output in a given context (which is determined by g). The produced numbers are integers within the given (inclusive) bounds. If the numbers are generated pseudo-randomly, the node identifier is used as part of the seed. This enables other parties to reproduce the same sequence of numbers. As an additional constraint, Ψ will produce any number at most once.

Based on these elements, we define a key selection function F, which returns for a given node ID a set of keys:

$$F(ID_u) = \langle \mathcal{K}[v_1], \ldots, \mathcal{K}[v_m] \rangle$$

where $\Psi^g(ID_u, 1, S, m) = \langle v_1, \ldots, v_m \rangle$ for a given g.

4.2.2 Pre-Distribution Phase

The initial phase of a key pre-distribution scheme is performed in a secure environment, assuming that the adversary does not have access to the nodes during this phase, e.g. before deployment. The key distribution center (KDC) is responsible for performing this initial phase. The KDC computes for each node u the key ring $F(ID_u)$ and loads the selected keys onto the node.

There are two parameters in this scheme that can be varied and that are important regarding the security properties of the scheme: the size S of the key pool and the size m of key rings. There is a trade-off between connectivity and attack resilience. The larger the key pool is, the more resilient the scheme will be against an attacker, since the probability that a captured node contains the root keys required to derive a certain link key is lower. However, connectivity suffers since the intersection of the key rings of two nodes trying to establish a link key tends to be smaller. Larger key rings, on the other hand, increase connectivity as the intersection of two key rings contains more elements. But an attacker also learns more root keys when capturing a single node, thereby increasing his chance to compromise a link key.

4.2.3 Identity-based Key Rings

The selection of keys in a key ring can be entirely random, as it has been initially proposed by Eschenauer and Gligor. Alternatively, it can be based on a public pseudo-random sequence of indices that is derived from the identity of the node (as proposed by Zhu et al. [204]). This identity-based selection of keys has the advantage that during the key agreement phase, nodes have to exchange only their IDs in order to be able to reconstruct which keys the other node holds. It has the additional advantage of providing a kind of entity authentication. More precisely, by verifying that a node has knowledge of a specific set of keys, it is established that this node belongs to the group of nodes that are legitimatly participating in the network's operation.

Since the generator Ψ is accessible to all nodes, every node can determine the indices of the keys in any other node's key ring, if the other node's identity is known. However, the actual keys are not disclosed. Thereby, nodes can determine their common set of keys, but a party that does not know the keys in advance will not learn them.

4.2.4 Establishing the Common Key Set

In order to derive a link key, two nodes have to learn which of the nodes from the key pool they have in common. There are several possibilities for that, with different advantages and disadvantages.

As proposed in [64], a node can simply broadcast the indices of the keys in its key ring. Neighbours overhearing this message compare the indices to their own and decide whether they are able to establish a link key to the broadcasting node.

When key rings are selected based on a node's identity, it is sufficient that a node broadcasts its own ID. Nodes that receive that message can derive the set of of key indices from that ID and determine the shared set of keys. This is easily achieved by making the following function available to each node:

$$\psi(ID_u) = \Psi^g(ID_u, 1, S, m) \ .$$

(Since storing the complete output of Ψ during the pre-distribution phase in each node is infeasible, this is only efficient if Ψ generates its output pseudo-randomly, for example based on a hash function, such that a node can do this calculation itself.)

These two approaches have the disadvantage that an adversary learns which keys (more precisely, their indices) are contained in a node's key ring. This might facilitate certain attacks since the adversary can now selectively target

nodes and try to capture them in order to obtain certain keys which he does not yet possess. The first approach has the additional disadvantage that the broadcast messages are quite large, comprising at least $m \log_2 S$ bits of index information.

Another possibility is to broadcast the key indices only in an indirect manner. As proposed similarly in [143], a node broadcasts the following information:

$$\alpha, H(x_0, \alpha), H(x_1, \alpha), \ldots, H(x_{k-1}, \alpha)$$

where α is a random nonce, H is a keyed hash function, and x_i are the keys from the node's key ring. A value $H(\cdot)$ is called a *hash commitment*. An overhearing node computes $H(y_i, \alpha)$ for every key y_i in its own key ring. By comparing the results to the received values, it can determine which node it shares with the broadcasting node.

This last approach has the advantage that it does not reveal any information about the key indices unless the corresponding keys are already known. Therefore, it does not help an adversary to focus his attack on certain nodes. On the other hand, overhearing nodes have to perform a moderate number of computations. They have to execute m applications of function H and m^2 comparisons (each of their own results with each of the received values). Also, the broadcast message is quite large.

Note that in [143], instead of a keyed hash function, encryption and decryption operations were used. The broadcasting node encrypts α with each of its keys, and a receiver decrypts every value with each of its keys, resulting in m^2 decryptions. Additionally, the m^2 comparisons are still necessary. By eliminating the need to apply each key to each of the transmitted values, our version reduces complexity from $O(m^2 + m^2)$ to $O(m + m^2)$ on the receiver's side.

Sending hash commitments uses a lot of bandwidth. The output of a hash function has n bits (e.g. $n = 160$ for SHA-1), and is considered completely random. The size of a broadcast message can be reduced by including not the full hash commitments, but only a substring for each of them. Instead of n bits, only $k < n$ bits could be transmitted for each hash commitment. The comparisons are then based on these substrings. Since they are smaller, the likelihood that a comparison yields a false positive result is higher than for the full length.

The impact of a false positive match would be that a node assumes that its set of shared keys with another node is larger than it is in reality. This leads to an attempt at establishing a shared key based on this extended set of keys, which would fail. The nodes would then not be able to communicate securely. Therefore, we would like to keep the probability of such an event as small as

possible. Based on the birthday paradox, we can compute which k is required such that the probability of such false positives is below a certain threshold.

The probability that at least two collisions occur if v samples are given out of d possible values, is [192]:

$$p(v,d) = 1 - \frac{d!}{(d-v)!d^v}$$

Of course, if $v > d$, the probability becomes 1. In our case, we set $d = 2^k$ and $v = S$. Using the approximation given in [196], we then get for a desired collision probabiliy p,

$$k = \left\lceil \log_2 \frac{S^2}{2\ln\left(\frac{1}{1-p}\right)} \right\rceil$$

Example 1 (Reduced hash commitments for key comparison)**.** Let's assume the key pool contains $S = |\mathcal{K}| = 50000$ elements. We demand that the probability for false matches when comparing the hash commitments of keys is $p = 0.01$. Using above formula, we get $k = 37$, i.e. compared to full hash commitments of length 160 using SHA-1, we can save 123 bits for each key without significant impact to the connectivity.

4.2.5 Key Derivation

After establishing its common set of root keys, a pair of nodes can derive a link key from the root keys. If the common set is empty, the nodes don't share any keys and thus they cannot establish a link key. In the basic scheme of Eschenauer and Gligor, one common key is sufficient to establish a link key. The q-composite scheme [39], which is an extension of the basic scheme, demands that at least q ($q \geq 1$) keys are avaialable and are combined to form the link key.

In the basic scheme, even if there are two or more keys in the common set, only one of them is being used as the link key. However, better resilience can be achieved if all the available keys are used. We now discuss different methods for deriving a link key from a number of root keys.

Let q be the number of root keys in the common key set, and x_i ($0 \leq i < q$) denote these common keys. These keys can be combined in the following way (proposed in [143]):

$$K = x_0 \oplus x_1 \oplus \ldots \oplus x_{q-1}$$

where $\oplus$ is the bitwise exclusive-or operation. This will yield a link key of the same size as the root keys.

Another possibility for deriving a link key is to apply a hash function on the input material. The root keys are concatenated first, resulting in a bitstring to which the hash function is applied:

$$K = h(x_0|x_1|\ldots|x_{q-1})$$

This method has been proposed in [39]. In contrast to the previous approach, the order in which the keys are concatenated must be the same at both communicating parties. For example, the order in which the root keys occur in the key pool could be used. Key derivation through the application of a hash function is a common method for deriving a cryptographic key from input material such as a password [103].

Note that a hash function can handle input data of arbitrary length (see section 2.8.1) but yields output data of fixed size. If the result is too large (for example, SHA-256 yields 256 bit), a substring of the output data can be used as the key.

It may be advisable to include a piece of random data in the key derivation. One of the nodes (or both) may generate a random value that is then included by both nodes in the key derivation, for example by appending that random value to the root keys before applying the hash function. This gives better resilience against attacks that occur later. Even if the adversary finds out which keys were involved in deriving the key, he cannot simply repeat the process. Only if he has recorded all the network traffic during key establishment (when the random values are exchanged), he has all data he needs to reconstruct the keys. Otherwise, he would have to try out all possible random values, which is infeasible if the space from which they are drawn is large enough.

Note that if we assume the adversary does not record *any* traffic data during key establishment, we can drop the key establishment scheme altogether and simply use random values as pairwise keys. This is the approach proposed in [5].

4.2.6 Connectivity

The term *connectivity* refers to the probability with which two nodes can establish a link key. In a pre-distribution scheme where pairwise keys are distributed before network deployment, *full connectivity* can be achieved, since every node has a link key with every other node. In a probabilistic scheme as described above, connectivity is traded against attack resilience. High resilience can be achieved if lower connectivity is acceptable. In a highly redundant sensor network, a relatively small connectivity (for example, 30%) could be appropriate.

The link keys define the *authentication graph* that exists "on top" of the communication graph, in which the edges are determined by radio connectivity. (The vertices of both graphs correspond to the sensor nodes.) If we allow link keys only between nodes that are spatially close, i.e. between which exists a radio link, the authentication graph is a subgraph of the communication graph. If we also allow link keys between remote nodes, the edgeds in the authentication graph become arbitrarily distributed, being constrained only by the ability of pairs of nodes to establish link keys. In the former case, the authentication graph is geometric, while in the latter case, it is closer to a random graph.

Nodes with few neighbours may not be able to connect to any neighbour at all. A small number of nodes being disconnected from the main component of the network should not pose a problem in practice. As a matter of fact, it can be expected that some nodes will fail to function during the course of normal operation, and the network will have to deal with such failures anyway. Some application scenarios may depend on a network that is densely connected, with each node having many neighbours to talk to. For other scenarios, a lightly connected network may suffice. It is part of deployment planning to determine the necessary and economically feasible density of the network.

A random predistribution scheme yields a certain probability p_c with which two nodes are able to establish a link key. This probability depends both on the key pool size and the key ring size. While the key ring size is usually constrained by the available memory in the nodes, the size of the key pool can be varied arbitrarily, since it exists in full only in the key distribution center.

As defined in section 4.2.1, let S be the key pool size and m the number of root keys in a node's key ring. In order to determine the probability p_c, we first determine the probability with which two nodes share exactly i keys ($0 \leq i \leq m$), which is

$$Pr[i \text{ shared}] = \frac{\binom{S}{i}\binom{S-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{S}{m}^2} \tag{4.1}$$

Chan et al. [39] have given a derivation of this expression. Here, we give a slightly modified explanation of its terms. The numerator designates the number of possibilities to select two key rings which have exactly i common elements:

- We assume i common keys. There are $\binom{S}{i}$ ways to select i items from the key pool.
- There are $m-i$ distinct keys left to be drawn for each node from the key pool, so $2(m-i)$ keys in total. The i keys that already have been deter-

mined as common keys cannot be selected again. Thus, there are $\binom{S-i}{2(m-i)}$ ways to select those remaining $2(m-i)$ keys.

- These $2(m-i)$ keys have to be distributed among the two nodes. Since each node gets $m-i$ keys, there are $\binom{2(m-i)}{m-i}$ ways to do that.
- The denominator equals the total number of possibilities to select two key rings.

Two nodes can establish a link key if they share sufficiently many root keys. For the q-composite schemes, q keys are required. The basic scheme requires one common key, thus its connectivity equals that of the $q = 1$ composite scheme. It is therefore sufficient to give the connectivity of the q-composite schemes, which is determined as

$$p_c = 1 - \sum_{i=0}^{q-1} Pr[i \text{ shared}] \; . \tag{4.2}$$

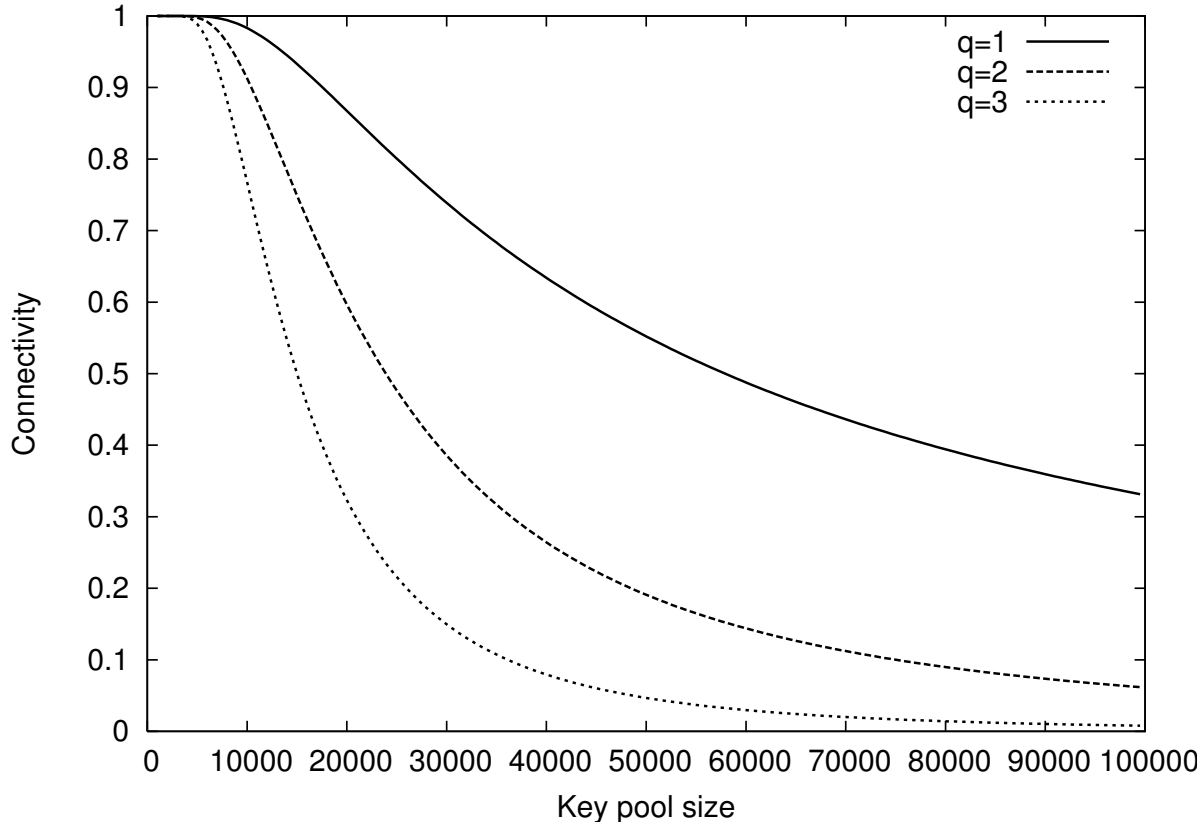


Figure 4.1: Connectivity depending on key pool size. A key ring size of 200 is assumed

Figure 4.1 shows the connectivity depending on the size of the key pool for the q-composite schemes for a fixed key ring size. The larger the key pool is chosen, the smaller the connectivity will be. Additionally, the connectivity

degrades more steeply the more root keys are required to set up a link key. This behaviour is, of course, expected, since the likelihood that a pair of nodes has q common root keys gets smaller with increasing q.

4.2.7 Resilience Against Link Key Compromise

Our attacker model assumes that if a node is compromised, the adversary gains complete access to the key material stored in that node. By collecting the key material of various compromised nodes, the adversary may gather a significant fraction of the key pool. He might then be able to recover the link key established between a pair of uncompromised nodes. This would allow the adversary to eavesdrop on the communication between the nodes (if the link key is used for encryption) or to inject messages into the network with a faked origin (if the link key is used for authentication).

As the attack parameter, we consider the number x of captured nodes. All root keys from these nodes are available to the adversary. We denote the probability with which a link key can be reconstructed from this collected key material as p_κ.

Under the basic scheme, a link key is derived from exactly one root key. The probability that this key is known to another node is $\frac{m}{S}$. Therefore, when faced with x compromised nodes, the probability that at least one of them has knowledge of that key is

$$p_\kappa = 1 - \left(1 - \frac{m}{S}\right)^x \tag{4.3}$$

Assuming the q-composite scheme, the probability that two nodes can establish a link key is given by p_c. The probability that a link key is being derived from i root keys is $(i \geq q)$:

$$\frac{Pr[i \text{ shared}]}{p_c}$$

In accordance to the reasoning above, the expected fraction of root keys being compromised is $1-(1-\frac{m}{S})^x$. If i keys were involved in deriving a link key, the probability that this link is compromised is $(1-(1-\frac{m}{S})^x)^i$ as all involved root keys have to be compromised. When x nodes have been captured, the probability that a link key between two uncompromised nodes is being compromised is therefore [39]:

$$p_\kappa^q = \sum_{i=q}^{m} \left(1 - \left(1 - \frac{m}{S}\right)^x\right)^i \frac{Pr[i \text{ shared}]}{p_c} \tag{4.4}$$

These measures of resilience show that these schemes reveal information about the whole network to an adversary who captures a fixed number of nodes.

p_κ and p_κ^q denote the probability with which a communication link between a pair of nodes are subject to eavesdropping or manipulation by an adversary who has captured x nodes. It has to be noted that this probability is independent of the number of nodes that are present in the network.

For illustration, Figure 4.2 shows graphs of equations 4.3 and 4.4 for different connectivities.

The difference between the basic scheme and the $q = 1$ composite scheme stems from the fact that in the basic scheme only one root key is used to determine a link key while in the composite scheme, all shared root keys between a pair of nodes are used for deriving the link key. This makes the composite scheme more resilient against an attacker who needs to obtain more key material to compromise a link key.

As pointed out in [39], the composite schemes are more resilient than the basic scheme for small-scale attacks. However, the composite schemes for $q > 1$ tend to reveal more information than the basic scheme to an attacker as he captures more nodes. Hence, at some point, this advantage turns into an disadvantage.

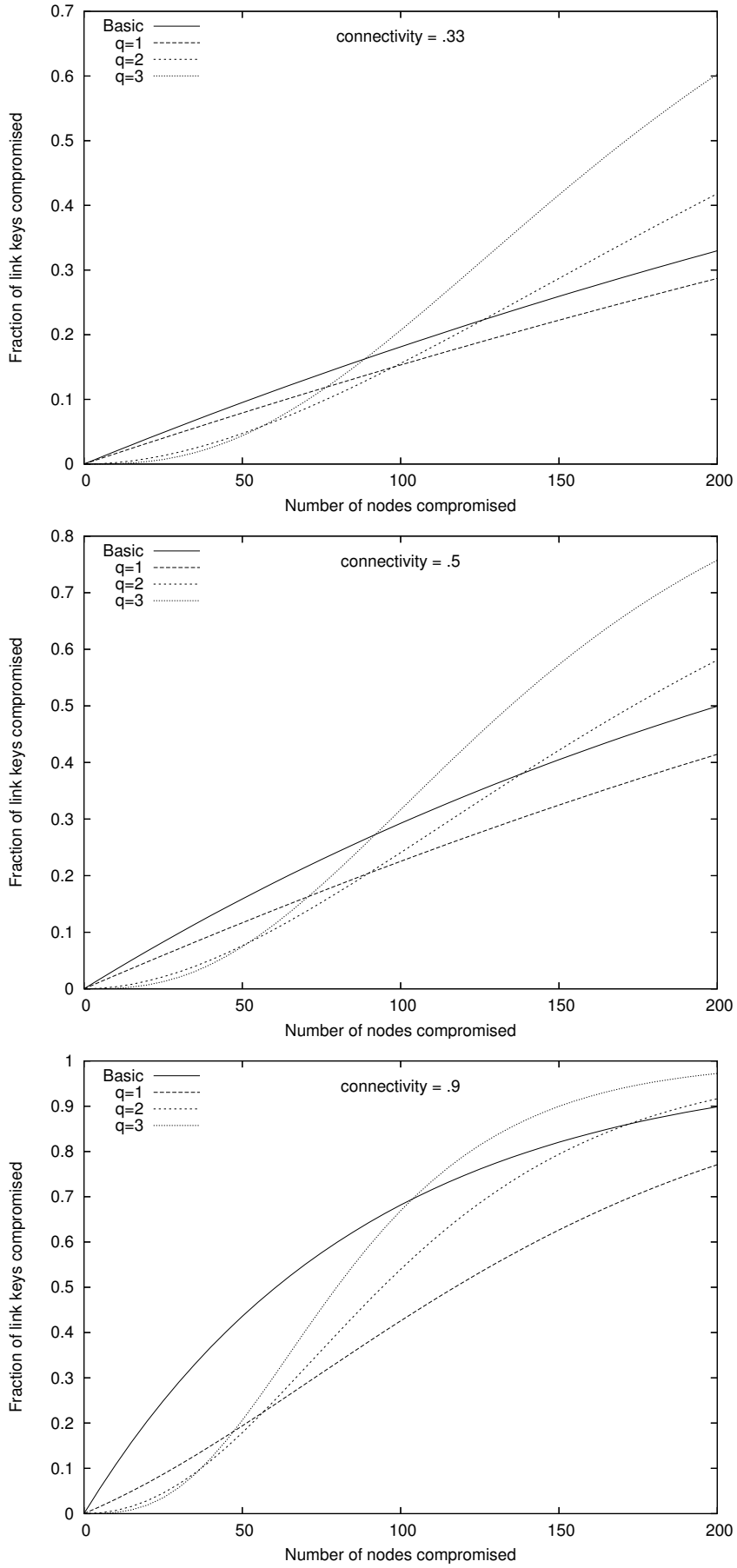


Figure 4.2: Link key compromise probability of key pre-distribution schemes under attack. The key ring size is $m = 200$. The key pool size depends on the connectivity and the parameter q

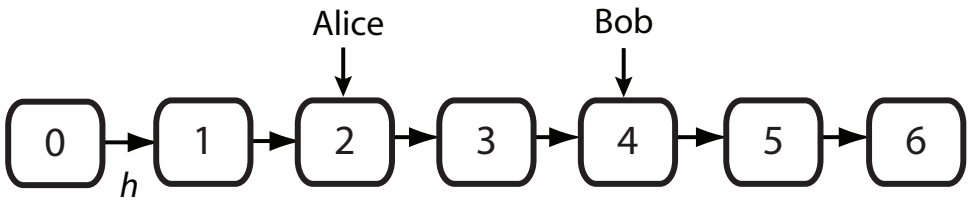


Figure 4.3: Two nodes using a hash chain to agree on a key

4.3 Key Agreement Based on Hash Chains

In this section, we describe how to use hash chains as a means for key agreement between two parties. We exploit the fact that element q of a hash chain can be derived from element p if and only if $p \leq q$.

4.3.1 Hash Chains

Let h be a one-way hash function that is publicly known. A *hash chain* is a sequence

$$\sigma = (z_\omega)$$

for $0 \leq \omega < T$ where $T \in \mathbb{N}$ is the *length* of the hash chain. z_0 is the *seed* of the hash chain. For all $\omega \geq 1$, the element $\sigma[\omega] = z_\omega$ is obtained by applying the hash function h to the previous element of the sequence, i.e.

$$z_\omega = h(z_{\omega-1}) = h^\omega(z_0) \ .$$

We refer to the elements of a hash chain as *(hash) chain values*. The *position* of a chain value z_ω is that value's index ω in the hash chain. Whenever we refer to a chain value, we implicitly assume that its index is also available.

Assuming that one element z_u of the hash chain is known, it is easy to compute all following elements $z_{u+v}, v > 0$ in the chain. However, the one-way property of h forbids it to compute any elements of the chain *preceding* the known value. In particular, it is not possible to reconstruct the seed z_0 of a hash chain unless z_0 is already known.

4.3.2 Single-Chain Key Agreement

Key distribution The key distribution center (KDC) generates a hash chain σ of length T. For each node X, the KDC selects randomly (and uniformly) a position ω_X on the hash chain ($0 < \omega_X < T$). The index ω_X and its associated chain value $\sigma[\omega_X]$ is distributed to the respective node. We assume that Alice and Bob receive the chain values $\sigma[\omega_A]$ and $\sigma[\omega_B]$, respectively.

Algorithm 1 Determine common key

Global values:
h: a one-way hash function

Input:
X: ID of the local entity executing the algorithm
Y: ID of the peer entity
ω_X, H_X: local hash chain information

Output:
The common chain key K

1: send(ω_X)
2: ω_Y := receive()
3: **if** $\omega_X < \omega_Y$ **then**
4: $\quad K := h^{\omega_Y - \omega_X}(H_X)$
5: **else**
6: $\quad K := H_X$
7: **end if**
8: return K

Key agreement Alice and Bob agree on a common value using the hash chain in the following way. First, they exchange their chain indices. Then, they select the chain value that is "lower" in the chain, i.e. the one with the bigger position index, as their common value. The agreed-upon value K will either be $\sigma[\omega_A]$ or $\sigma[\omega_B]$:

$$K = \sigma[\max(\omega_A, \omega_B)]$$

The node whose value is "higher" (i.e., closer to the seed) in the chain performs a number of applications of the hash function to eventually arrive at the other node's value. This number is determined by the difference between both position indices. This procedure is described as Algorithm 1.

Example 2 (Key agreement based on a hash chain)**.** Consider the situation in Figure 4.3 where a hash chain of length seven is shown (only indices, no key values are visible). Alice's value $\omega_A = 2$ is "higher" in the hash chain than Bob's value $\omega_B = 4$. Therefore, the agree-upon common value will be $\sigma[\omega_B]$. Alice performs two hash computations to obtain Bob's chain value from $\sigma[\omega_A]$. Note that Bob cannot construct Alice's value $\sigma[\omega_A]$ on his own and also does not learn that value during the protocol's execution.

4.3.3 Chain Key Resilience

Let's assume that there is an adversary Eve who tries to sneak on Alice and Bob's communication, i.e. carry out an attack on the confidentiality of their

shared key. As she cannot get direct access to Alice's or Bob's key material, she obtains key material from a single or several other nodes (such as Carol), for example by breaking into their key stores or by talking them into collaboration. This allows Eve to obtain up to j chain values and their respective position indices ν_i:

$$(\nu_1, \sigma[\nu_1]), (\nu_2, \sigma[\nu_2]), \ldots (\nu_j, \sigma[\nu_j])$$

The amount of key material that Eve is able to obtain is limited in most cases. Possible reasons are:

- Eve has to pay a certain price for each value, but has only a limited amount of money.
- The KDC distributes chain values to a finite number of nodes in the first place.
- There is only a limited number of nodes willing to cooperate in an attack.

The attack on Alice and Bob's key K is successful if Eve can construct K from her own chain values, i.e. if the following condition holds:

$$\mathcal{A} \equiv \exists i \in \{1, \ldots, j\}.\nu_i \leq \max(\omega_A, \omega_B) \ .$$

We observe that with a bigger j, it is more likely that Eve will be able to derive the key. We consider the probability of the complementary event that comprises the outcomes where all of Eve's chain values are located "below" $\max(\omega_A, \omega_B)$, which is the event that is favourable to Alice and Bob:

$$\overline{\mathcal{A}} \equiv \forall i \in \{1, \ldots, j\}.\nu_i > \max(\omega_A, \omega_B)\} \tag{4.5}$$

The probability of this event is

$$Pr[\overline{\mathcal{A}}] = \frac{\sum_{u=1}^{T} \sum_{v=u}^{T} (T-v)^j + \sum_{v=1}^{T} \sum_{u=v+1}^{T} (T-u)^j}{\sum_{u=1}^{T} \sum_{v=1}^{T} T^j} \tag{4.6}$$

which is explained as follows. We consider the number of favourable (from Alice's and Bob's point of view) outcomes versus the number of possible outcomes, when assigning a position to Eve. The favourable outcomes are computed by enumerating all possible locations of Alice's value (u) and Bob's value (v) and counting the possible positions of Eve that are favourable for Alice and Bob. Since Eve's positions have to be located further "down" in the chain than the maximum of Alice and Bob's values, there are $T-v$, respectively $T-u$, of such positions for each of the j values.

The number of possible outcomes is obtained by summing up all possible positions of Eve, which leads to $\sum_{u=1}^{T} \sum_{v=1}^{T} T^j = T \cdot T \cdot T^j = T^{j+2}$

Example 3. Consider hash chains of length $T = 100$ and $T = 500$, respectively. For different attack strengths j, the probability that a key established between Alice and Bob based on σ remains secure is, according to equation (4.6):

	$T = 100$	$T = 500$
$j = 1$	$Pr[\overline{\mathcal{A}}] = .328$	$Pr[\overline{\mathcal{A}}] = .332$
$j = 2$	$Pr[\overline{\mathcal{A}}] = .163$	$Pr[\overline{\mathcal{A}}] = .166$
$j = 3$	$Pr[\overline{\mathcal{A}}] = .0975$	$Pr[\overline{\mathcal{A}}] = .0995$

The example shows clearly that the resilience of this key agreement scheme is quite low for a single hash chain σ. In most practical cases, this would not be a favorable scheme if there is any chance that Eve obtains even a small number of values on σ. As we will see later, the scheme can be extended by using multiple hash chains simultaneously, which yields much better resilience.

The example also suggests that a larger hash chain is slightly advantageous, since the resilience of a shared key increases with a larger chain. On the other hand, a larger hash chain implies a larger overhead for Alice and Bob when they compute their shared value. Therefore, we have to consider the question what the optimal length of a hash chain would be.

4.3.4 Choosing the Length of a Hash Chain

The previous example suggests that the size of T makes a (small) difference for the resilience of this key agreement scheme. A larger T increases the probability that the key remains secure under attack. Intuitively, the reason seems to be that with a small T, the values are too close together. This makes a "collision" between Eve's value(s) and the key very likely. A collision occurs when one of Eve's values is located exactly at the same position in the hash chain as either Alice's or Bob's value. With a larger T, such collisions become more unlikely, giving Alice and Bob a small advantage.

When choosing a value for T, a trade-off is involved. In order to reduce collisions, T should be as large as possible. But the value of T is important in two more respects. First, it determines the overhead in establishing a link key between a pair of nodes. Second, storing a position index requires $\lceil \log_2 T \rceil$ bits. Thus, due to computational and storage efficiency, it is advisable to choose T as small as reasonably possible.

When T approaches infinity, the probability of collisions approaches zero. Obviously, such an infinite hash chain can not be implemented in a computing system. However, it defines the theoretically achievable resilience of the key agreement scheme. For each practical model, we can examine the disadvantage (for Alice and Bob) it results in, compared with this ideal model.

The resilience of the infinite model can be derived from the following simple combinatorial construction. We assume an urn model, with an urn initially holding two *blue* balls and j *red* balls. The blue balls represent the chain values of Alice and Bob, while the red balls represent those of Eve. The position indices on the hash chain are independently chosen. We simulate this by randomly placing the balls on a line of fixed length. We stipulate that the leftmost ball on that line represents the chain value with the smallest position index. The ball next to it corresponds to the key with the second-smallest index and so on. The two left-most balls determine whether Alice and Bob's key is secure or not. If these are both blue balls, their key is secure. Otherwise, at least one red ball is placed on the left from a blue ball, which means that there is at least one of Eve's values from which the key can be constructed.

There are $j+2$ balls in total, from which the two left-most balls are selected. Out of all possibilities to choose these two balls, only in one case two blue balls are chosen, which yields a probability for key security of

$$\beta = \frac{1}{\binom{j+2}{j}} = \frac{2}{(j+2)(j+1)} \tag{4.7}$$

It turns out that the difference between β and $Pr[\overline{\mathcal{A}}]$ is, from a practical point of view, quite small even for moderate sizes T. This fact is illustrated in Figure 4.4. It is therefore unnecessary to use long hash chains in order to achieve a good approximation to the theoretically possible resilience. For practical purposes, $T = 256$ should be sufficient in most cases. For example, this would lead, for $j = 2$, to a difference of $\beta - Pr[\overline{\mathcal{A}}] = 1.29 \cdot 10^{-3}$. This means that about one in one thousand link keys is compromised due to the fact that hash chains are finite. Compared to the overall compromise probability of $5/6$, this is a negligible fraction.

4.3.5 Discussion

Man-in-the-middle attack

Note that the protocol as it stands does not provide any authentication of the transmitted position indices, and the identities of the involved nodes are not verified. This makes a man-in-the-middle attack possible. An attacker, Eve, may intercept Alice's message to Bob and instead send her own position index to Bob. Bob would proceed, computing the shared key between him and Eve, but assuming that he shares this key with Alice. Similarly, Eve tricks Alice into establishing a shared key with her. Every further message exchange

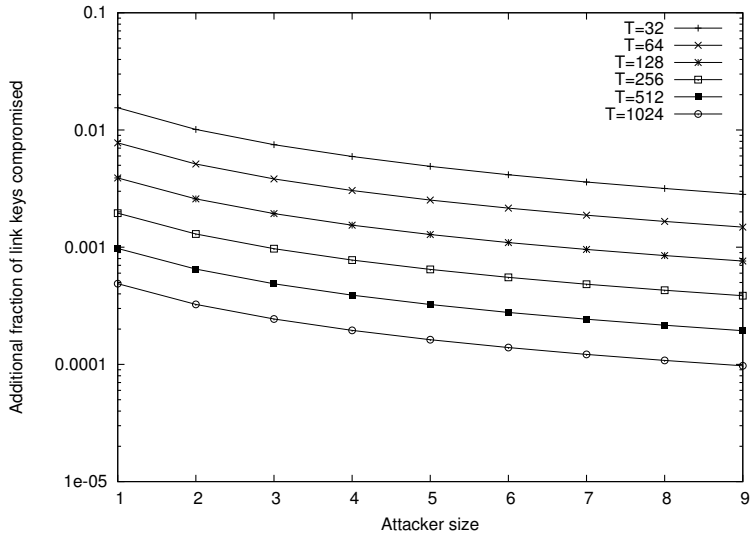


Figure 4.4: Fraction of compromised link keys due to the finite length of hash chains

between Alice and Bob passes through Eve, who can thus eavesdrop on their communications.

Key weakening

Another problem is the fact that each of the communication partners can deliberately weaken the shared key by transmitting a position index that is bigger than its own index. In the extreme case, a node may transmit the value $T-1$ as its position index. The agreed-upon-value will then be the last value of the hash chain, which can easily be constructed by any other node. Weakening the key is, of course, not in the interest of legitimate nodes. However, it may be a vulnerability in a poor implementation. Also, a communication partner collaborating with Eve can exploit it to make the key visible to Eve without having to send any message to Eve.

Computational overhead

Without loss of generality, assume that $\omega_A < \omega_B$. The number of applications of h that Alice must perform to arrive at Bob's chain value is the difference between both indices, which is at most $T-1$ (there are only T elements in the hash chain). We claim that on average, Alice must perform approximately $T/3$ such iterations. We do not formally prove this claim here, but refer to the geometric analogy: On a line segment of unit length, the expected distance of two randomly selected points is $1/3$ (proof is given in [193]). Scaling this distance up to a line of length T, the expected distance between two points (and therefore between Alice's and Bob's chain indices) is $T/3$.

Applicability of the Scheme

The rather low security of this key agreement scheme may seem disappointing at first. Without any effort, an adversary can expect to learn the keys resulting from at least $2/3$ of such agreements. This has to be considered an *unsecure key agreement scheme* compared to other schemes, such as those based on Diffie-Hellman or RSA. These provide a level of security that corresponds to the complexity of the discrete logarithm problem or integer factorization, respectively. They contain no element of chance that gives the adversary access to the secret key for free in a fraction of protocol executions.

Nevertheless, the proposed scheme has certain advantages. First, it does not require any public-key infrastructure. Second, it is easy to implement, and third, it has low computational complexity. As we will see in the following, multiple instances of the scheme used at the same time can be effectively used not only to implement a more secure key agreement scheme, but it can also be used to improve the security of random key distribution schemes.

4.4 Multiple Hash Chains for Key Agreement

We have shown that by using a single hash chain, it is possible to agree on a shared secret key, but with quite high probability that the adversary is able to break such a key. Still, a hash chain does provide a small degree of resilience (approximately one third of key agreements are secure when facing a single-valued attacker) and we are able to devise a scheme that uses multiple hash chains at the same time, thus providing a significantly higher resilience.

4.4.1 Key Distribution

For key distribution, the KDC sets up m hash chains $\{\sigma_i | 1 \leq i \leq m\}$. For simplicity, all hash chains are of equal length T (there would be no advantage choosing different lengths). Now, each node is assigned a position index on each of these chains. We assume a ID-based, pseudo-random assignment function Φ^g, where g is a seed. For node u and a hash chain i, Φ yields a chain position a, $0 \leq a < T$:

$$\Phi^g(ID_u, i) = a \ .$$

Node u is then assigned the key ring

$$\langle h^{a_1}(\sigma_1[0]), \ldots, h^{a_m}(\sigma_m[0]) \rangle$$

where $\langle a_1, \ldots, a_m \rangle = \langle \Phi^g(ID_u, 1), \ldots, \Phi^g(ID_u, m) \rangle$.

The assignment function can be made public, such that two nodes can autonomously determine each other's chain indices. This will not leak any information about the actual values at these positions, but minimzes the exchanged data. The assignment function can be made available to all nodes as φ, which we define as follows:

$$\varphi(ID_u, i) = \Phi^g(ID_u, i) = h(g \| ID_u \| i) \bmod T$$

where h is a hash function. Note that g is a constant value that can be chosen arbitrarily, but must be the same within one WSN deployment.

4.4.2 Key Agreement

Key agreement is straightforward. First, the nodes exchange their IDs. Then, each node determines the other's chain indices using φ and independently computes the common key for each hash chain. Finally, the keys of all chains are combined to form the link key as shown in Section 4.2.5.

4.4.3 Resilience

For evaluating the resilience of this key agreement scheme, we again assume that the adversary, Eve, has access to j key rings. In order to compromise a link key, Eve has to reconstruct all t chain keys that contribute to the link key. The probability that a single chain key "survives" an attack is approximated by equation (4.7). Let $\overline{\mathcal{A}_i}$ be the event that the key on chain i ($1 \leq i \leq t$) "survives" Eve's attack. The complementary event $\mathcal{A}_i$ corresponds to the chain key i being compromised. Since values on a hash chain depend only on the seed of that chain, and all seeds are chosen independently of each other, all these events $\mathcal{A}_i$ are pairwise independent. Hence, the probability that *all* t chain keys are being compromised by the attacker is

$$\begin{aligned} Pr[\mathcal{A}_1 \cap \mathcal{A}_2 \cap \ldots \cap \mathcal{A}_t] &= Pr[\mathcal{A}_1] \times Pr[\mathcal{A}_2] \times \ldots \times Pr[\mathcal{A}_t] \\ &= \left(1 - \frac{1}{\binom{j+2}{2}}\right)^t \end{aligned} \tag{4.8}$$

Depending on the strength of the adversary, which is given by j, and a desired resilience ε, we are able to choose t accordingly. Let ε be the desired probability of a link key compromise. The following condition has to be met in order to achieve this level of resilience:

$$\left(1 - \frac{1}{\binom{j+2}{2}}\right)^t < \varepsilon$$

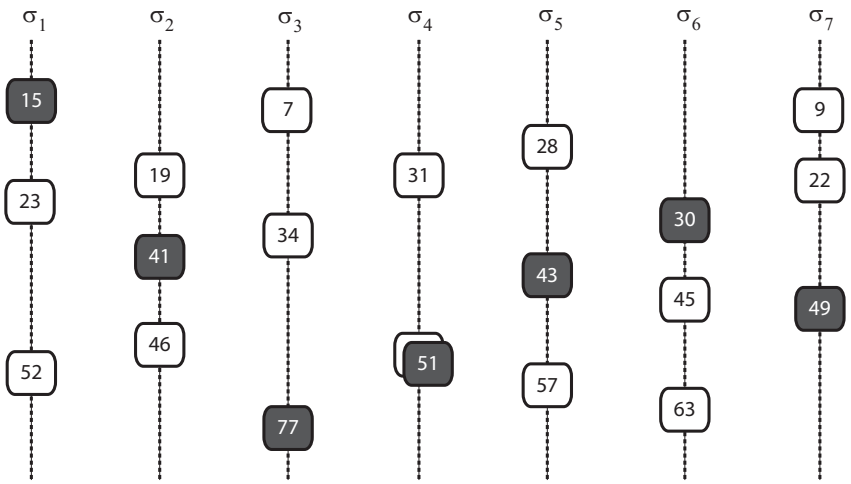


Figure 4.5: Attack configurations

After some transformation steps, we get

$$\frac{1}{\log_{\varepsilon}\left(1-\frac{2}{(j+2)(j+1)}\right)} \quad < \quad t \tag{4.9}$$

From this equation, we obtain that t grows with approximately the square of j if a certain resilience has to be maintained. Figure 4.6 illustrates this fact (a ten-fold increase in the attacker strength roughly requires 100 times the number of chains to maintain the same resilience).

Example 4. Figure 4.5 shows an example where two nodes establish a key based on seven hash chains. The white boxes represent the chain values of the two legitimate nodes. The black boxes represent those of the adversary ($j = 1$). The numbers shown indicate the position indices of the values. The actual numbers are not important, and it is also irrelevant to which legitimate node which white box belongs. Only the position of the black box relative to the lower white box is important. The chain value represented by the lower white box is the contribution of the hash chain to the shared key. Here, the adversary is able to construct all but two of these contributions (σ_3 and σ_7 are secure). It is sufficient for one contribution to be secure to get a secure link key.

Example 5 (Resilience of multiple hash chains)**.** Let's assume we expect an attacker of strength $j = 10$ and we would like to achieve a resilience of at least $\varepsilon = 0.01$, i.e. only one out of hundred link keys should be compromised. Using equation 4.9, we can determine the number of hash chains necessary to achieve this resilience:

$$t = \left\lceil \frac{1}{\log_{0.01}\left(1-\frac{2}{(10+2)(10+1)}\right)} \right\rceil = 302$$

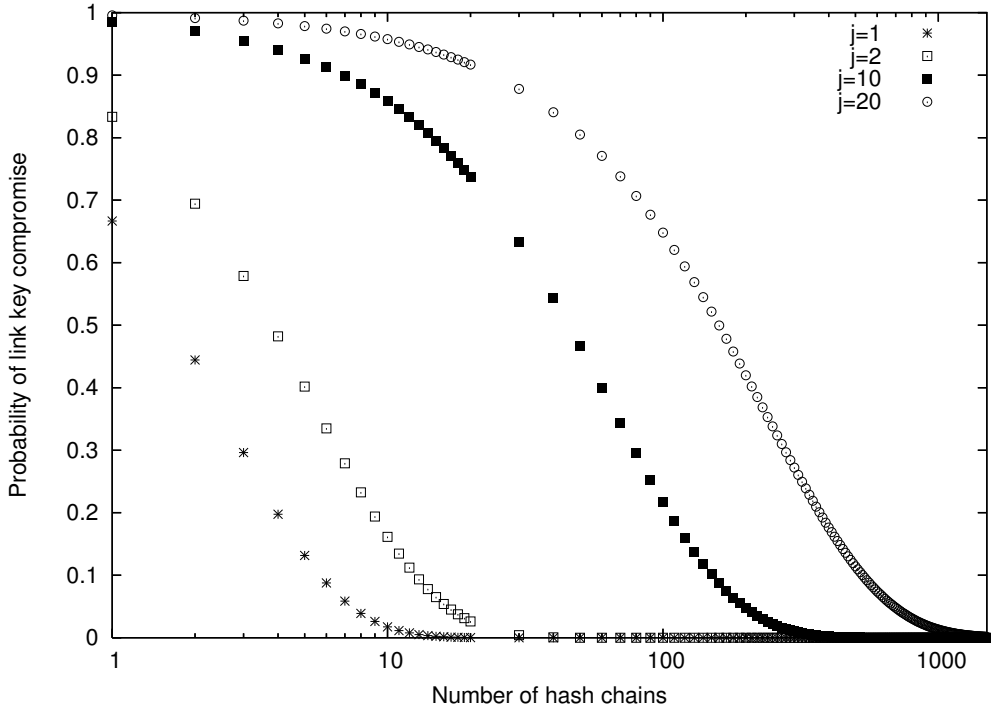


Figure 4.6: Probability of key compromise depending on the number of hash chains, for different sizes of adversaries. Note the logarithmic scale of the horizontal axis

4.4.4 Comparison with Random Key Pre-Distribution

Both the key agreement scheme based on multiple hash chains and the q-composite random key predistribution scheme provide a probabilistic level of resilience. In both cases, it is measured by the fraction of a set of link keys that can be expected to be compromised.

The hash chain scheme provides full connectivity, i.e. every node is able to establish a link key with any other node. In contrast, the q-composite scheme provides only partial connectivity. However, the latter allows the connectivity to be set arbitrarily close to one if a reduced level of resilience is accepted.

The resilience of the hash chain scheme depends on the number t of chains being used, and the size j of the attacker. The resilience of the q-composite scheme depends on the size of the attacker n, the connectivity p_c, and the parameter q.

In terms of computational complexity, the hash chain scheme requires a certain overhead since a number of hash computations have to be made by each node before the link key can be derived.

We are going to compare the resilience of both schemes by an example. We assume the same key ring size in both cases and ignore the storage overhead for indices, which are roughly the same size in both cases and small compared to the key size.

Example 6. Assume a key ring size of $m = 200$. For the $q-composite$ scheme, we further assume $q = 1$ and a desired connectivity $p_c = 0.999$. This requires a key pool size $S = 5992$. For the multiple hash chain scheme, we are using $t = m = 200$ hash chains. This results in approximately the same storage requirements for both schemes. Figure 4.7 shows both schemes in comparison with a varying attacker size. Obviously, the hash chain scheme performs worse than the q-composite scheme. Only when the number of hash chains is increased (which leads to higher memory consumption), the scheme becomes competitive.

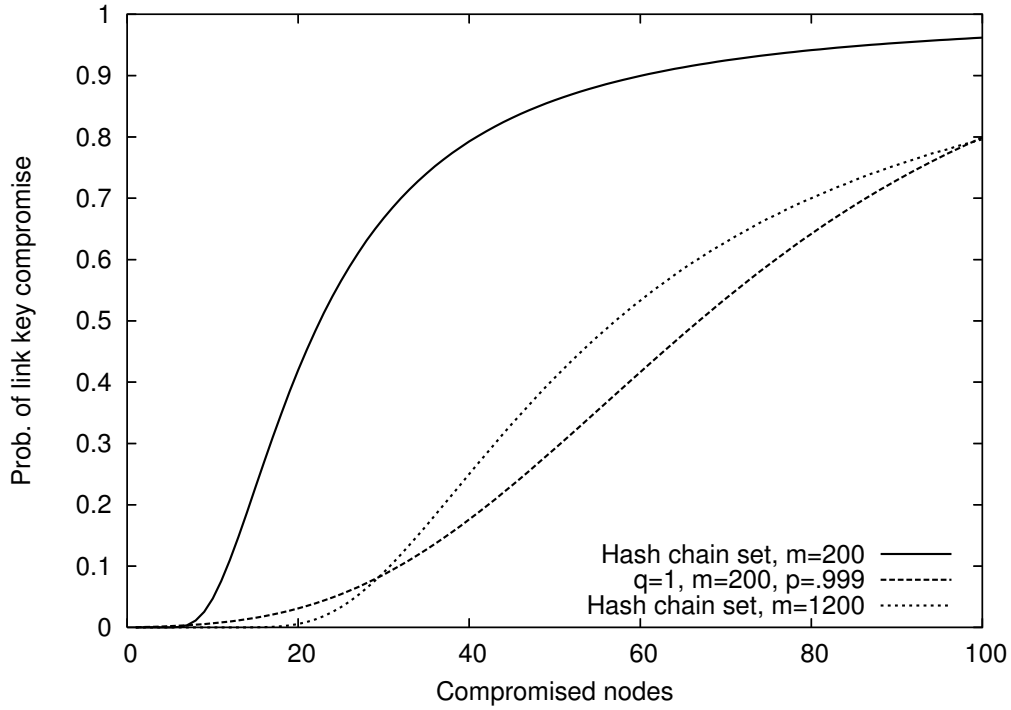


Figure 4.7: Resilience of hash chain sets for key agreement

This comparison shows that the q-composite scheme is in principle superior to the hash chain scheme and will be preferable over the latter in most cases. We conclude that the hash chain scheme may be considered for use in cases where full connectivity is required, and only attacks on a small scale are expected.

4.5 Strengthening Random Key Pre-Distribution

In this section, we reconsider the elements of the key pool of a random key pre-distribution scheme. In the basic and q-composite key pre-distribution schemes elements from the key pool were directly combined to form a link key between two parties. Now, we rather regard a key pool element as the seed of a hash chain. Instead of distributing these seeds, only derived values, i.e. elements

from the hash chains, are distributed to the nodes. After determining the common set of hash chains, these are used to agree on a link key between two nodes as described above. By combining both key agreement schemes in this way, we retain the advantages of both. This results in an increased resilience compared to either one of the schemes.

4.5.1 A Combined Approach

We generalize the key selection function F to include the additional step of transforming an element of the key pool into a hash chain value. First, we fix two seeds g_1 and g_2 that define two different contexts for the pseudo random sequence generator Ψ and the pseudo random number generator Φ. We will also need to choose the maximum length of key chains, T, and a hash function h for generating hash chains. Next, we introduce two new functions F_1 and F_2 that represent the different phases of the new scheme:

- $F_1(ID_u) = \Psi^{g_1}(ID_u, 1, S, m) = \langle v_1, \ldots, v_m \rangle$
- $F_2(ID_u, s, i) = h^{a_i}(s)$ for $\Phi^{g_2}(ID_u, 0, T-1, m) = \langle a_1, \ldots, a_m \rangle$

This leads to our new definition of the key selection function F:

$$F(ID_u) = \langle F_2(ID_u, \mathcal{K}[F_1(ID_u)[1]], 1), \ldots, F_2(ID_u, \mathcal{K}[F_1(ID_u)[m]], m) \rangle$$

Operationally, this means that, for each node u, the KDC first selects m keys from the key pool uniformly at random, using the (pseudo-) random sequence generator Ψ^{g_1}. On each of these root keys, it then applies the hash function h repeatedly, where the number of repetitions is determined by Φ^{g_2}, to obtain the final keys that go into the node's key ring.

The key establishment between two nodes now proceeds in two steps. First, the common set of indices into $\mathcal{K}$ is determined in the same manner as previously described. Then $\varphi = \Phi^{g_2}$ is used to determine the hash chain positions of the other node and the link key is established as described in section 4.4.

We will now show that although key establishment based on hash chains alone yields only small resilience, the resilience of a random key predistribution scheme is significantly improved through this combined approach. We assume that hash chains are of sufficient length such that equation (4.7) provides a valid approximation to hash chain key resilience.

4.5.2 Resilience

The strength of the adversay is now not only determined by the number of distinct root keys he obtains, but also on their hash chain positions. The more

keys from the same hash chain the adversary captures, the better are the chances that he can construct a larger part of that hash chain. We therefore consider the probability that the adversary has access to r keys on the same hash chain. Let x be the number of captured nodes. S denotes the size of the key pool, while m is the size of an individual node's key ring. The probability that the adversary obtains r values from a single, arbitrary hash chain, is determined as:

$$Pr[X = r] = \frac{\binom{x}{r}\binom{|S|-1}{m-1}^{r}\binom{|S|-1}{m}^{x-r}}{\binom{|S|}{m}^{x}} = \binom{x}{r} m^{r} \frac{(S-m)^{x-r}}{S^{x}} \tag{4.10}$$

Explanation: The r keys are located on the same hash chain but in different key rings, thus r out of the x compromised nodes are selected, which hold these keys. For each of these r nodes, the remaining $m-1$ keys are selected from the $|S|-1$ remaining chains in the key pool. The rest of the nodes are assigned all of their m keys from these $|S|-1$ chains. The denominator's term denotes the total number of ways to select the compromised key rings.

We use the probability given in equation (4.7) as (an approximation of) the probability that a chain key is compromised when faced with an adversary of certain strength. The overall probability that a chain key is compromised is thus determined by the sum over all possible attacker strengths:

$$p_{\mu} = \sum_{r=1}^{x} Pr[X = r] \left(1 - \frac{1}{\binom{r+2}{2}}\right) \tag{4.11}$$

We can now examine how the hash chain scheme collaborates with the q-composite random schemes. A link key is compromised if all partial link keys are compromised. For the q-composite scheme, equation (4.4) describes the probability that a link key between two uncompromised nodes is compromised. If we use hash chains for strengthening the components of the link keys, it is not sufficient that the adversary knows values from the hash chains that are used to establish the link key, but these values must also be suitable for deriving the respective partial link keys. Thus, both conditions have to be met in order to break a link key.

We obtain the final compromise probability by summing up over the possible numbers of root keys involved in creating a link key. Each single root key must be compromised, thus we consider p_{μ} to the power of i (the number of root keys). We also have to take into account the probability with which i root keys are involved. Therefore, the probability p_{ξ} for key compromise, which depends on the parameters S, m, x, q, is given as

$$p_{\xi} = \sum_{i=q}^{m} p_{\mu}^{i} \frac{Pr[i \text{ shared}]}{p_c} \tag{4.12}$$

$p_c = 0.33$	
Du-Deng	$\omega = 3, \tau = 1$
q-composite / HC	$q = 1, S = 100081$
MIOS	n/a
$p_c = 0.5$	
Du-Deng	$\omega = 15, \tau = 3$
q-composite / HC	$q = 1, S = 57918$
MIOS	n/a
$p_c = 0.98$	
Du-Deng	$\omega = 24, \tau = 8$
q-composite / HC	$q = 1, S = 10437$
MIOS	n/a

Table 4.1: Example parameters

Example 7. Fig. 4.8 illustrates various key agreement schemes in comparison. For all schemes, the same storage requirements are assumed, determined by the key ring size $m = 200$. Three different probabilities for sharing a common key between two nodes (connectivity) are shown. The parameters chosen are shown in Table 4.1.

MIOS (multiple ID-based one-way function scheme) is a scheme devised by Lee and Stinson [110] that is based on a combinatorial key assignment. DDHV is another scheme, described by Du, Deng, Han, and Varshney [60], which is a variant of Blom's key distribution scheme that has been adapted to the needs of wireless sensor networks. Finally, LN designates Liu and Ning's scheme [117], which relies on polynomials for key agreement.

4.6 Related Work

Key (pre-)distribution A general overview over key distribution schemes can be found in [122] (chapter 12). One of the earliest key pre-distribution schemes was described by Blom [24]. It allows each pair of hosts to derive a shared secret key. The scheme relies on a generator matrix G and a secret symmetric $k \times k$ matrix D. Each host U_i gets G and a secret key S_i, which is row i of the $n \times k$ matrix $S = (DG)^T$. Two hosts U_i and U_j are then able to compute their common key as element $K_{ij} = K_{ji}$ of the symmetric $n \times n$ matrix $K = (DG)^T G$.

The parameter k defines a security threshold. If less than k hosts are compromised, all pairwise keys are secure, since no knowledge about them is revealed to the adversary. However, as soon as k hosts are compromised, all keys are compromised. This scheme saves storage compared to full pairwise key distribution since k is usually chosen much smaller than the number of hosts.

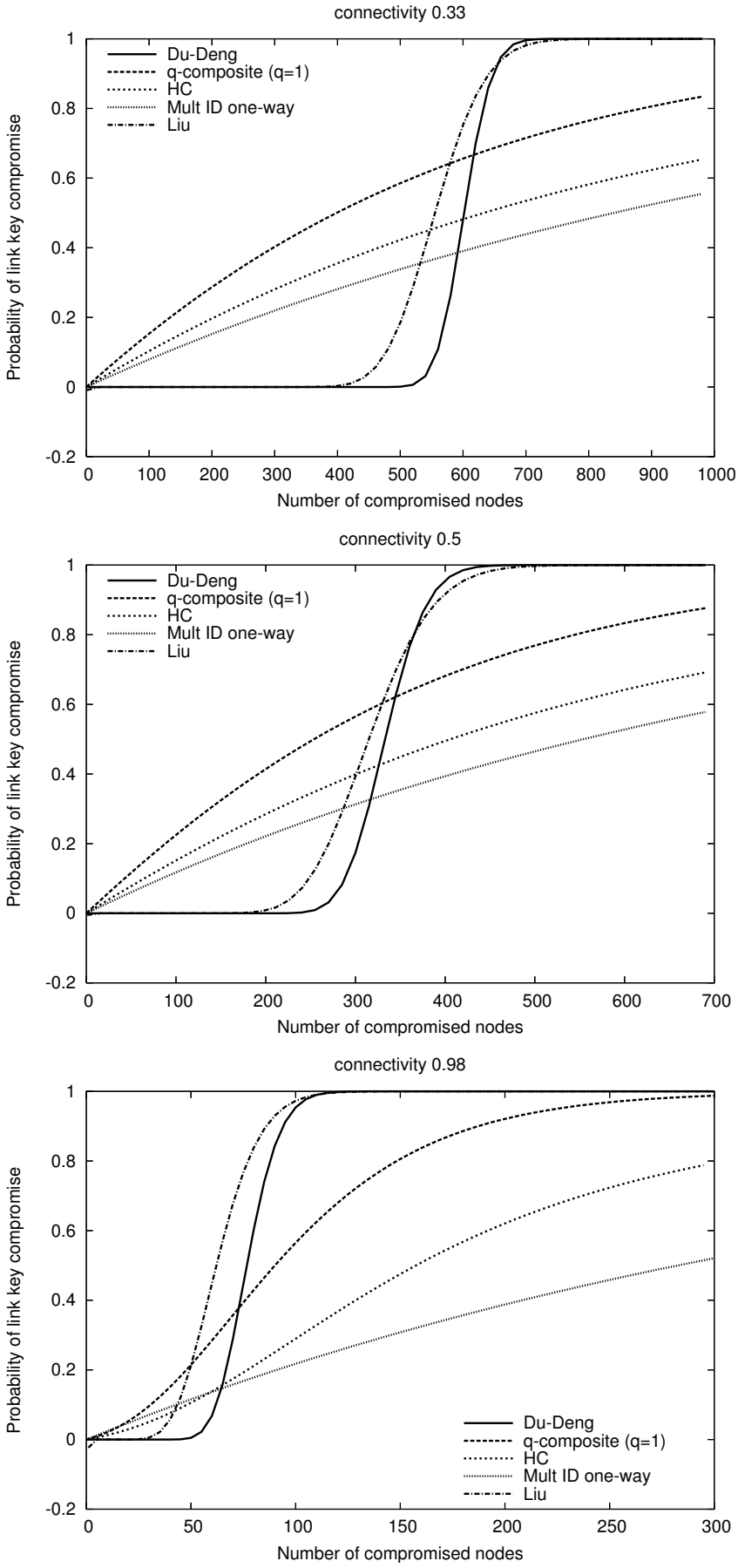


Figure 4.8: Comparison of different schemes ($m = 200$)

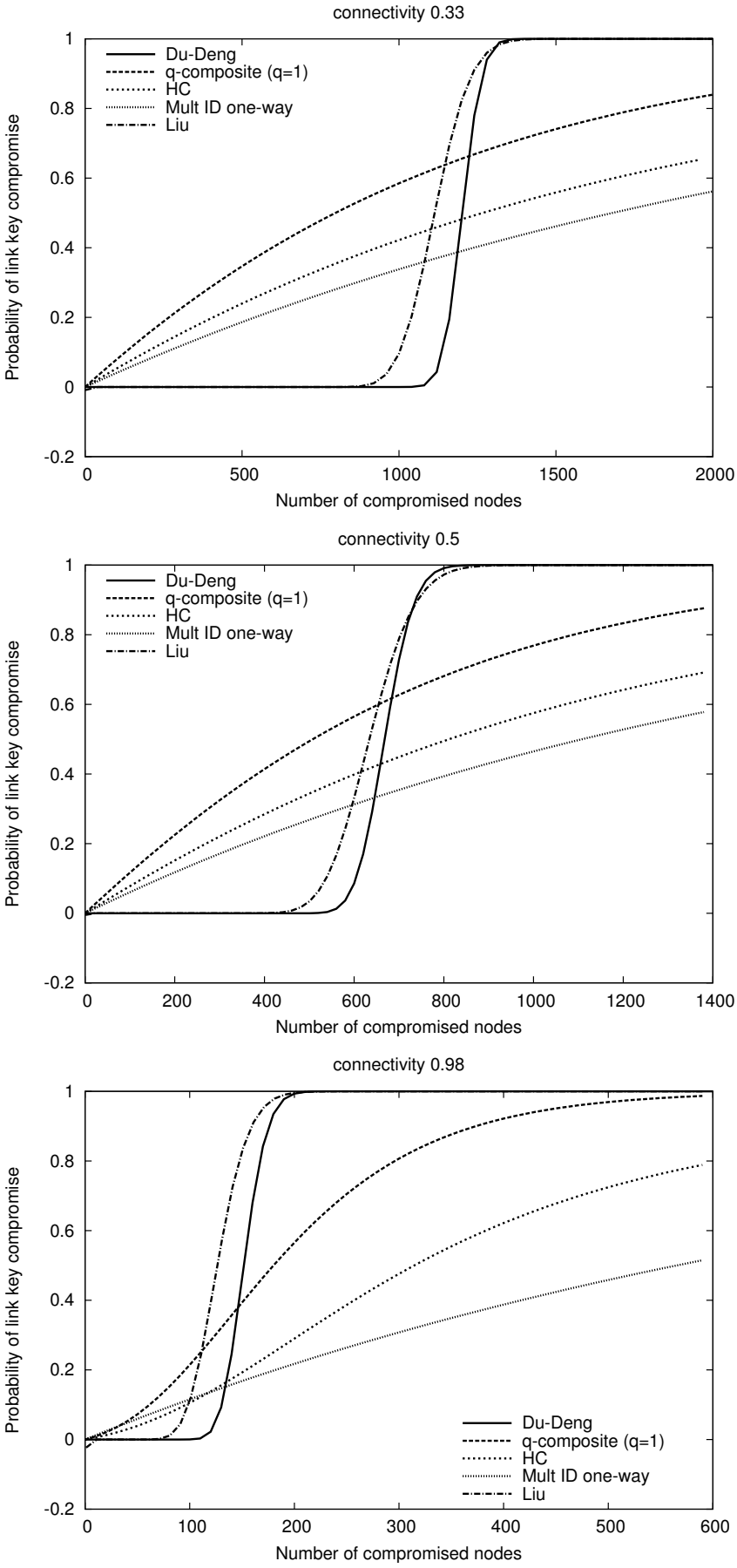


Figure 4.9: Comparison of different schemes ($m = 400$)

Based on Blom's scheme, an optimized version for wireless sensor networks has been proposed in [61]. Two modifications have been made: first, not only one key space but multiple spaces are being used (i.e. multiple Blom schemes are run in parallel), and second, the matrix G is not explicitly given but instead generated from a seed (which saves storage space). Out of the ω key spaces generated, the KDC chooses τ of them randomly for each node. Therefore, this approach is a combination of random key-predistribution with the deterministic Blom scheme. Two nodes can establish a shared key if they have at least one key space in common. This is the complementary probability of the event that two nodes share no key space. It is given by

$$1 - \frac{\binom{\omega}{\tau}\binom{\omega-\tau}{\tau}}{\binom{\omega}{\tau}^2} = 1 - \frac{((\omega-\tau)!)^2}{(\omega-2\tau)!\omega!}$$

The resilience of the scheme can be expressed as follows. Assuming x nodes have been compromised, the probability that a pairwise key of two uncompromised nodes is broken is

$$p_\kappa = \sum_{j=k+1}^{x} \binom{x}{j} \left(\frac{\tau}{\omega}\right)^j \left(1 - \frac{\tau}{\omega}\right)^{x-j}$$

Key agreement based on hash chains Hash chains were introduced by Lamport [107] for the verification of one-time passwords. First, a hash chain is generated from a seed. The seed is secretly given to the user, while the last element of the chain is given to the host where the user wants to log on. In order to log on, the user sends the value of the chain that hashes to the value that is stored in the host. If both values match, access is granted and the host stores the value just sent. This procedure continues until the seed has been used for logging on.

This method has the advantage that passwords are safe even if they are overheard by an enemeny, since each password is used only once. The one-way property of h ensures that verification of passwords is possible, while it is infeasible to derive the next password to log on from the current one.

Key agreement based on multiple hash chains has been described in [112]. The authors describe a more general variant first, where purely random values instead of hash-computed ones are distributed to the participants. This provides unconditional security, but has the disadvantage that large data sets have to be distributed. They then develop the version based on hash-computed values, which is more efficient but provides only computational security that is based on the infeasibility to inverse one-way functions.

The security of the scheme is discussed, as in our work, in terms of the probability with which an attacker can eavesdrop on a pairwise key. While they derive upper bounds for this probability, we provide an exact combinatorial derivation as well as an approximation that is well-suited for practical purposes.

Combination of random key pre-distribution and hash chains The combination of the hash-based scheme with random key pre-distribution schemes has been independently described by Ramkumar and Memon [148], who call this scheme "hashed random preloaded subsets" (HARPS). Using a similar methodical approach, they arrive at basically the same conclusions as us.

4.7 Summary

The tight resource contraints on nodes in wireles sensor networks, both in computational and storage terms, make traditional key management mechanisms based on public-key cryptography largely impractical. Nevertheless, security goals like resilience against eavesdropping, impersonation, and the creation of fake identities should be achieved. On the other hand, wireless sensor networks are constrained in several regards that make it possible to rely on certain assumptions when designing appropriate key management mechanisms. It can be assumed that all nodes in the network originate from the same administrative domain, the set of nodes is known before deployment and relatively static throughout the network's lifetime, the number of secure associations per node is small compared to the overall size of the network, and the contribution of a single node to the overall functionality of the network is also relatively small.

The key agreement schemes discussed in this chapter are designed to be applicable in wireless sensor networks as their resource consumption is very low. They require a moderate memory size, which is independent of the network size, and only computationally cheap operations, such as hash functions. The fact that nodes are within one administrative domain means that for authentication, nodes have to simply prove that they belong to this domain, which makes it possible to rely on a probabilistic authentication scheme. Pre-distributing key material before deployment is easily possible since it is known in advance which nodes are going to be deployed together. Under the assumed node capture attack model, it has to be assumed that a certain fraction of nodes is under control of the attacker. As a consequence, it has to be assumed that some of the reported data by sensor nodes is potentially manipulated, a fact that has to be dealt with on the application layer.

The security of these key agreement schemes is based on the fact that no central entity within the sensor network has complete knowledge of the basic key pool. With a small set of captured nodes, an attacker has only a very small probability of acquiring knowledge about the key material of other nodes. However, with an increasing number of captured nodes, this probability rises and allows the attacker to leverage the key material from her captured nodes for breaking into the communication of other nodes, i.e. eavesdrop on links or impersonate nodes.

We have discussed two basic key agreement schemes for wireless sensor networks. For the q-composite scheme, a set of keys is distributed to each node before deployment. Since all keys are drawn from a large, common pool of keys, it is likely that any pair of nodes has a certain number of keys in common. However, it is unlikely that any other pair of nodes shares the *same* keys. Whenever two nodes want to establish a common key, they can use their common keys to set up a *link key*, which is unique with a high probability. The second scheme is based on hash chains. A set of hash chains is created in advance. Each node is assigned a position on each of these chains. The one-way property of hash functions makes it easy to compute the chain values at positions in the chain that are located below a given position. Out of this, a key agreement scheme can be constructed that provides comparable resilience as the q-composite scheme, though at a higher memory complexity. Finally, we have devised a novel approach that combines both key agreement schemes and provides higher resilience than any one of them alone.

Key agreement is a prerequisite for secure communication between nodes, both for ensuring the confidentiality and the authenticity of messages. As we have discussed in Section 3.4, key agreement between remote nodes in a WSN incurs a significant overhead that is often not justified given the dominant transient communication patterns. Therefore, our goal is to leverage the existence of shared keys on a local level, i.e. within a limited neighbourhood of a node, in order to enable secure communication between remote nodes without incurring the usual overhead. In the next chapter, we show how to achieve secure communication over long distances by interleaved local message authentication. The basic scheme will be extended by using few long-range security relationships per node, which provides additional protection against certain attack patterns. The presented schemes protect the integrity of messages and thus are effective countermeasures against manipulation attacks.

Chapter 5

Multipath Communication

The use of multiple paths for transmitting a single message or a stream of messages has been studied for a long time. Multiple paths potentially provide increased bandwidth, improved resilience against network failures, and security against tampering and eavesdropping. However, they require additional effort in order to actually achieve these goals. In most cases, either node- or link-disjointness is required. For example, bandwidth can only be increased if no low-bandwidth links are shared between the used paths. Ensuring such a property increases the complexity and the overhead for a multipath routing protocol. This may be the reason that in practice, multipath routing is seldomly used.

In this chapter, we propose a multipath communication scheme for wireless sensor networks that provides security against node capture attacks. The scheme is designed to minimize path selection complexity, and it achieves its security goals through the spatial separation of paths.

5.1 Principles of Multipath Communication

5.1.1 Single vs. Multiple Paths

Some network topologies, such as stars or trees, admit only a single path between any two nodes. In such networks, multipath communication is obviously not possible. But even a simple topology like a (bidirectional) ring provides more than one path, and in complex mesh topologies, many paths between any pair of nodes may exist. In practice, redundant links are often introduced in order to provide resilience against link failures.

Finding the "optimal" path from a sender to a receiver is one of the common tasks of routing protocols. Usually a path is selected that performs best according to some cost metric, such as transmission delay or energy consumption. This single best path can then be used to transmit all messages from sender to receiver until an event occurs that breaks the path, such as a node or link fail-

ure, or a re-evaluation of the available paths shows that a different path would perform better.

The transmission of all data over a single path has certain advantages. For continuous data streams, such as digitized voice or other media streams, a single path provides predictable communication parameters like delay and bandwidth and thus avoids jitter. ATM networks have been designed to support a variety of applications running over the same packet-switching network simultaneously, including streaming applications [124]. In order to support continuous data transfer, a path is set up between endpoints before actual data transfer begins.

In networks with high failure probabilities, running a separate path setup routine introduces long delays when a failure occurs. Fast switching to a backup path avoids this, but requires the capability to quickly find an alternate path. Some routing schemes therefore concurrently maintain multiple paths, e.g. OSPF [128], or construct a detour around the failure location dynamically [68].

Using multiple paths simultaneously for transmitting either a single message or a data stream is often not advantageous. Even if multiple paths exist, they may overlap at some node. This node then becomes a bottleneck for data transmission, and the resilience of this transmission is only partially increased.

The real advantages of mulipath communication are only achieved with disjoint paths. Node-disjoint paths, and similarly link-disjoint paths, can provide increased bandwidth and better resilience, among other advantages, than a single path. Since we are concerned with wireless sensor networks, which operate on a shared medium, we are mainly interested in node disjoint paths.

These considerations demonstrate that the transition from using single paths to multiple paths is not as sharp as one might expect. In the literature, "multipath" is often associated with using single paths one after another. Some of the benefits, discussed in the next section, can be achieved through such schemes, while others are best achieved by using multiple paths at the same time.

5.1.2 Advantages of Multiple Paths

There are some advantages when using multiple paths for routing messages between endpoints, which can best be used for data that does not have strict timing requirements:

- By splitting up a message stream into several parts and transmitting every part over a distinct path, the **bandwidth** may be effectively increased. Note that despite the fact that paths may be partially overlapping, they

could still increase the available bandwidth. For example, they could share some high-capacity links while using disjoint low-capacity links.

- Similarly, the **load** for transmitting and relaying messages can be distributed over a larger set of nodes.

- The **reliability** of message transmission is increased if multiple copies of a message are transmitted over independent paths. As long as one path delivers the message, it will be received, thereby ensuring **availability**.

- The **integrity** of a message is supported if multiple copies of a message are transmitted. If a message is sent over n independent paths, its integrity is protected as long as less than $n/2$ paths are compromised. Under this assumption, the receiving node can determine the original content of a message by a majority decision. If $\lfloor n/2 \rfloor + 1$ identical copies of a message are received, it is clear that these match the original message posted.

  An attack on the integrity and authenticity of a message can still be detected if at least two disjoint paths are being used, and at least one of them is not compromised. If the copies of the message received over all paths do not match, it is obvious that there is an active attacker.

- Using a threshold scheme [165], a message is encoded into n parts in such a way that a party that obtains at least t out of the n parts will be able to reconstruct the original message. On the other hand, if less than t parts are known, no information about the message can be inferred. Such a scheme protects the **confidentiality** of a message if every part of the message is transmitted over a separate path.

- If failures or attacks are highly linked to locations, **spatial separation** of paths can provide additional reliability and security.

The disjointness of paths is crucial to the materialization of most of these benefits. However, partially disjoint paths may often be practically helpful as well.

5.1.3 Path Setup

In general, the path selection criteria for single paths also hold for multiple paths. At best, multiple paths should be usable at minimal total cost. However, in some cases the cost for a single end-to-end connection is not the most important aspect to consider. Other criteria such as path setup cost or spatial

separation may prove more relevant when a trade-off between cost and security is to be found, for example.

Most algorithms for constructing multiple paths treat every pair of sender and receiver separately. Since there is substantial overhead required for constructing and maintaining multiple paths, this does only pay off for longer-lasting communication relationships between nodes. In WSNs, there are usually many node pairs exchanging messages at the same time, and communication relationships are mostly transient. Conventional multipath schemes do not amortize set-up overhead over all node pairs, and are inefficient for short-term connections.

There are many possible metrics that could be used for evaluating and selecting paths. Especially in dynamic (e.g. mobile) networks, it is important that these metrics can be efficiently evaluated. There is a trade-off between the overhead induced by the metric evaluation and the penalty that must be paid for using non-optimal paths. Our attacker model suggests that spatially separated paths are desirable as such paths can circumvent areas that are under attack. For efficiency reasons, we would like to amortize the set-up and maintenance overhead over many communication relationships. Current approaches mainly emphasize the set-up of an alternate path in case the currently used path fails. Usually, the replacement path is very close to the replaced path, which means that most nodes are shared between both. In terms of our attacker model, this means that if one path is being compromised, the replacement path is probably also compromised.

The idea for setting up multiple paths we present in this work is based on the concept of spanning trees, which is fundamental for shortest path construction (cf. Dijkstra's algorithm, described in [44]) and multicast. A multicast tree can be shared by all senders that are located on the tree, thereby reducing the overhead compared to a separate tree for each sender [12]. We use the same principles for end-to-end communication. Messages are routed based on the paths determined by spanning trees, and all senders use the same set of spanning trees. This leads to suboptimal path lengths, but using different trees simultaneously provides spatially separated paths.

5.2 Routing on Spanning Trees

5.2.1 The Basic Scheme

We propose a multipath communication scheme that adequately addresses the requirements of wireless sensor networks. It uses spanning trees as routing

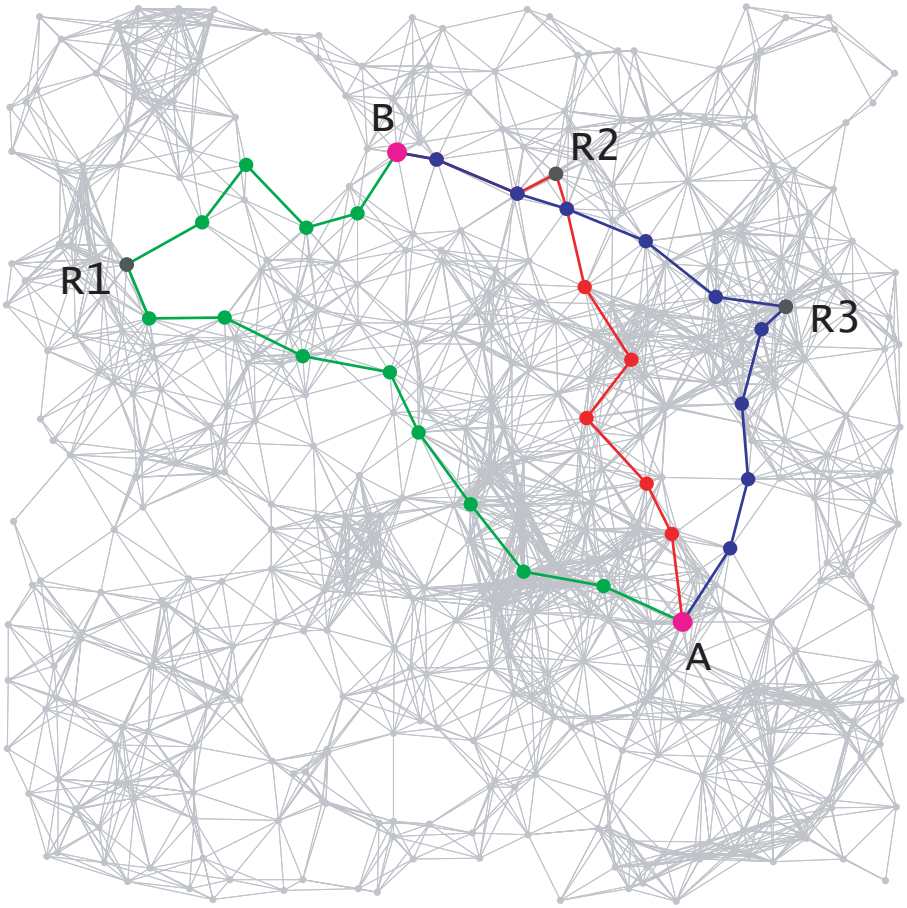


Figure 5.1: Spanning tree multiple path routing

structures. Each spanning tree determines a path between any two nodes, and multiple such paths are used for message exchange. We show that, similar to *Canvas*, this scheme provides an adequate level of integrity protection for wireless sensor networks.

The fundamental idea of this approach is to set up multiple spanning trees and use them to create paths between pairs of nodes, which are node-disjoint to a certain extent. An example is shown in Figure 5.1. Three spanning trees with roots at R_1, R_2, R_3 determine three paths between A and B.

Obviously, pairs of paths are not necessarily node-disjoint. In the example, only pairs AR_1B, AR_2B and AR_1B, AR_3, B are node-disjoint, while AR_2B, AR_3B overlap. Despite the overlap, path AR_3B does not degrade the security of the set-up but adds to it. The overlap occurs close to B, so the principle of locality applies: In order to exploit the overlap, the attacker must be active in the vicinity of B. If the attacker is only active further away from B, three locations have to be attacked in order to compromise a message exchange. This significantly increases the burden on the attacker.

Setting up a spanning tree is a costly procedure, as all nodes are involved.

However, a tree provides a routing path for each pair of nodes. In total, the spanning tree requires less effort than maintaining pairwise paths individually.

As for the price to pay, the length of a tree path is usually longer than the shortest path between any two nodes. We therefore prefer constructions for spanning trees that minimize their depth. Using optimal (shallow) spanning trees, we can expect the average path length to be approximatly twice the length of a direct path.

5.2.2 Spanning Tree Construction

We describe a simple distributed spanning tree construction algorithm. This algorithm is not optimized to take parameters like link quality into account. However, an improved version could be designed based on existing approaches such as [9].

One node initiates the construction of a tree and will become its root. New spanning trees are constructed according to a certain schedule, or they may be constructed "spontaneously", i.e. any node can initiate a tree construction when it sees fit. When using tree-based routing, the message load is unequally distributed. Nodes closer to the root of a tree are likely to be burdened with forwarding more messages than nodes closer to the leaves, thus a tree should have limited lifetime. When a tree's lifetime expires, the associated information can be deleted. In order to avoid synchronisation issues between nodes, we assume that all nodes have lightly synchronised clocks and we assume that when the lifetime of a tree t expires, a node would cease to send own messages along that tree, but still relay messages from other nodes for a certain period.

The algorithm for constructing and routing on a tree is semi-formally given as Algorithm 2. It is defined in an event-driven manner, i.e. each node continously runs this algorithm and reacts to the described events. Construction starts with a node sending a INIT message to its neighbours, who will forward this message to their own neighbours, and so forth. Each node then waits for its neighbours to respond with either a CHILD or a BOUND message. After all neighbours have answered, the node itself responds to the neighbours from where it has received the INIT message. The response contains an address space A that describes all the addresses being reachable within its subtree. The actual type for addresses will be discussed later.

Note that the algorithm does not explicitly enforce shallow trees. For simplicity, we assume that this property of trees emerges from the fact that a message on a direct link is always faster than on an indirect link over multiple hops. As this assumption may not hold in practice, the algorithm might have to

be adapted to ensure that a node is connected to the parent that is closest to the root.

Each node maintains a data structure T for each spanning tree, which holds the information required for routing. For a tree with identifier t, an entry $T[t]$ is maintained. Such an entry has the following components:

- *parent* – The parent node within this tree (where the root is its own parent).
- *reach* – A mapping from child identifiers to their covered address space. For a child c, $T[t]$.reach(c) is a representation of the address space reachable through c. If a message is destined to address d and $d \in T[t]$.reach(c), node c may either be qualified to handle the message itself, or deliver it to a node closer to d.

It is well possible that, during tree construction, overlaps between children occur and part of the address space is covered by multiple children. This is due to the (possible) fuzziness of the $\uplus$ operator, which merges the address spaces of a node and its children. We will discuss the inefficiencies that may arise when instantiating the routing framework with concrete schemes.

5.2.3 Addressing on Spanning Trees

A critical aspect of tree-based routing is which type of addressing should be used. We discuss a canonical addressing scheme, which is useful only for a limited range of applications, and a more generally applicable geographical addressing scheme.

Definitions

Generally, we can consider an abstract address space $\mathcal{D}$. With each node u, a set

$$cover_u \subset \mathcal{D}$$

is associated that comprises the set of addresses under which node u can be reached. Viewed from a different perspective, if for a target address $d \in \mathcal{D}$ of a message, $d \in cover_u$, node u will be eligible to receive and process the message. A node may be part of multiple trees and for every tree, the above predicate should yield a different part of the address space. For a specific tree t, we will therefore write $cover_u^{(t)}$.

We require an operation for combining addresses, which we will denote as $\uplus$. Two address covers $D_1, D_2 \subset \mathcal{D}$ combined using this operation satisfy the

Algorithm 2 Tree construction and routing algorithm, running on node u

on **construct spanning tree:**
    /* This event is spontaneously fired */
    create unique tree-identifier t
    create tree record $T[t]$
    set $T[t]$.parent := u /* the root is its own parent */
    send INIT$\langle t, u\rangle$ to all neighbours

on **receive message** INIT$\langle t, v\rangle$**:**
    if $T[t]$ already exists **then**
        respond with message BOUND$\langle t, u\rangle$
    else
        create tree record $T[t]$
        set $T[t]$.parent := v
        send INIT$\langle t, u\rangle$ to all neighbours except v
    end if

on **receive message** CHILD$\langle t, v, A\rangle$**:**
    set $T[t]$.reach(v) := A

on **receive message** BOUND$\langle t, v\rangle$**:**
    note response of v

on **all neighbours have responded to** INIT **(or timeout):**
    construct $A := \text{cover}_u^{(t)} \uplus \biguplus_{v \in \text{dom}(T[t].reach)} T[t].\text{reach}(v)$
    send CHILD$\langle t, u, A\rangle$ to $T[t]$.parent

on **data** D **ready to be delivered to destination** d**:**
    for $d \in \text{dom}(T)$ **do**
        send DATA$\langle t, u, d, D\rangle$ to self
    end for

on **receive message** DATA$\langle t, s, d, p\rangle$ **from** j**:**
    if $d \in \text{cover}_u^{(t)}$ **then**
        /* handle message */
    else if $\exists c : c \neq v \wedge d \in T[t].reach(c)$ **then**
        forward DATA$\langle t, s, d, p\rangle$ to c
    else if $T[t].parent \notin \{u, v\}$ **then**
        forward DATA$\langle t, s, d, p\rangle$ to $T[t].parent$
    else
        /* invalid destination */
    end if

on **lifetime of tree** t **exceeded:**
    delete t from T

Algorithm 3 Variant: Forward a message to all covering children

```
1: on receive message DATA⟨t,s,d,p⟩ from j:
2:     if d ∈ cover_u^(t) then
3:         /* handle message */
4:     end if
5:     ∀c : c ≠ v ∧ d ∈ T[t].reach(c): forward DATA⟨t,s,d,p⟩ to c
6:     if T[t].parent ∉ {u,v} then
7:         forward DATA⟨t,s,d,p⟩ to T[t].parent
8:     end if
9:
```

condition

$$D_1 \cup D_2 \subseteq D_1 \uplus D_2 \ ,$$

i.e. all addresses contained in either D_1 or D_2 are also contained in the resulting set. For some address spaces, it may be impractical to represent subsets of that space both efficiently and accurately. We therefore admit some "fuzziness" in the union operation, i.e. we allow

$$D_1 \uplus D_2 \setminus (D_1 \cup D_2) \neq \emptyset \ .$$

This means that additional addresses may be in the combined covers that are not covered by either of the components.

In the following, we show how to instantiate this general scheme with concrete addressing schemes.

Tree Addressing

Within a tree, there is a canonical addressing scheme, assuming a static order on the children of each node. Starting from the root, each node can be given a unique *index address*

$$i_0.i_1.i_2.\ \ldots\ .i_h$$

where $i_j \in \mathbb{N}$ denotes the index of a child of the current node. If two such addresses have the same prefix, they are located in the same subtree. Note that the root's address is the empty index sequence ε. Addressing nodes in this way is efficient since this scheme requires no further state information except for the static order on children.

Index addressing has the drawback that the same node has a different address in every tree, therefore an address is only valid for the lifetime of the tree. Also, the sender has to know the index address of the destination node in order to send a message. Getting the current index address of the destination to the sender requires additional overhead.

An application scenario where index addresses are useful is aggregation. Assume that multiple spanning trees are used for routing. Let one node be the aggregation point, collecting messages with sensor data from a large number of other nodes. The aggregating node has a different address in each spanning tree (note that the aggregator needs not necessarily be the root of any such tree). In order to send their sensor data messages to the aggregator, the sensing nodes have to know the tree addresses of the aggregator. This can be done by a single broadcast message. Compared to the overall task, this constitutes a moderate overhead, so using tree addressing is feasible in such a scenario.

The instantiation of the abstract addressing scheme is as follows:

- $cover_u^{(t)}$ represents the index address of node u with regard to a specific tree t. This means, a node *covers* only itself.
- $a \uplus b := a \cup b$, i.e. the union operator is simply the set union. The root of a subtree covers all addresses of its children (and the nodes below them). In fact, representing $T[t].\text{reach}(v)$ is straightforward: it is simply node v's tree address as this is the prefix of the addresses of all of its children.

Geographic Addressing

In a WSN, often geometric addressing is desired, e.g. for sending a message to a certain location (geographic anycast). Tree addresses as discussed before require the address of a specific node to be known to the sender, which is often not desirable. Geographic addressing is not supported by such tree addresses. Nevertheless, we would like to be able to use spanning trees for routing messages based on geographic addressing as well. This can be achieved in the following way.

Each node is assigned a geographic area for which it directly accepts messages. This is the "cover area" of a node. The shape of such an area may be arbitrary, but since we need to represent such shapes, they would be usually restricted to simple shapes such as rectangles or circles. The cover area may or may not be correlated to the radio range of a node, or its sensoric or actuatory reach.

When a tree is established, all nodes report the area they cover directly or indirectly, i.e. including the cover areas of their children, to their parent nodes. The representation of this cover area poses a problem, since an exact representation would require a large data structure. In the worst case, one entry per node has to be kept. However, we would like to keep the overhead to a minimum, so a different solution is required. One possibility is to give up exactness and choose a less accurate but also less costly representation.

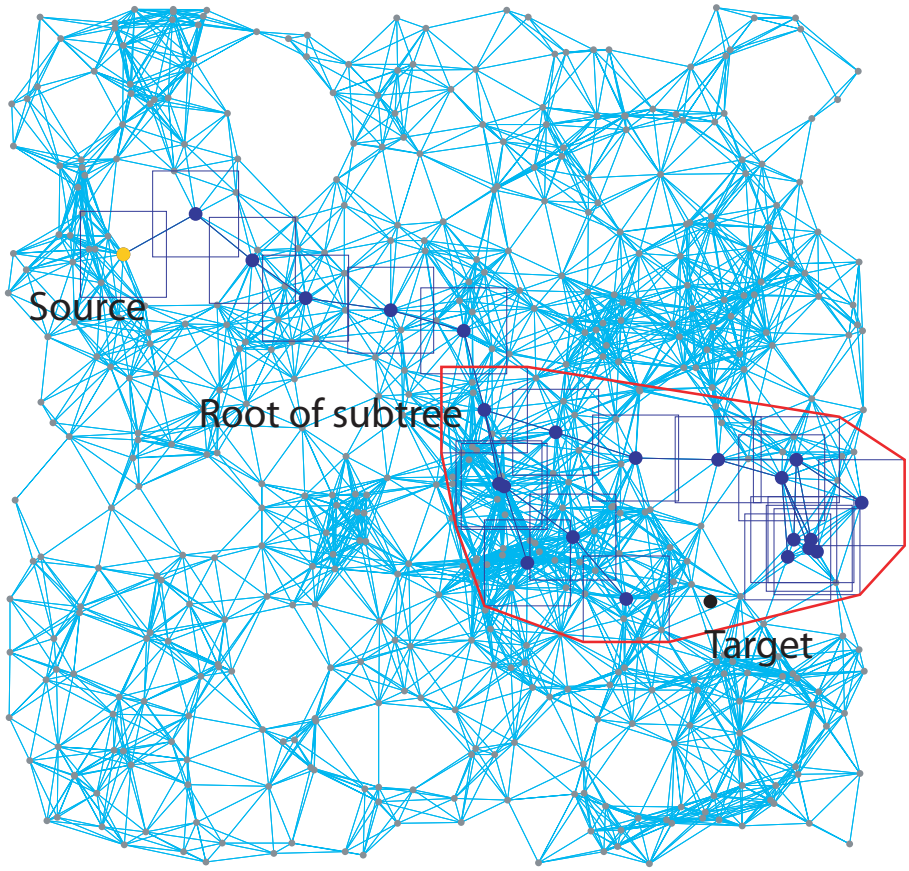


Figure 5.2: Target address contained in the convex hull of the cover area, but not in the cover area itself

Here, we propose to use the convex hull of a set of regions to represent the sum of the regions. Convex hulls have some desirable properties. First, a convex hull can be represented by very few points compared to the point set it represents. Second, if the point set is extended, the new convex hull can be solely computed from the existing points on the hull and the new point set. No information about the points contained in the convex hull is required. A third property is efficient computability using Graham's algorithm (cf. [44]), which has $O(n \log n)$ time complexity in the number of nodes.

Using convex hulls for representing the area covered by a set of nodes in a subtree has the disadvantage that the convex hull may contain areas that are in fact not covered by any nodes in the subtree. This means that although an address, i.e. a geographic location, may be purportedly reachable through a subtree, since it is contained within the convex hull covered by the subtree, there is in fact no node in the subtree that actually covers this location. Hence, such a message will be routed through that subtree without ever reaching a node that actually covers the addressed location.

Such a situation is depicted in Figure 5.2. The boxes illustrate the cover areas of the nodes, the union of these boxes is the total cover area of the subtree.

Obviously, the target location is contained in the convex hull of the subtree's cover area, but none of the nodes actually covers the target location. Therefore, the message cannot be delivered within this subtree.

This problem can be mitigated by forwarding a message not only to one child node that is supposed to cover the target location, but to all of them. This variant of the algorithm is shown as Algorithm 3. Here, we make sure that the message reaches all of the eligible nodes by forwarding it up the tree as well. This increases the overall load but makes sure that the message is eventually delivered.

The fact that a location is usually covered by multiple nodes and many of them may receive a message addressed to this location may lead to problems in certain applications, where only one node should actually process a message. In such a case, the receivers of a message, which are located in close proximity, have to coordinate themselves.

For geographic addressing, we instantiate the addressing scheme as follows:

- $cover_i^{(t)}$ represents the covered area of node i, which we set to a square with side length w and the node i at its center.
- The operator $a \uplus b$ yields the convex hull of a and b, where a and b are point sets representing polygons.

5.2.4 Message Forwarding

Hierarchichal tree addressing, as discussed above, allows the direct forwarding of a message to a specific node. At each node, it is immediately clear if the message should be forwarded towards the root of the tree, or to a certain child node.

Geographic addressing is more "fuzzy". Although the cover area of a node may contain the target address of a message, there might be no node among the children and subchildren of that node that actually covers this location. Therefore, even if a message is forwarded to that node, the message may not reach its target. This deficiency may be overcome by an additional local routing scheme that is initiated by the node holding the message without any child nodes to which the message can be delivered. However, this node may still be far away from the target location, and the advantages of the tree routing may be lost.

Also in contrast to tree addressing, multiple nodes may qualify as receivers when geographic addressing is used, which we denote as *geographic anycast*. Such a situation needs to be resolved on a higher system layer, for example

by the application. However, this leads to a higher overhead as messages are unnecessarily forwarded to certain nodes. Nevertheless, forwarding a message to all eligible child nodes mitigates the coverage problem described in the previous paragraph. Some of the redundant messages will be dropped anyway further down in the tree, but the message will actually be delivered to a qualified target node.

5.2.5 Choice of Tree Paths

At any given time, there exist S active spanning trees in the network. A sender can freely choose the trees that it wants to use for routing. In order to minimize the probability of intersections, the sender can select trees whose roots are located on different sides of the line between itself and the target location. This will probably only lead to intersections that are close to the end-points.

5.3 Properties of Tree Paths

We are interested in routing messages over multiple paths that are spatially separated and mutually node-disjoint. In addition, the overhead incurred in terms of message load and delay should be acceptable. We now examine the characteristics of tree paths with regard to these criteria.

5.3.1 Spatial Separation

Spatial separation describes a property of pairs of paths. Two paths with a high degree of spatial separation will have minimal exposure to a location-constrained attacker. We will not go beyond this informal definition of the term.

Spatial separation can be measured in several ways. We will assume any measure that is negatively correlated with the impact of a location-constrained attacker. A high spatial separation would correspond to a minimal attack surface to a location-constrained attacker.

The spatial separation of paths that share the same source and target nodes is naturally limited. A location-constrained attacker, even if he would be constrained to capturing a single node, would be able to compromise both paths if he captured one of the end-points.

Tree paths are not necessarily disjoint, thus they may share more than just the end-points. This increases their vulnerability to location-constrained attacks. However, in many cases they provide a good spatial separation even if

they are not disjoint. For many nodes on one path, the distance to the closest node on the other path is quite high.

5.3.2 Path Disjointness

On first sight, tree paths seem inadequate for providing spatially separated paths since there is a non-zero probability that two tree paths intersect. Thus, it may happen that whenever a message that has been split into multiple shares is sent over a set of tree paths, there will be at least one node that sees multiple shares of the message. This is a violation of the assumptions on which the secret sharing mechanism is based. The overall security therefore depends on the probability with which intersections occur.

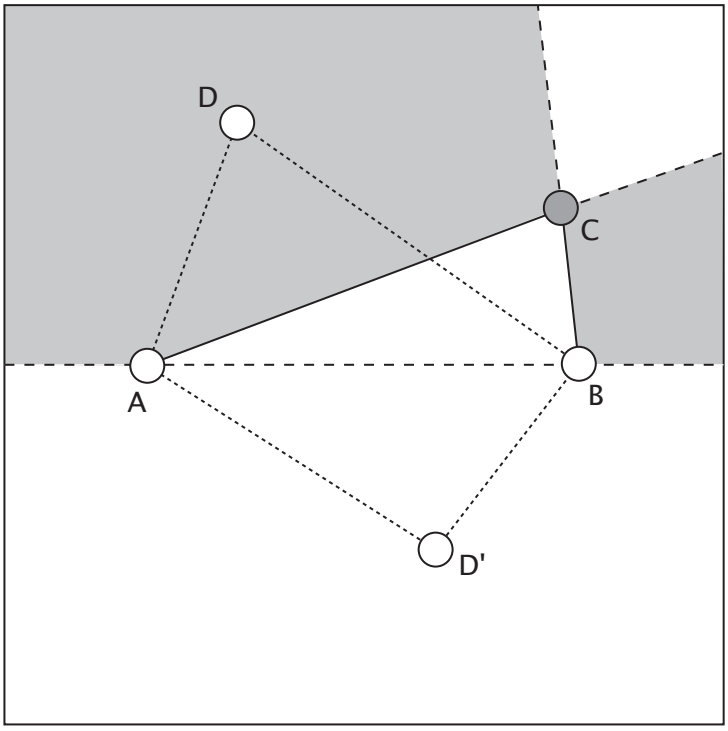


Figure 5.3: Intersection and intersection-free zones

We first consider an idealized geometric model of the network. Nodes are represented by points in the Euclidean plane. Figure 5.3 shows an example. The two communicating nodes, A and B, form triangles with the tree roots C and D (respectively, D'). Two tree paths intersect in at most one point.[1] In this example, the tree paths ACB and ADB intersect, while ACB and $AD'B$ do not. Of course, whether an intersection occurs or not depends on the relative position of the two tree roots. Given the position of C, if D falls within the non-shaded area, both triangles are intersection-free, while D placed in the shaded

[1] Note that we simplify here and ignore bordercases such as $C = D$ or D falling onto one of the lines AC or BC.

area would yield an intersection.

When the deployment area is a square, we can roughly approximate the probability with which an intersection between two trees occurs in the following way. The "grey" zone occurs only on the side of line AB on which C is located. The closer C is to one node, the smaller the grey zone on this node's side will be, but it will be larger on the other's (this can be seen in Figure 5.3, where the shaded area close to B is much smaller than that close to A). In the extreme case, the grey zone will fill out the whole upper area almost completely. The other extreme is C being close to the line AB, and placed between A and B; in this case, the grey zone will be rather small in total. For a rough estimation, we may assume that one half of the upper area will be grey. In total, this means that in about one quarter of all pairs of tree paths, there will be an intersection.

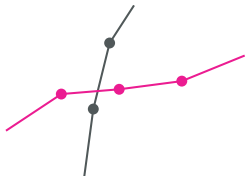


Figure 5.4: Paths crossing each other without an intersection node

This estimate is based on an ideal, continuous model. A real WSN is a discrete system, communication paths are not straight lines, and their density is limited. This leads to a number of differences to the ideal model:

- Paths that intersect in the model do not necessarily intersect in a real deployment. An example is shown in Figure 5.4. Note that if a Gabriel graph would be used, which is not necessary for tree routing, a situation as in this example cannot occur and the paths will definitely intersect.

- In the real world, it is likely that paths intersect in a location close to the communication endpoints since paths are closer to each other and, since there are only finitely many nodes to choose from, it is more likely that they have one or more nodes in common.

- An intersection need not be restricted to one common node. It is possible that two paths have a segment in common, especially close to the endpoints. An example is shown in figure 5.6.

These considerations lead us to the conclusion that the fraction of tree paths that intersect should be roughly above one-quarter, but below one-half. In order to validate this assessment, we performed a simulation experiment. Simulation runs were executed for ten different networks, 30 spanning tree pairs for each

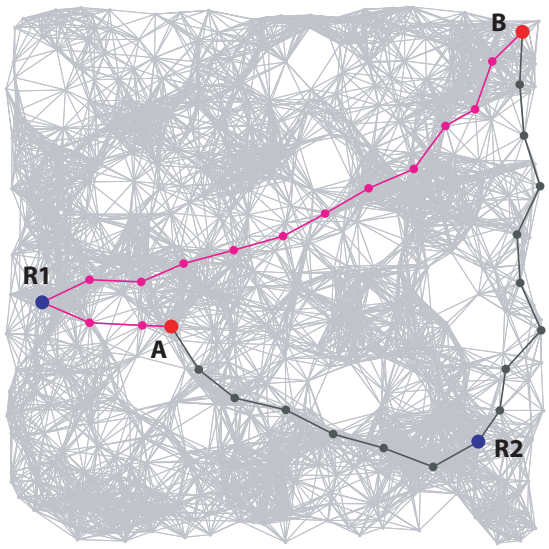


Figure 5.5: Intersection-free (disjoint) tree paths

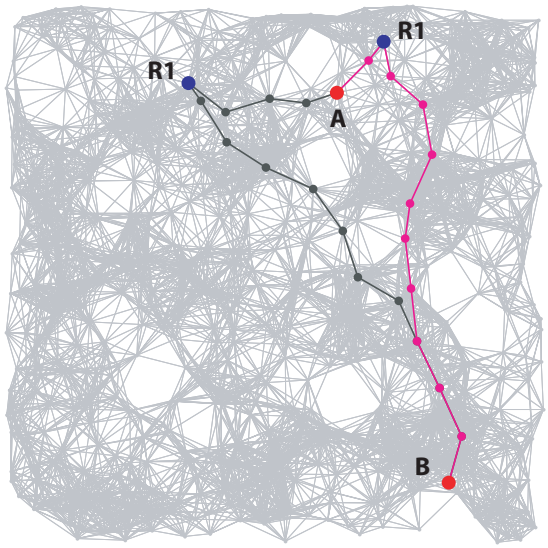


Figure 5.6: Tree paths with a common subpath

network, and 100 node-to-node connections for each pair of trees, i.e. 30 000 samples were taken in total. Each network consisted of $N = 500$ nodes, the communication range was set to $R = 100$ units, and the deployment area was the standard square with a side length of 1000 units. The simulation experiment did not take a specific addressing scheme into account but considered node-to-node connections.

We define the *distance of an intersection* of two tree paths as follows:

$$\delta_I = \max_{\forall X} \min\{\delta(X,A), \delta(X,B)\}$$

Informally, this means that the shared node that is furthest away from both end-points determines the distance of the intersection itself. A small value means that an intersection occurs close to an endpoint, which is advantageous for the network due to the locality principle. For the same reason, a higher value provides an advantage for the attacker.

Figure 5.7 shows the distribution of the intersection distance for our simulation experiment. The disjoint tree paths are not included in this diagram; they account for approximately 40% of all instances. Another 18% have an intersection distance of one, which means that the (only) shared node is a neighbour of one of the end-points. Therefore, about 58% of all tree paths are virtually disjoint. Another positive aspect is that very great distances are very rare, which means that the attacker can exploit intersections only in very few cases.

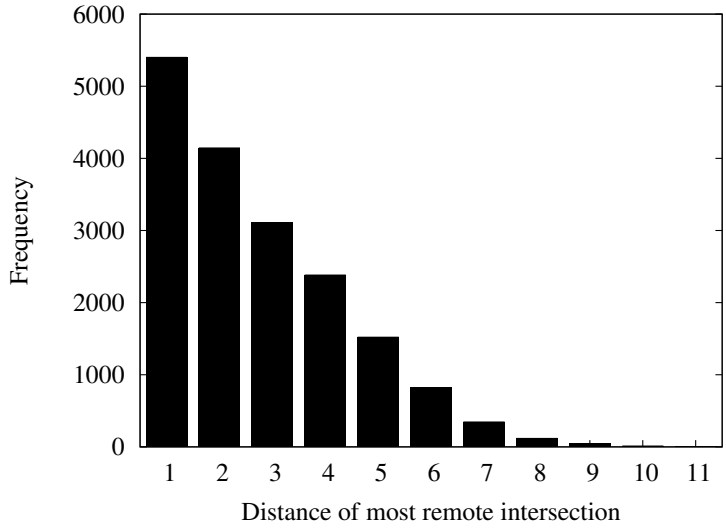


Figure 5.7: Distribution of the intersection distance

Another interesting measure is the number of nodes at which two tree paths intersect. The higher this number, the easier it is for an attacker to exploit an intersection. Figure 5.8 shows the distribution obtained from the simulation experiment. Here, zero is included, illustrating the fraction of intersection-free

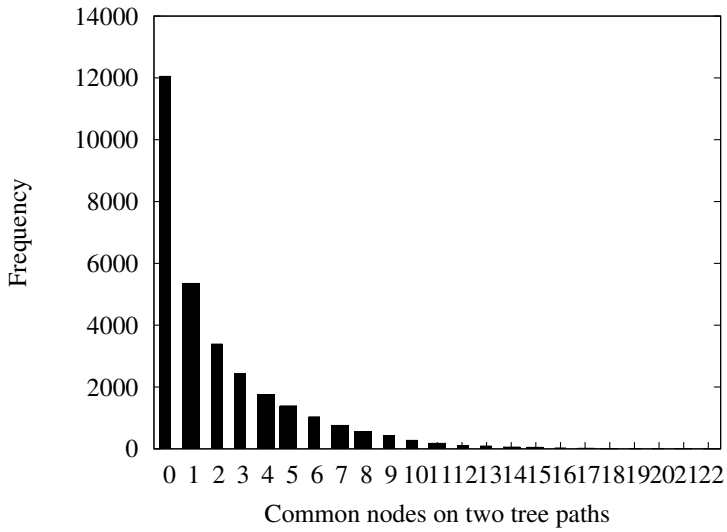


Figure 5.8: Distribution of common nodes between tree paths

pairs of tree paths. As expected, due to the physical separation of tree paths, the number of tree pairs that share a large number of nodes decreases rapidly.

5.3.3 Traffic Overhead

Tree paths are not designed to minimize the number of hops necessary to deliver a message. They are intended to provide for disjoint and spatially separated paths. In addition, the use of tree paths in conjunction with a geographical addressing scheme leads to failures in delivering messages. Thus, the intended gain in security must be paid for in terms of a higher hop count and a delivery rate below one.

Let L be the average length (hop distance) of shortest paths in the network. Using a tree path, a message is routed first to the root, and then on to the target. If all messages are routed through the root, this yields a path length of approximately $L_m \approx 2L$. However, a few messages do not need to go through the root as the source and the destination node are both contained in the same subtree. These connections lower the average tree path length by a small amount. From the same simulation experiment as above we also obtain the distribution of the path lengths. Figure 5.9 shows the distribution of the shortest path lengths, i.e. the shortest connections between source and destination with no tree involved. The average path length is 6.9. Figure 5.10 illustrates the distribution of the tree path lengths; this yields an average of 11.9, i.e. $L_m \approx 1.72L$. This experiment shows that the expected path length using tree routing is well above the best achievable path length but would still be acceptable in many cases.

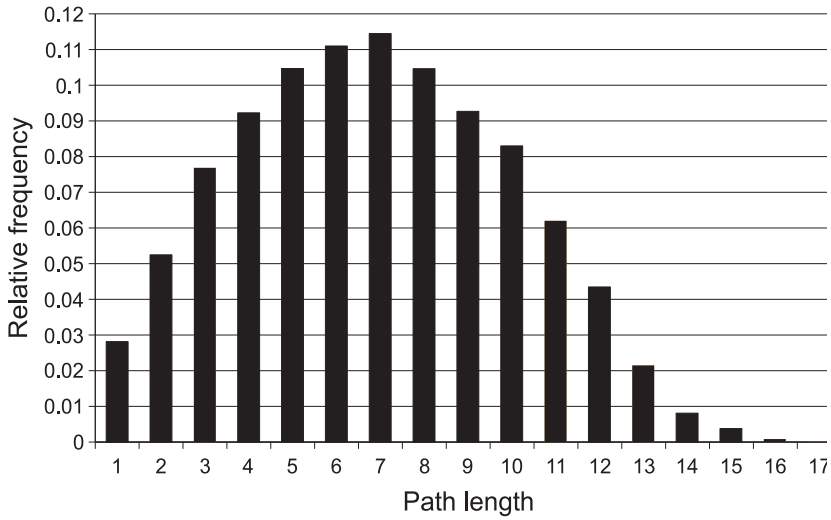


Figure 5.9: Distribution of shortest path lengths

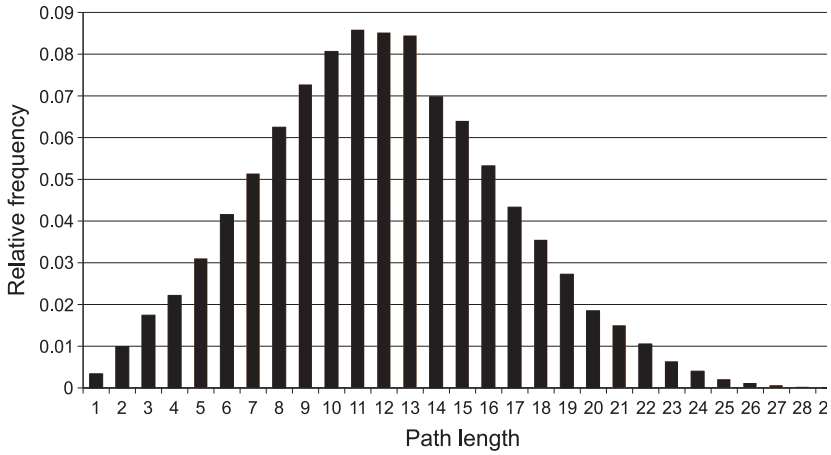


Figure 5.10: Distribution of tree path lengths

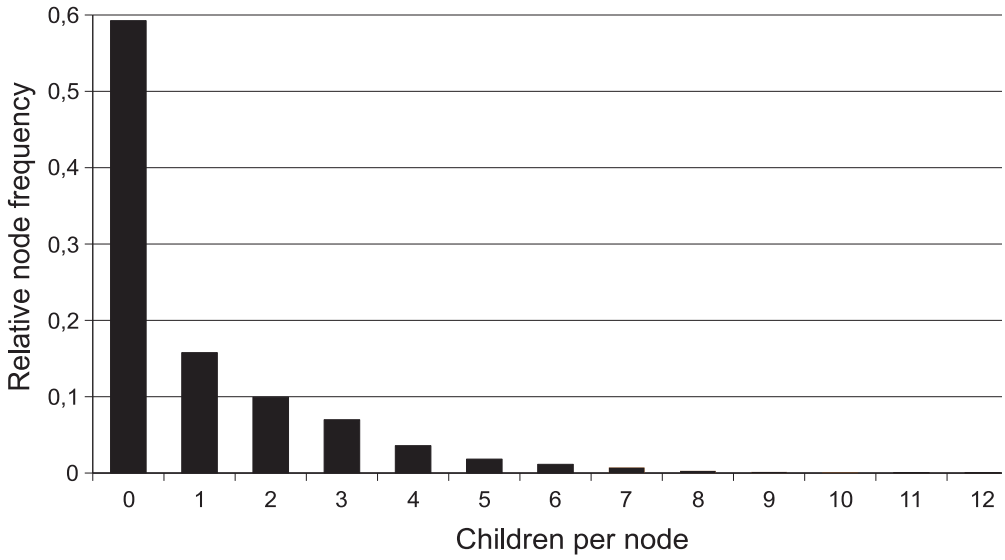


Figure 5.11: Distribution of out-degree

5.3.4 Delivery Rate

As discussed above, the convex hull created from the cover areas of a set of nodes may contain locations that are actually not in the cover area of any of these nodes. Thus, a message may be routed to a subtree that does not contain any node covering the target location of the message. The delivery rate depends on the topology of the network and the cover density, i.e. the number of nodes whose cover area contains a certain location.

Figure 5.12 shows the increase of the delivery rate when the cover area of single node is increased. It seems clear that a nearly full delivery rate, for example $r > 0.99$, cannot be achieved practically. The reason is that this would require a cover area per node that is very large compared to the size of the deployment area. For high precision of results, and low resource usage, it is best to keep the cover area small.

An increase node density d, which denotes the average number of neighbours of each node, leads to a larger number of subtrees in each node as the tree construction favours shallow trees. This causes the available cover nodes to be distributed among a larger number of subtrees, thereby reducing the probability that a cover node is contained in a subtree.

Figure 5.13 shows that an increased cover density has the same effect on graphs whose topology is determined by the Gabriel graph. However, using the Gabriel construction, the topology is invariant to changes in node density. Therefore, node density shows little to no effect in the simulation results.

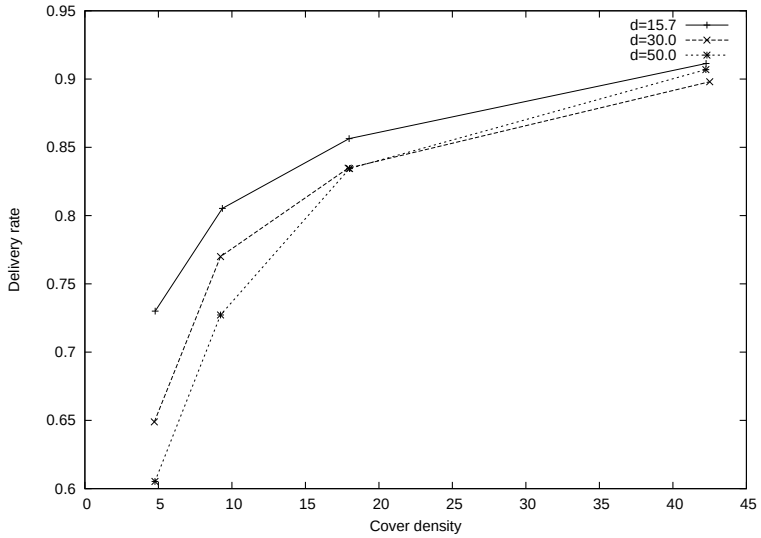


Figure 5.12: Delivey rate for varying node density (d) with increasing cover density; graph topology is based on reachability

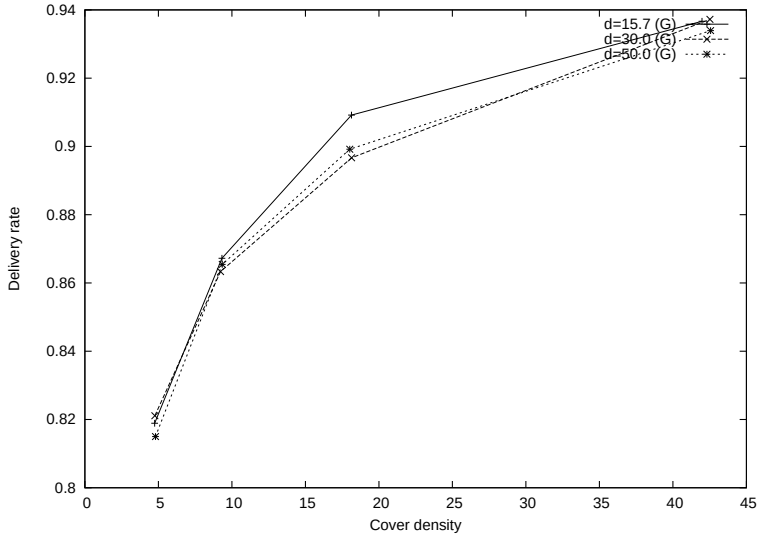


Figure 5.13: Delivery rate for varying node density (d) with increasing cover density; graph topology is determined by Gabriel graph (G)

There are several other ways to increase the delivery rate. One method is to switch the routing mode as soon as the message arrives at a node of the routing tree where neither the node's cover area nor any of its children contains the message's target location. Now, the subtree is chosen whose cover area is closest to the target location, and the message is routed to this subtree. This continues until the message arrives at a leaf node, from which point on the message is routed, for example, using a geographical routing scheme. This procedure only kicks in when routing has gone wrong. The overhead of this method would be slightly more complex code in the nodes to implement the search for the closest cover area and geographical routing scheme over short distances. No additional messages are sent, but the delivery rate is increased to 100%.

Another method to increase the delivery rate is to pass a message always to *all* subtrees that presumably cover the target location. That way, the message will eventually reach a node that covers the target location. However, the message will also travel many more hops than are necessary, increasing the message overhead significantly at much simpler node logic.

5.4 Security Evaluation

5.4.1 Basic Security Model

The security of a multi-path communication scheme is, first of all, provided by the fact that the attacker has to compromise multiple intermediate nodes in order to break a single communication relationship. If k ($k \geq 2$) paths are used for transmitting a message, or authentication codes, compromising a number of paths smaller than k will at least lead to detection of the attack. Thus, if at least one path remains sound, the integrity of messages is ensured.

We assume that the individual paths of a multi-path scheme use only link authentication. This means that a single compromised node on such a path will compromise the complete path. It is, of course, possible to employ more advanced authentication schemes on individual paths. This possibility will be explored in the next chapter.

For the basic determination of the security of tree paths we consider a random spread attack. Using a link authentication scheme, a single compromised node will break an individual path. Let x be the number of compromised nodes in the network, and N be the total number of nodes. For a tree path, this leads to the following compromise probability: $p_c = 1 - (1 - \frac{x}{N})^{L_m}$. Using k paths in a multi-path scheme, the integrity of a message is compromised if all paths are

compromised, i.e. with probability p_c^k.

Under a concentrated attack, we intuitively expect an advantage from the spatial separation of tree paths. This is in contrast to a random attack, where spatial separation provides no such advantage. But when a certain confined area is under attack, compromising all paths that lead through that area, we can hope to circumvent the area with at least one of multiple paths. We would therefore expect that a concentrated attack is not as effective as a random attack against the multiple tree path scheme.

The improvement spatial separation can provide has its limitations, however. First of all, the approach does not scale well beyond two paths. A third path will likely have intersections with the other paths, and it will be on the same side of the line given by the communication endpoints as another path. Thus, with more than two paths, there is no real spatial separation anymore.

The second limitation is that spatial separation works well only if the geometrical arrangement of paths and attack area is favourable. In many cases, spatial separation does not help. If the attack area is close to one of the communication endpoints, the spatial separation between paths is low, and it is not unlikely that both paths go through the attack area. If the attack area grows, it also becomes more likely that it covers both paths.

If both endpoints are close to each other, multiple paths are mostly useless. If the endpoints happen to be both outside of the attack area, it is likely that the nodes between them are also uncompromised. In that case tree paths or, for that matter, any multiple path scheme, can provide no additional security. On the other hand, if one of the endpoints is compromised, no security scheme can help.

5.4.2 Resilience Against Attacks

Figure 5.14 shows simulation results for the tree-based multipath scheme under various attacks. It is easy to see that, as discussed in the previous section, the multipath scheme provides to advantage to hop-to-hop authentication under a random spread attack. This type of attack does not geographically cluster the compromised nodes, thus the spatial separation of the multipath scheme is of no use.

The scheme is also not helpful against a partitioning attack. If both endpoints are located on different sides of the attack line, all paths between them are affected by the attack, so using multiple of them does not provide an advantage.

The simulation verifies, however, that the scheme is helpful against the types

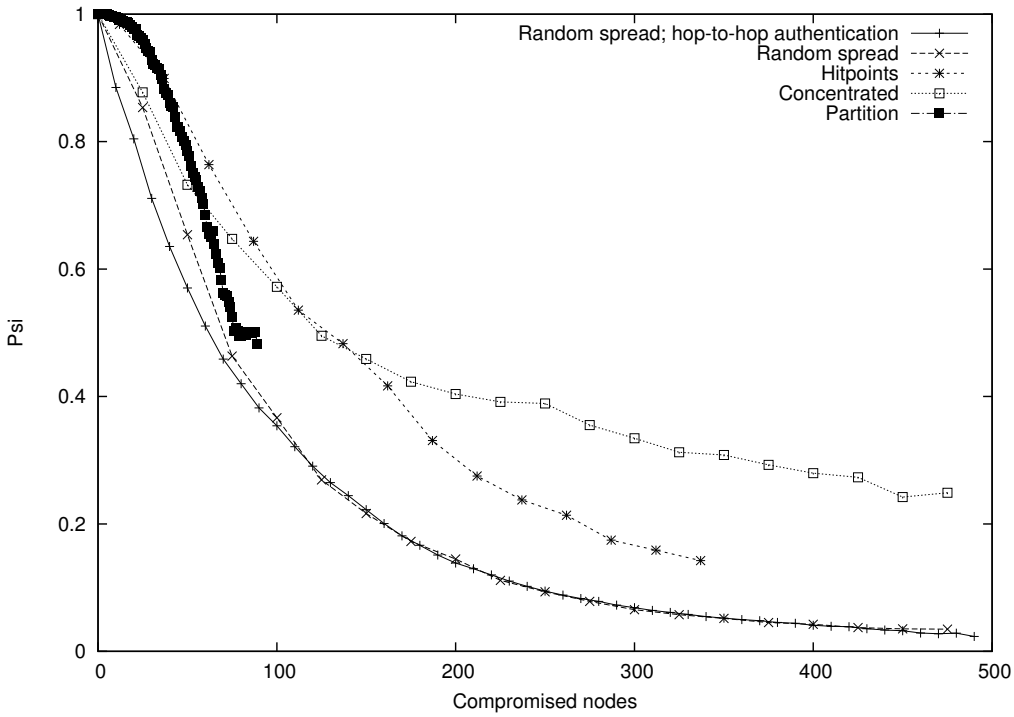


Figure 5.14: Resilience of the tree-based multipath scheme under various attacks

of attacks where compromised nodes are clustered. In both cases, the hitpoint and the concentrated attacks, the scheme has significant advantages.

5.5 Related Work

The transmission of data over multiple paths has been widely studied and is extensively used in modern communication networks. Most practically used schemes, however, do not use the paths to transmit data redundantly but to increase the availability or the bandwidth of a connection. The basis for the security of a multipath scheme is threshold security.

5.5.1 Multiple Paths for Performance

Multipath routing is a commonly used technique on the Internet, where the OSPF (Open Shortest Path First) protocol [128] is widely supported. It supports the maintenance of routing information about multiple paths that all have equal cost. A router computes the shortest paths to all target nodes and keeps a list of possible next hops for each target. One of them is selected for each new transmission. Thereby, multiple paths are used between two end-points sequentially.

For mobile ad hoc networks, node-disjoint multipath routing has been widely studied. The main focus has been on improving the performance, i.e. band-

width usage and latency, and the reliability of message transmissions over multiple hops. Establishing fixed routes between end-points is impossible in such networks due to the mobility of the intermediate nodes or the end-points themselves. Node-disjointness is desirable in order to minimize the dependency of paths on specific nodes. There are many variants how multiple node-disjoint paths can be established, but most of them fall into two categories, either *on-demand* or *proactive* protocols.

In the *on-demand* approach, a sender that wants to transmit a message first sends out a message requesting path establishment. This message will be forwarded to the destination over multiple paths by intermediary nodes. The path information is recorded in these messages and can be used by the destination to select the best paths (for example, the most disjoint with the lowest delay). The information about the selected paths is then transferred back to the sender. An example of such a protocol is *split multipath routing* [111].

The *proactive* approach demands that nodes continuously maintain routing information. One example is "meshed multipath routing" [48]. A centralized communication architecture is assumed, with many source nodes and a single data sink, where nodes relay messages always towards this sink. For this purpose, they classify their neighbours into two categories: nodes that are farther away from the sink and nodes that are closer to it. Messages are generally forwarded to all neighbours that are closer to the sink. Alternatively, only one neighbour is selected to which the message is forwarded. Note that in this approach, node-disjoint paths are not explicitly created but if they exist, they are implicitly used.

The multipath routing schemes described by Ganesan et al. [68] are intended to provide alternate paths in the event that a node on the primary path fails. The primary path is usually the shortest path, or the path that provides the lowest latency. There are two kinds of alternate paths, either completely or partially node-disjoint to the primary path (the latter are called braided baths, since they try to circumvent single nodes on the primary path). The method of construction prefers alternate paths that are geographically close to the primary path, for both types of alternate paths. The construction of alternate paths depends on local interactions only, nodes need no global knowledge about the topology of the network. The goal is to be as energy-efficient as possible. By construction, this approach does not lead to spatial separation but instead prefers paths that are in close proximity.

Spanning trees have been proposed by Chen et al. [40] for routing in mobile ad hoc networks. In that work, the focus has been put on maintaining a single spanning tree while nodes are mobile. Nodes are identified by their IP addresses

and each node keeps a complete list of all addresses of the nodes in its subtree. Based on that work, multiple trees have been proposed in [139] for improving the message delivery rate and optimizing path length. The set-up of a new tree is initiated when this new tree can provide a more efficient route for a message flow. Using (multiple) tree paths for security purposes has not been considered in these papers.

A related problem to ours is finding disjoint paths such that every communicating pair of terminals uses an exclusive path. This is important for admission control (where paths are allocated to sessions based on quality-of-service parameters) and optical routing (where router switches along the path are synchronized). This allows a maximum of $N/2 = O(N)$ communicating pairs at any time out of a maximum of $N(N-1)/2 = O(N^2)$. An algorithm for approximating the exact number of pairs that can be connected is given in [93]. In mesh networks, it is hard to achieve a high number of disjointly connected nodes. Each node acts as a router as well, and each connection being routed through a node diminishes its choice of connections to build itself. The maximum number of overall connections is reached if all connections are between direct neighbours.

5.5.2 Spatial Separation

An interesting approach to multipath routing has been proposed by Burmester et al. [33]. Based on the locations of the sender and the target, circles are calculated that are incident with both the sender and the target. The center of a circle and its radius, as well as an orientation are piggy-backed on a message. Each node forwards the message to the next best node according to these parameters such that the message travels close to the circle line. The piggy-backed information is sufficient to make this decision locally. This approach is problematic in networks with "holes" as it may not be possible to stay close to the circle line. For such cases, this approach could be extended with a mechanism similar to face routing.

Figure 5.15 shows an example with four possible circles. One of them is clearly too large, since it exceeds the boundaries of the network and would not be usable for routing. The other three are usable, and it is easy to send three messages on routes that are spatially well separated from each other. In order to achieve a high degree of spatial separation, the orientation for one of the small circles would be chosen clockwise and counter-clockwise for the other one. For the medium-large circle, either clockwise or counter-clockwise orientation would be appropriate, depending on the cost one is willing to spend for spatial

separation.

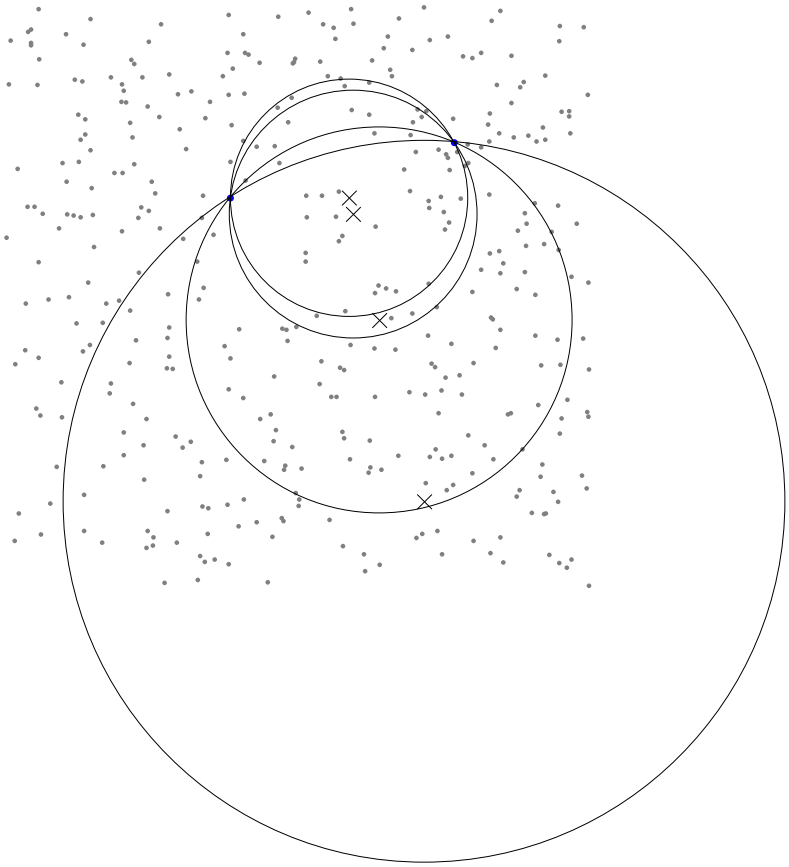


Figure 5.15: An example for circle based geographic multi-path routing. The crosses indicate the centers of the circles

Spatially separated paths are also created by the approach proposed by Voigt et al. [185] with the intention of avoiding interference between transmitting nodes. The goal is to allow the transmission of messages on multiple paths at almost the same time. To that end, a "forbidden zone" between sender and target is established, which is a corridor of a certain width through which a message may not be routed. Two paths around this corridor, one on each side, are established for routing. If the corridor is broad enough, interferences are minimized.

5.5.3 Threshold Security

Generally, the attacker model in a multipath environment is determined by an *upper bound k*, where *k* is the number of compromised paths between a fixed source and a fixed destination [165]. Compromising a path means either that messages on this path can be read by the attacker (a *passive* attack, breaking the secrecy), or that messages can be manipulated or injected by the attacker (an *active* attack, breaking the resilience).

Threshold security schemes can cope with up to k compromised parties, which could be represented by nodes, links, or paths, for example. A (n,t)-threshold scheme is defined by the number of shares n and the minimum number t of shares that are necessary and sufficient to reconstruct the secret d. The security of the scheme relies on its properties: (1) less than t parties cannot construct d (2) any set of t distinct shares is sufficient to construct d.

As shown in [56], $2k+1$ disjoint paths are required to obtain secure message transmission when faced with a k-bounded attacker who completely controls k paths, i.e. can read and write messages at will on these paths. Intuitively, this result is obtained by encoding and splitting up a message such that (1) $k+1$ correct parts are sufficient and necessary to reconstruct the message and (2) no information about the message can be learned by reading k or less parts.

When using multipath routing in order to achieve a security goal, such as confidentiality or integrity, one must not only think about encoding and routing the messages themselves. It is also important to consider the possibility that the adversary tries to manipulate the process by which the used paths are constructed. If the adversary manages to convince a sender that multiple disjoint paths exist while in fact there is only one path that is controlled by the adversary, the sender cannot communicate securely. This issue is addressed in [10], where an algorithm is described for constructing edge-disjoint paths that is itself resilient against attacks.

5.6 Summary

Establishing short multiple disjoint paths in a network may be hard, thus we devised a method that yields longer paths but makes path set-up very easy. Our method has the additional advantage that the constructed paths are spatially separated, which can be an advantage in certain attack scenarios.

In general, establishing disjoint paths involves a trade-off between set-up complexity and path length. In order to minimize path lengths, a set-up procedure is required that involves complex message exchanges and keeping local state information. On the other hand, a simple procedure can be used if longer path lengths are acceptable.

Chapter 6

Integrity-Preserving Communications

6.1 Authentication and Integrity Protection

Authentication is an important tool for ensuring the identity and integrity of objects that are outside of one's own control. In particular, we consider the protection of the integrity of messages that are transmitted within a WSN. We show how message integrity relates to the communication patterns within a WSN and the assumed threat model.

6.1.1 Definitions

Being *authentic* for an entity means "being actually and exactly what is claimed", especially "worthy of acceptance [...] as conforming to or based on fact" and "true to one's own personality, spirit, or character" according to [125]. The last part of this definition suggests that authenticity is strongly linked to certain characteristics ("personality, spirit, or character") of an entity, which could be subsumed under the notion of *identity*. We will not go into the philosophical details of this notion. For our purpose it is sufficient to assume that there are features based on which the identity of an object can be established. The process, which accomplishes this, is called *authentication*.

In the context of computer and network security, there are three main kinds of objects to authenticate: *entities* (users, principals, nodes), *keys*, and *data* (messages). For the latter, an important property is *integrity*. The following definitions are taken from the *Handbook of Applied Cryptography* [122]:

Definition 6.1. *Entity authentication* is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired).

Definition 6.2. *Key authentication* is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.

Definition 6.3. *Data origin authentication* is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past.

Definition 6.4. *Data integrity* is the property whereby data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source.

In the course of this work, we most often refer to "messages" instead of "data". As a message, we understand a container for data, which is represented in some encoded form. However, in many cases these terms are used interchangeably. Also, we talk about "hosts" or "nodes" instead of "parties", but these terms are used interchangeably.

There are a variety of methods for entity and message authentication being used. In the physical world, biometric techniques are used for authenticating human beings: Fingerprints may be used for entity authentication, while voice recognition is commonly used for entity and message authentication (over the phone, for exmple). In electronic communication systems, cryptographic, key-based methods are required. Standard mechanisms for authentication include digital signatures and message authentication codes (cf. 2.8.2).

Maintaining data integrity is often achieved through mechanisms like redundant storage or checksums. This is sufficient to counter unsystematic errors which occur in natural phenomena like fatigue of material or random interference. In cases where data integrity is threatened by a skillful, malicious adversary, this is not sufficient. In such a case, data origin authentication through cryptographic means is used to detect unauthorized tampering with data. There is no known cryptographic primitive, which can be used to ensure integrity (against malicious adversaries) without recourse to authentication.

It has to be stressed that node authentication alone does not guarantee correctness. A compromised node may correctly report its identity and authenticate its message as it is expected from a correctly operating node. However, the data it reports may be manipulated, and there may be no way of detecting such manipulations. This is the reason why node capture attacks are so powerful.

The last two definitions will be the most important ones for this work, and both data integrity and data origin authentication (also called message authen-

tication) are strongly related. We are going to discuss these issues in further detail in the following section.

In order to make message authentication possible, some identity must be attributed to the source of a message. Otherwise, it would not be meaningful to claim that a message originated from some specific entity. The message source has to be described in a unique way to assign an identity to it. Usually, this is achieved by giving all potential sources (nodes) names that are unique within the respective frame of reference. This makes it possible to issue a statement like "message M originates from node p", where M may be represented by a bitstring, and the name p by an integer, for example. The purpose of authenticating message M is then to yield evidence about the validity of that statement.

When the authentication of message M is successful, the association to its source p is established. At the same time, it is ensured that M has not been changed since it was created by its source. Thus, authentication provides the authenticating party with two pieces of information about a message:

1. A source is attributed to the message.
2. The integrity of the message is ensured.

In this regard, authenticity is a stronger property than integrity alone, since it implies integrity and additionally links an attribute (usually an identifier) of the source to a message. Vice versa, if the integrity of a message is violated, i.e. part of the message is altered or deleted, its authenticity (with regard to the statement above) is lost.

Some considerations may illustrate this fact. If a message is altered (i.e. at least one bit changes) after the message has emerged from its source, a new message is in fact created. The original message has served as input for this transformation, and both messages may share large pieces of data. Still, two different messages now exist. And, unless the modification was done by the original source, they have different sources. Thereby, an altered piece of data looses its authenticity.

Differently stated: If it is possible to verify that a message originates at the source that it is claimed to come from, this means that the message has not been changed since the time it emerged from its source. Such a verification is usually based on a statement issued by the source itself, which acknowledges that the source has created the message. This statement applies to the message in its specific form at creation time, and any change to the message invalidates the statement.

These considerations are intended to illustrate that integrity can be achieved through authentication, and the violation of a message's integrity makes it im-

possible to authenticate the altered message with regard to the original statement of origin.

6.1.2 Identity, Integrity, and Authentication in WSNs

Identities of nodes in a communication network may exist on different system levels. For example, serial numbers or structured, unique addresses such as the 6-byte MAC layer identifiers used in Ethernet networks, simple random numbers (with a high probability of uniqueness), or public/private key pairs are being used. Often, identities are the foundation for security measures such as access control, which could be based for example on ID filtering or public key signatures.

The key distribution schemes of Chapter 4 assume that a (random) identifier exists for each node. This identifier is used to create a pseudo-random number sequence that determines, which keys are assigned to a node. Additionally, the assigned subset of keys can also be considered as an identifier for a node, although it is only unique with a certain probability. However, a key subset has the advantage that its validity can be checked by other parties by testing whether the node actually knows the keys that have been assigned to it. In fact, the major purpose of this kind of identifier is not to corroborate some unique name for each node, but to prove that the node is legitimately participating in the sensor network.

As mentioned above, the knowledge of the identity of a message's source does not imply any trust in that source or the truthfulness of a message's content that is attributed to that source. This trust has to be established through some means that is usually outside of the scope of a direct communication relationship. In a WSN, trust is usually assumed based on the fact that nodes belong to the same WSN deployment, or are operated within the same administrative domain. Membership in the same deployment can be established, as mentioned in the previous paragraph, by using a key predistribution scheme. Alternatively, a list of valid identifiers could be predistributed to all nodes, or certificates could be used.

A system where trust is assumed if a valid key set is presented may be vulnerable to the so-called *Sybil attack* [58]. This attack is based on the creation of new identities. By recombining the key sets of captured nodes, new key sets can be created, which can then be used to simulate a potentially large number of virtual (fake) nodes and influence the result of a WSN's operation. An intrusion detection system [137] may help to detect such virtual identities, but it may involve a significant overhead.

It has been noted [46] that in many systems, other characteristics than identity are usually more important to know, such as location or behaviour. For example, it may be important to know that the device, with which a communication relationship is being established, is indeed located at a certain place. However, this information cannot be conveyed by an identifier of the device alone and must be established through other means. For example, if an interface based on physical contact, such as a cable plug, can be accessed, proximity is immediately established. Verifying the location of a remote device is much harder and usually involves a trusted entity, such as a location beacon [158].

Sometimes, authentication is merely used to ensure integrity. In [42], the protection of military, large-scale WSN deployments against attacks on their integrity has been studied. The main objective is to prevent an adversary from placing a majority of malicious nodes within an area. This ensures that reliable information (e.g. for surveillance) can be retrieved. For that purpose, nodes are equipped with batch keys, which correspond to their deployment area, and diversity keys, which provide uniqueness within a deployment area. When information is retrieved from a set of nodes, it is made sure that nodes with duplicate diversity keys or inappropriate batch keys are disregarded. Thus, cryptographic node authentication servers as a means to ensure the integrity of application-level data.

In summary, we observe that identity serves an important purpose for message authentication. However, identity is not as important in WSNs as in other systems. The reasons are that trustworthiness is not guaranteed solely by a known identifier, and group membership is more relevant than individual identity. We thus conclude that integrity protection without individual authentication is sufficient for many applications. In this chapter, we show how message integrity protection can be achieved with much less costly means than would be necessary for end-to-end authentication.

6.2 Basic Interleaved Authentication

In the following authentication scheme, messages are authenticated not end-to-end, i.e. between the source and the sink, but *locally*, i.e. only between intermediate nodes within a small hop distance. Thereby, it is unnecessary to transfer any secret keys or certificates to the sink, while the scheme still effectively preserves the integrity of messages.

6.2.1 Protocol Description

Consider the basic linear graph in figure 6.1 representing a simple sensor network. The solid edges between adjacent vertices represent physical communication links that are used for transmitting messages. This *communication graph* is obtained, for example, by constructing the routing graph used for geographic routing protocols such as GPSR [91]. For constructing such a graph, certain edges are removed from the full connectivity graph, which contains information about which nodes are reachable from which other nodes. This could mean that, in principle, S_1 may be able to send a message directly to S_3. However, for various reasons this link is not used.[1] Thus, S_2 is a one-hop neighbour, while S_3 is a two-hop neighbour of S_1.

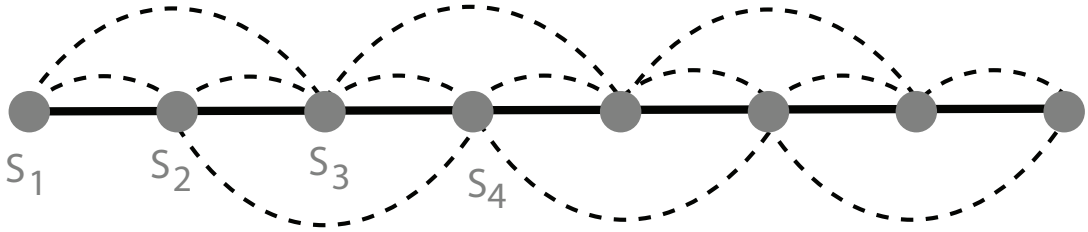


Figure 6.1: A simple communication graph with interleaved security relationships

The dashed edges represent pairwise shared keys. Together with the vertices, they form the *authentication graph* of the network. In the example, each node has a shared key with each node within its two-hop neighbourhood, i.e. with all of its one-hop and two-hop neighbours. The key shared between nodes S_i and S_j will be denoted as $K_{ij} = K_{ji}$. It is relatively straightforward to set up such a setting using the techniques described in chapter 4.

Whenever a message is forwarded along a communication path, it is being authenticated using these keys. There are two cases we have to consider, message creation and message relaying.

Message creation When a message is generated from scratch, the source node creates k MACs targeted at the subsequent nodes on the path. As each of these nodes has a shared key with the source of the message, the authenticity of the message can be directly verified. For the first k hops on a path we can therefore speak of message *authentication*. If any of the first $k-1$ nodes tampers with the message, the following node will detect the manipulation based on the MAC from the source.

[1] One of the reasons is energy efficiency. When transmitting a message from S_1 to S_3 via S_2, both S_1 and S_2 can reduce their transmission power. In total, this saves energy compared to S_1 directly sending to S_3 with higher signal strength. Another reason is that geographic routing requires a planar communication graph.

In our example, S_1 creates a message M and two MACs $\{M\}_{K_{12}}$ and $\{M\}_{K_{13}}$, and sends everything to S_2. The message includes an identifier denoting its origin, S_1 in this case. S_2 uses this identifier to determine that it shares a key with the message source, and can therefore directly verify the authenticity of the message. S_2 computes $\{M\}_{K_{21}}$ and checks that the result matches the respective value received from S_1. As S_2 has established the authenticity of the message, it creates two new MACs using its keys K_{23} and K_{24}, and forwards them to S_3 together with the message itself and the remaining MAC that was created by S_1. Similarly, S_3 checks that the MAC from the source is valid. The MAC $\{M\}_{K_{23}}$ provides no significant additional value beyond link authentication as S_1 and S_3 share a key anyway, and since S_1 is the creator of the message, S_3 does not require a "second opinion" on the validity of the message.

Message forwarding A node that is too far away from the message source to share a key with it accepts a message only if it has k MACs from different nodes attached that the relaying node can verify. Since these MACs have been created not by the message source itself but by other relaying nodes, we call them *attestations*. If all attestations are verified correctly, the relaying node can be more or less sure that the message has not been tampered with. This belief is justified if there are not too many colluding compromised nodes. Under the assumption that key material has not been disclosed, all k attestating nodes would have to collaborate in order to convince the relaying node to accept a falsified message.

In the example, S_4 is the first node on the path that cannot verify the message's authenticity directly but has to rely on attestations only. Both S_2 and S_3 have created such attestations for S_4. If both can be verified, S_4 assumes that the message is correct and creates attestations for S_5 and S_6. If both S_2 and S_3 are compromised, they can convince S_4 to accept a message as originating from S_1 even if they have altered the content of the message. There is no way for S_4 to know what the original content has been as there is no direct channel between the source and S_4.

In this basic form, we call this authentication scheme *k-Canvas* or simply *Canvas* if the parameter k is known. In most examples throughout this work, we will use $k = 2$. The name has been chosen since the graphical representation of the scheme resembles the visual impression of a cut through a canvas fabric common in cloth manufacturing.

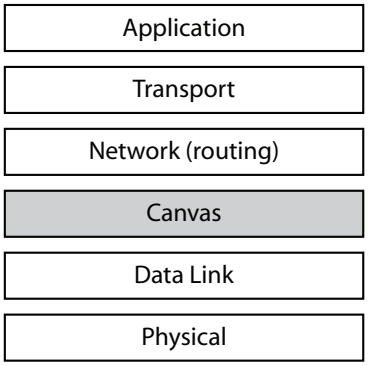


Figure 6.2: Location of the *Canvas* layer in the network stack

6.2.2 Formal Specification

We are now going to take a closer look at the details of the *Canvas* authentication scheme. We describe the embedding of *Canvas* in the network stack of a sensor node, and the operations necessary for its implementation.

The task of the network stack is to determine how to handle incoming and outgoing messages. For example, it must decide whether a message should be handed over to a higher system layer (e.g. an application), or rather be routed to a neighbour. The *Canvas* scheme is being used as part of the routing layer of the network stack: each incoming message is checked for its attestations before it is further processed, and to each outgoing message the necessary attestations are attached. The placement of *Canvas* within a (simplified) network stack is shown in Fig. 6.2.

We require a few helper functions for the *Canvas* layer to work. The first is `prefetch-hop`, which exists in the routing layer and can be called from the *Canvas* layer. It takes a small integer n as an argument and returns the ID of the n-th-next node on the path towards the message's destination:

$$\texttt{prefetch-hop} : \texttt{Integer} \rightarrow \texttt{Node-ID}$$

Thereby, it can be determined which keys must be used for creating the necessary attestations.

Another function, called `neighbours`, delivers the IDs of a node's neighbours that are located exactly in a specified hop-distance:

$$\texttt{neighbours} : \texttt{Integer} \rightarrow \mathcal{P}(\texttt{Node-ID})$$

Note that $\texttt{neighbours}(i) \cap \texttt{neighbours}(j) = \emptyset$ for $i \neq j$.

The core functionality of the *Canvas* layer is checking whether a message should be accepted as "authentic" (or "genuine"). The decision procedure for

this is encapsulated in the function *canvas-accept*. Pseudo-code for this function is shown in Algorithm 4. The function takes all available message parameters as input, i.e. its source, destination, payload, and authentication information. The latter, given in the variable $\mathcal{C}$, is a set of tuples. The components of these tuples are drawn from the following data types:

- A node ID denoting the creator of the authentication code.
- A node ID denoting the addressee of the authentication code.
- An integer counter value.
- An authentication code.

The function returns *true* either if the message is authenticated directly by the source, or there are k attestations for the message. There must be an attestation from a node in every i-hop distance from the current node. This prevents acceptance of attestations that have only been created within, for example, the 1-hop distance of a node, which reduces the opportunities for an attacker. Note that counter values are also checked for freshness to prevent the replay of messages that have already been seen.

The function *canvas-auth* (Algorithm 5) creates new authentication codes for a message. First, incoming authentication codes are removed from the authentication information block of the message. MACs that are not intended for the current node are retained. The function prefetch-hop is used to obtain the IDs of the subsequent nodes on the path. For the next k of them, the respective counter values are increased and MACs are created.

direct-accept-and-forward:
$M = \langle \text{DIRECT}, A, P, m, \mathcal{C} \rangle$
$d(X,P) > \tau$
$d(X,P) \leq \delta$
canvas-accept$(A,P,m,\mathcal{C})$
$\mathcal{C}' :=$ *canvas-auth*$(A,P,m,\mathcal{C})$
send: $\langle \text{DIRECT}, A, P, m, \mathcal{C}' \rangle$

direct-accept-and-process:
$M = \langle \text{DIRECT}, A, P, m, \mathcal{C} \rangle$
$d(X,P) \leq \tau$
canvas-accept$(A,P,m,\mathcal{C})$
process A, m

Table 6.1: *Canvas* message accept rules

Algorithm 4 *canvas-accept*$(A,P,m,\mathcal{C})$

Global values:
X: ID of current node
k: Parameter of the *Canvas* schme

Input:
A: message source
P: destination location
m: message text
$\mathcal{C}$: authentication information, a set of tuples

Output:
Return *true* if *Canvas* authentication succeeds

```
1:  if ∃(A,X,c,a) ∈ C then                          ▷ Is there a MAC from the source?
2:      b := {A,P,m,c}_{K_XA}
3:      if a = b ∧ c > c_AX then
4:          c_AX := c
5:          return true
6:      else
7:          return false
8:      end if
9:  else
10:     for i in {1,...,k} do                       ▷ Otherwise, check attestations
11:         if ∃V.(V,X,c,a) ∈ C ∧ V ∈ neighbours(i) then
12:             b := {A,P,m,c}_{K_XV}
13:             if a ≠ b ∨ c ≤ c_VK then
14:                 return false
15:             else
16:                 c_VK := c
17:             end if
18:         else
19:             return false
20:         end if
21:     end for
22: end if
23: return true
```

$M = \langle$ *flag, source, dest, data, mac* $\rangle$	
flag	DIRECT (marking a *Canvas*-authenticated message)
source	purported source node's identifier
dest	destination specifier
data	data payload of the message
mac	*Canvas* message authentication codes

Table 6.2: *Canvas* message format

d	distance function
τ	allowed acceptance distance from a destination
δ	allowed forwarding distance from destination

Table 6.3: Auxiliary rule parameters

Algorithm 5 *canvas-auth*$(A,P,m,\mathcal{C})$

Global values:
X: ID of current node
k: Parameter of the *Canvas* schme

Input:
A: message source
P: destination location
m: message text
$\mathcal{C}$: authentication information

Output:
Return modified authentication information

1: $\mathcal{C}' := \mathcal{C} \setminus \{(\_, X, \_, \_) \in \mathcal{C}\}$ ▷ Remove all MACs for X
2: **for** i in $\{1 \ldots k\}$ **do**
3: V := prefetch-hop(i)
4: $c_{XV} := c_{XV} + 1$
5: $a := \{A, P, m, c_{XV}\}_{K_{XV}}$
6: $\mathcal{C}' := \mathcal{C}' \cup \{(X, V, c_{XV}, a)\}$
7: **end for**
8: return $\mathcal{C}'$

The fundamental message acceptance rules for the *Canvas* authentication scheme are shown in Table 6.1. Messages are marked with flags that denote their type. Here, only type DIRECT is used, which indicates that the authentication of the message is completely handled locally, i.e. at each hop. Later, we will see another type, SHORTCUT, which extends the authentication of messages to larger distances. The format of messages is explained in Table 6.2. The destination location of a message is specified by geographic coordinates.

Table 6.3 lists the remaining parameters that are being used in rules. The distance function d yields the geographic distance between a node and a location. τ is a global parameter that determines the acceptable deviation from the exact destination location. If a node is located within a range of τ from a destination location, this node is qualified to be the final receiver of a message. This means that this node will not further relay the message. It could, however, inform the nodes in its vicinity that it has received the message. There could be multiple nodes that are qualified to receive a message for a certain destination, and multiple messages addressed to the same destination could be received by different nodes. It is up to a higher system layer, such as a clustering protocol or the application itself, to coordinate the activities of all these nodes.

Another global parameter denoting a geographic distance is δ. This parameter denotes the maximum distance over which *Canvas*-authenticated messages should be forwarded. If $\delta = \infty$, there is no limit on that distance, and full reachability is maintained. For variations of the basic authentication scheme

discussed later in this chapter, a finite value of δ can limit the impact of compromised nodes without losing reachability.

The rule *direct-accept-and-forward* is applied when the current node does not qualify as a receiver of the message (since it is too far away from the destination), the destination is within a distance of δ, and *Canvas* authentication is successful. If these conditions are fulfilled, attestations are created and the message is forwarded.

The rule *direct-accept-and-process* on the other hand is applied when the current node qualifies as a receiver of the message and *Canvas* authentication succeeds. Here, the message is processed by the current node.

6.2.3 Interaction with Routing Protocols

The purpose of a routing protocol is to forward a message between hosts (nodes in the context of sensor networks) so it eventually arrives at its destination. Two major issues in routing are addressing the destination of a message, and path setup. Conventional routing protocols are often not suitable for sensor networks since they fail to appropriately consider the limited resources of sensor nodes, the prevalent communication patterns, and the inherent redundancy in sensor networks. Here, we consider routing mechanisms that are well-suited for sensor networks, and examine how interleaved authentication interacts with them.

Flooding

The simplest mode of propagating a message through a network is flooding. While it guarantees that every relevant node receives the message, many redundant messages are transmitted. Due to its inefficiency, its applicability is rather limited. However, it deserves consideration as in some cases it is the only reliable way of distributing a message.

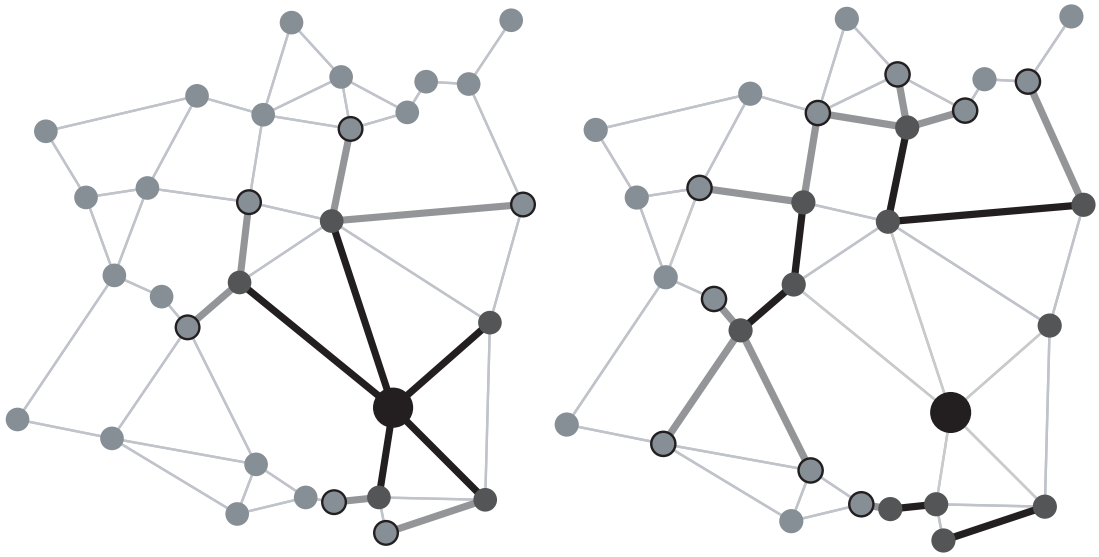
In a simple flooding protocol, each forwarding node transmits a message to all of its neighbours (except the one from which the message has been received). A node needs only to be aware of its immediate neighbours. Using the *Canvas* scheme for message authentication, a forwarding node has to transmit not only the message itself but also authentication codes for all the nodes on the next k levels of its forwarding tree. Although it is possible to send all these codes to all neighbours, it is very inefficient, since the nodes on deeper levels will not be able to make use of most of the authentication codes they receive. In order to reduce the overhead, it is reasonable for a forwarding node to impose some structure, a “forwarding tree”, on its k-neighbourhood, and use this structure to transmit only selected authentication codes to its neighbours.

This forwarding tree is constructed as follows. The source node starts by designating each of its neighbours as the root of a forwarding subtree of depth k. In Fig. 6.3(a), an example for $k = 2$ is shown. The source transmits authentication codes to these roots for all the nodes that are contained in their respective subtrees.

Each root then proceeds by adding another level to its own subtree. The nodes on the next $k-1$ levels of the tree are already fixed, so a root has to determine only the nodes that have to be included in the tree on the last level. It includes the nodes that fulfill the following two requirements:

1. They are located in a distance of k hops from the current root.
2. They are reachable from any node on the last level in the current forwarding tree.

Each of the thereby selected nodes is then assigned as a child to one of the $(k-1)$-level nodes of the current forwarding tree. Figure 6.3(b) shows the example situation after the first expansion step.



(a) (b)

Figure 6.3: The first two stages when flooding a message with *Canvas* ($k = 2$). The message source is shown as a big black dot. (a) In the first step, its immediate neighbours receive the message with authentication codes for the neighbour itself and the next level of nodes. (b) In the second step, the message is transferred to the next level and authentication codes for the second-next level are created. Thick black lines indicate the links over which the message is being sent. Thick grey lines represent the following tree level determined by the message authentication codes

This procedure guarantess that all nodes that are reachable from the source will eventually receive the message. The reason is that every node that is reachable from the source is included in at least one forwarding subtree, and all nodes

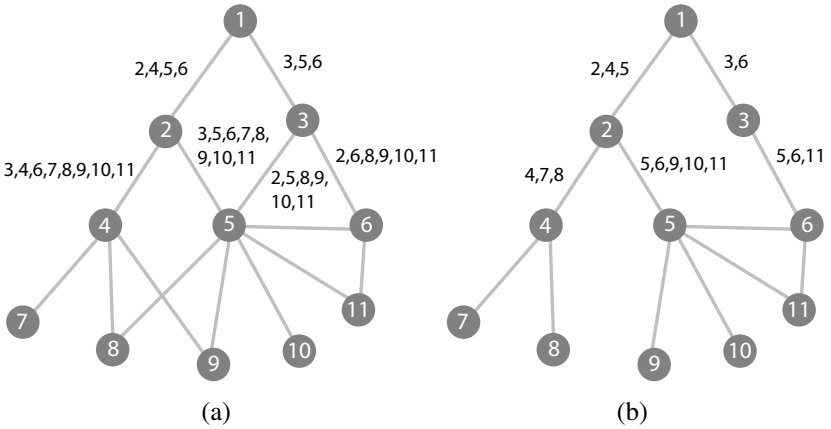


Figure 6.4: *Canvas* message overhead on the link level. Left: Authentication codes for broadcast-based flooding. Right: Authentication codes for a tree structure. Using a tree is much more economical

in a subtree will receive the message. Nodes that become a member of multiple subtrees, which is likely to happen for many nodes, will receive the message multiple times. As root, such a node will construct its own subtree as the union of all subtrees it is part of. This ensures that all nodes will eventually receive the message.

Compared to simple flooding, the number of authentication codes that are transmitted is significantly reduced. This can be seen by looking at a simple example. Figure 6.4 shows a network where nodes are labelled with numbers. Node number 1 is the source and floods a message to its neighbours. It adds authentication codes for its 1-hop and 2-hop neighbours (the links are labelled with the target nodes of these codes; the message flow is from top to bottom). In Figure 6.4(a), simple flooding is shown. In addition to the authentication code for its immediate neighbour, a node adds authentication codes for *all* of its 2-hop neighbours to each outgoing link. This is necessary since simple flooding is not aware of the topology beyond single-hop neighbourhood. Figure 6.4(b) shows a similar graph with some edges missing. Since a node constructs a tree before it forwards a message, not all links will be used to transmit the message. Nodes are now aware of their 2-hop neighbourhood and thus they can limit the set of authentication codes that have to be sent over a certain link. Thereby, a significant amount of data can be saved. This is confirmed by a large-scale simulation (500 nodes on a 1000×1000 plane with varying transmission range and therefore varying number of neighbours per node). As Figure 6.5 shows, the difference between simple flooding and the structured approach is approximately 10-fold.

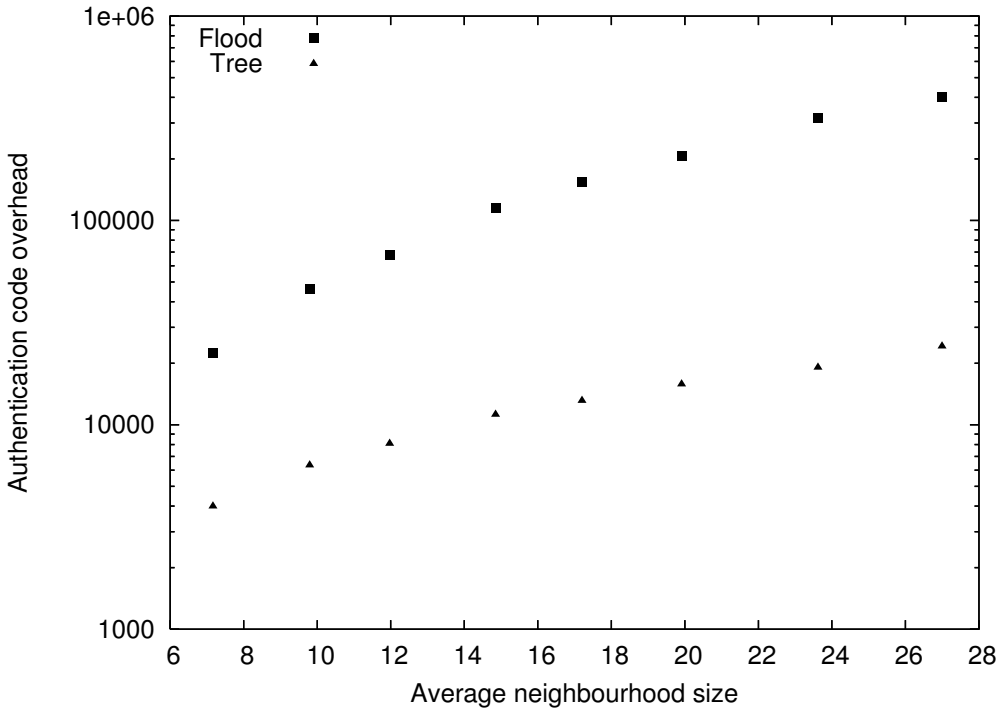


Figure 6.5: Total *Canvas* message overhead when using broadcast-based flooding vs. flooding over a tree structure. The difference amounts to an order of magnitude

Content-Based Routing

The main idea of content-based routing is to forward a message only to those nodes, which have stated an interest that matches the type of content the message carries [35]. This kind of data dissemination is especially suited for environments in which the sources and receivers of messages frequently change, and where there is no need for a mutual relationship between a source and the receivers. A good metaphor of this kind of data dissemination is the spreading of rumors: The exact source of a rumor is often unknown, and everybody can either act on the message the rumor conveys, tell it to others, or simply ignore it. This simplicity is advantageous in large-scale sensor networks. Consequently, a protocol for content-based data dissemination in sensor networks has been proposed that builds on this idea [31]. The main idea of "rumor routing" is that a message source distributes an event notification over a limited number of (random) paths. Any node interested in a certain kind of events releases a query that is forwarded along a (also random) path through the network. As soon as the query hits a node that is part of a matching event path, which happens with high probability, a path between source and sink can be established.

Rumor routing builds event and query paths randomly, i.e. at each hop, the next hop is determined by randomly selecting a node from the set of neighbours. The extension to a k-neighbourhood and adding *Canvas* authentication

codes is straightforward. A node receiving a message also receives authentication codes for the next $(k-1)$ hops, so the following hops on the path are already determined. The current node therefore has to select only the kth hop following the current node on the path. An authentication code for that node is added to the message before it is forwarded along the path. Thereby, event and query messages are being *Canvas*-authenticated. Establishing a random path using *Canvas* requires a "lookahead" of k hops instead of one. As the paths are set up randomly in any case, there is no fundamental difference. Thus, the functionality of such paths is not affected by choosing randomly the kth-next hop instead of the immediate next hop.

A second, query-driven, approach to content-based routing is to set up gradients from a message source to interested receivers, called "directed diffusion" [86]. First, the query is flooded. When it reaches a matching source, the source sets up gradients to those neighbours from which the query has been received. These nodes do the same until a path from the source to the sink is established. Gradients are used for selecting the optimal path, and for repairing broken paths.

Canvas authentication for flooded queries can be performed as described in the previous paragraph. The prospective message source receives possibly multiple copies of the request and sets up gradients in the opposite direction. Note that a message carries the node identifiers of the last k hops in the authentication records. The source stores the last k hops for every gradient. This allows it to do k-*Canvas* authentication for the event messages as they are being sent to interested nodes.

Note that *Canvas* authentication alone does not make these protocols "secure" per se. It may still be possible for an adversary to attack these protocols on a different level than message authentication. For example, the event paths set up for rumor routing may be misdirected in such a way as to minimize the chance of them hitting query messages. Or, a node may set up bogus gradients, thereby cluttering up the memory of other nodes. Message authentication alone cannot prevent such attacks.

Geographic Routing

Geographic routing [91, 100] uses location information for addressing nodes instead of identifiers. It is assumed that all nodes know their locations, and the locations of the nodes in their immediate neighbourhood. It is straightforward to extend this knowledge to the k-hop neighbourhood, which is required for *Canvas* authentication. Routing proceeds in two modes: greedy and face

routing.

In greedy mode, the next hop is selected based on distance from the target. The neighbour which is closest to the target will be the next hop. Since routing decisions are made deterministically, a node can pre-compute the decisions that will be made by the next k nodes as their location information is known. Therefore, it can attach the required authentication codes for the next k hops to the message. Effectively, each node adds the kth-next hop to the path instead of the next one.

If no suitable neighbour is available for selection in greedy mode, the protocol switches to face routing. Here, the selection of the next node is based on the location of the current node, its location relative to the target, and the location of the node that switched to face routing mode. The latter information is conveyed as part of the message. Again, all relevant information can be made available for the next k hops. Thus, the current node can anticipate the path and the required authentication codes can be attached to the message.

Sometimes it is necessary to make the selection of the next hop at the current node instead of the k-th-previous node, for example due to load balancing or compensation for node failures. This is equivalent to a change in the network's topology, which has to be communicated back to the previous nodes.

Dynamic Source Routing

Dynamic source routing (DSR) is a routing scheme which is not well adapted to the requirements of sensor networks. Route establishment is based on flooding, which can exhaust the resources of sensor nodes quickly if used too often. Therefore, it is better suited for small networks. It has been introduced in the context of ad hoc networking [87].

Route discovery is based on flooding a route request message until it arrives at the target node. A unique request identifier set by the initiating node allows the detection of messages that have already been handled. Each intermediate node adds its own identifier to the *route record* of the message before it rebroadcasts it. Thereby, loops can be detected, and the record can be returned to the initiator which uses it to determine the path to the target. The propagation of route requests can be adapted to a k-neighbourhood as discussed in the section on flooding above. Note that the flooding model results in optimal routes, i.e. with minimal hop count, as required by the DSR scheme.

Route maintenance is based on implicit (by overhearing the retransmission of the message by the next hop) or explicit acknowledgements (if implicit acknowledgement is not available) between hops. If a link fails, a route error is

returned to the source which then deletes the route from its cache. It may then initiate a new route discovery for that target.

The flooding of route requests can be *Canvas*-authenticated as described above. Data messages are routed according to the path included in the messages, which has been set by the source. A node can easily extract the next k hops from this path and create the *Canvas* authentication codes accordingly. Explicit acknowledgements are exchanged between neighbouring nodes, so the local keys shared between these nodes can be used for their authentication. Implicit acknowledgements can remain unauthenticated; the overhearing node can decide if the message has been forwarded correctly based upon the message text it receives. Route error messages are sent back to the message source over a potentially large distance. They would be *Canvas*-authenticated just as ordinary data messages. In summary, it can be said that *Canvas* authentication integrates well with DSR.

6.2.4 Application to Data Aggregation

Data aggregation is an important application in WSNs. Many (or all) nodes are supposed to send a message each to a single sink. Forwarding such a large number of individual messages is expensive, since the nodes that are close to the sink have to deal with many messages (for each node in their subtree, there is one message). However, such an approach would allow the sink to authenticate the messages it receives, assuming that it has a shared key with each node.

From an efficiency point of view, it is better to let forwarding nodes do some preprocessing and only send aggregated messages towards the sink. In contrast to broadcasting, where each node receives the broadcast message once, now each node has to *send* a single message. However, such aggregated messages cannot be attributed to single nodes, and therefore authenticating them is impossible.

The threat against aggregation schemes is that nodes that are close to the sink can control what is being forwarded to the sink. The sink is not able to check whether the information it receives has been calculated based on data generated by more remote nodes.

Some techniques have been devised to enable authentication of aggregated data (cf. Section 2.7.4). Here, we describe how the influence of a malicious node can be mitigated using interleaved authentication. Note that in the following, we do not explicitly mention authentication codes for 1-hop messages, though we assume they are always included. Figure 6.6 shows an example aggregation tree with node A as root. Node B aggregates data that it receives from

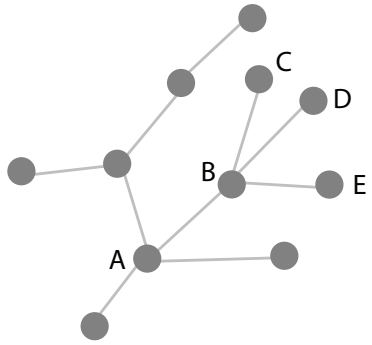


Figure 6.6: Aggregation tree

nodes C, D, and E, and forwards it to A. Usually, the message from B to A only contains the aggregated data calculated by B, based on the input from B, C, D, and E. This leaves A with the only choice of trusting that B sends correct data. However, from time to time, A checks whether the calculation done by B is correct. For doing so, it requests the following additional data:

- The original inputs from B, C, D, and E.
- Authentication codes from C, D, and E for their original inputs to B. Since these nodes are 2-hop neighbours of A, creating such authentication codes is possible.

This data allows A to check whether the calculation of B is correct by performing the same calculation itself. If the result differs from what B has sent, this should be interpreted as a sign for a potential ongoing attack. Additionally, B may be considered for exclusion from further operation in the network, and the aggregation tree rebuilt.

The times when A performs such checks must be chosen randomly to avoid that B being able to predict them, which would obviously allow B to render these checks ineffective. Also, the additional data that is required for checking amounts to a significant transmission load, so the check should only be done rarely. This constitutes, of course, a trade-off between efficiency and the likelihood of catching a malicious node.

6.3 Performance Evaluation

The evaluation of *Canvas* is based on a comparison with a generic end-to-end signature scheme. We compare the increase in time to delivery, which is due to additional, e.g. cryptographic, operations, and the transmission delay for additional data. We first look at the overhead for a single message. Then, we consider the overhead for a sequence of messages, e.g. a sensor data stream.

6.3.1 Single Message Overhead

For each message being transmitted from the source to a target node, the *Canvas* scheme requires that each node on the path checks k authentication codes (MAC) and generates another k MACs addressed to nodes further down the path. Nodes close to the source will check fewer MACs, while nodes close to the target will generate fewer of them. In general, on each link, $T = k(k+1)/2$ MACs are transmitted. For links that are close to the source or the sink, some of these MACs are "missing", and we can adjust the number of MACs being transmitted by:

$$\delta(d) = \frac{(k-d)(k-d+1)}{2}, d < k$$

where d is the distance of the link from the source/target, starting with $d = 1$ for links that are adjacent to the source/target. For each link with $d < k$, T is reduced by $\delta(d)$ in order to determine the number of transmitted MACs.

At each node, checking and generating the MACs consumes time, which delays the relaying of the message. If the HMAC construction is used, the complete message has to be processed separately for each MAC being verified or generated. From a performance point of view, the secret suffix construction is therefore preferred as it requires the message to be hashed only once. The hash value generated is then used for MAC verification and generation.

When a digital signature scheme is being used, the following information has to be transmitted in order to authenticate a message: the *public key* of the source node, a *certificate* stating that the public key is authorized, and a *signature* of the message. For each pair of communicating nodes, the public key and the certificate have to be exchanged only once. The encoding format for keys, signatures, and certificates may induce additional overhead. For example, a certificate format such as X.509 [81] contains information about the validity of the certificate and other facts that need not to be explicitly represented in the context of wireless sensor networks. Therefore, we can restrict ourselves here to the minimum amount of data required. An ECDSA signature is a pair (r,s). The size of both r and s is governed by the parameter n, which should be at least 160 bits large according to [88]. Thus the size of a signature is at least 320 bits. The public key is the result of a multiplication of the private key with a point. Using a standardized elliptic curve with key length 192 bits, this yields a public key size of around 600 bits.

The effort of generating and verifying the signature is only induced at the source and the target nodes. Intermediate nodes on the path do not have to perform any computation but need to simply relay the message. Thus, no further delay is introduced by this scheme.

We compare the 2-*Canvas* scheme with elliptic curve signatures. The comparison is illustrated along three measures: the overall single-message bandwidth overhead, i.e. the sum of the authentication data transmitted by all nodes; the time overhead for a single message; and the bandwidth overhead if multiple messages are being exchanged.

The parameters are fixed as follows: the message sizes used are 64 byte and 512 byte, which makes a difference for the *Canvas* scheme as the message must be hashed by each node. For the *Canvas* authentication codes we use either all 20 bytes that are output by the SHA-1 algorithm, or the truncated version with 7 bytes, which correspondes to the security level of DES. The key length for EC signatures is assumed to be 192 bits, which provides a lower security level than the 160 bits of the SHA-1 output. The times required for performing operations are taken from Tables 3.1 and 2.2 whereby the time for EC signature generation is estimated to be 600 ms.

Figures 6.7 and 6.8 show the time overhead induced by each of the authentication schemes. The delay of the signature scheme is constant as only the source and the target nodes have to perform additional operations. With the *Canvas* scheme, each intermediate node has to perform operations that pile up over the path. At some path length, the overhead produced by *Canvas* will be larger than that of a signature scheme. The graphs show clearly the disadvantage of HMAC, which requires the message to be hashed for each MAC. The time overhead increases the end-to-end latency of the message, in addition of the transmission time.

Figure 6.9 illustrates the bandwidth overhead of the authentication schemes. In each hop, authentication information must be transmitted that adds to the overall amount of transmitted data. It shows the clear advantage of the *Canvas* scheme, which produces less data overhead than a signature scheme. Especially when the output of the MAC generating function is truncated, only a small overhead is produced. This option is not available for signature schemes, where the signature always has to be transmitted in full length. However, MAC truncation means that the security level is effectively reduced.

6.3.2 Multiple Messages Overhead

When using a signature scheme, the certificate data only has to be exchanged once between a pair of nodes. This allows to amortize this additional overhead if multiple messages are being exchanged. This is illustrated in Figure 6.10. In this example, if more than seven messages are exchanged, the signature scheme is advantageous over *Canvas* when MACs are transmitted in full length. If

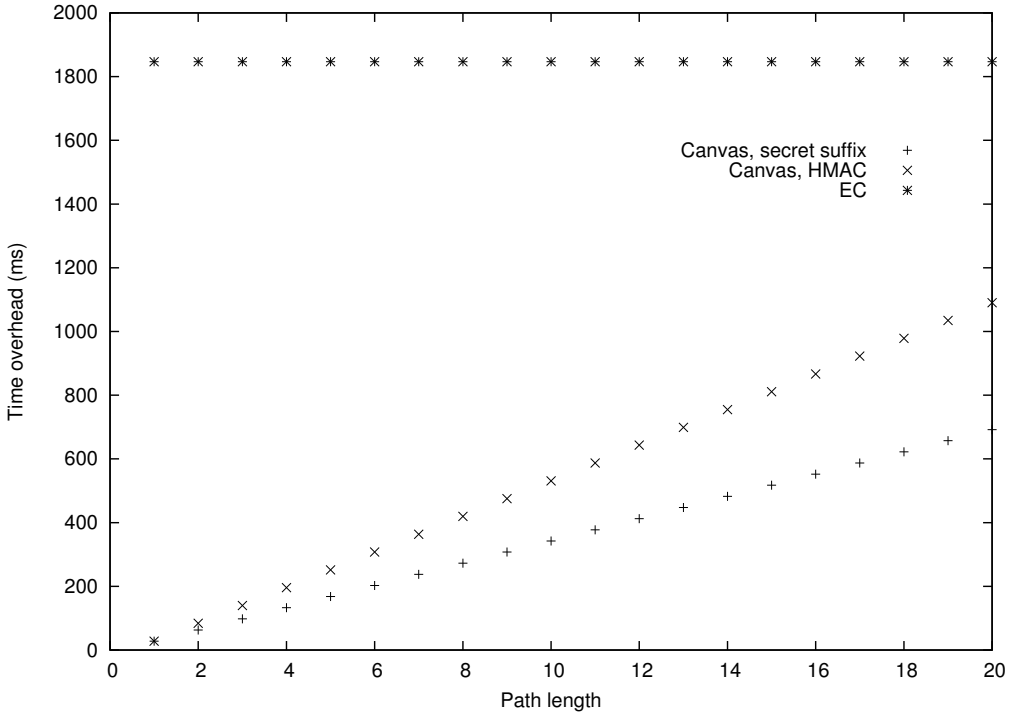


Figure 6.7: Time overhead for a 64 byte message

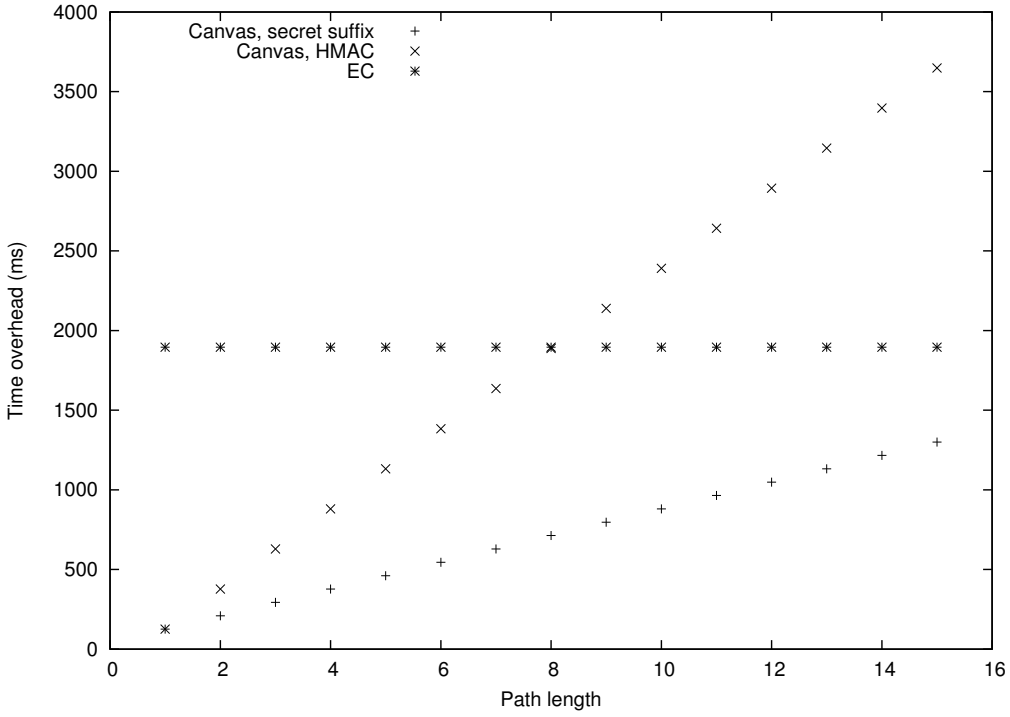


Figure 6.8: Time overhead for a 512 byte message

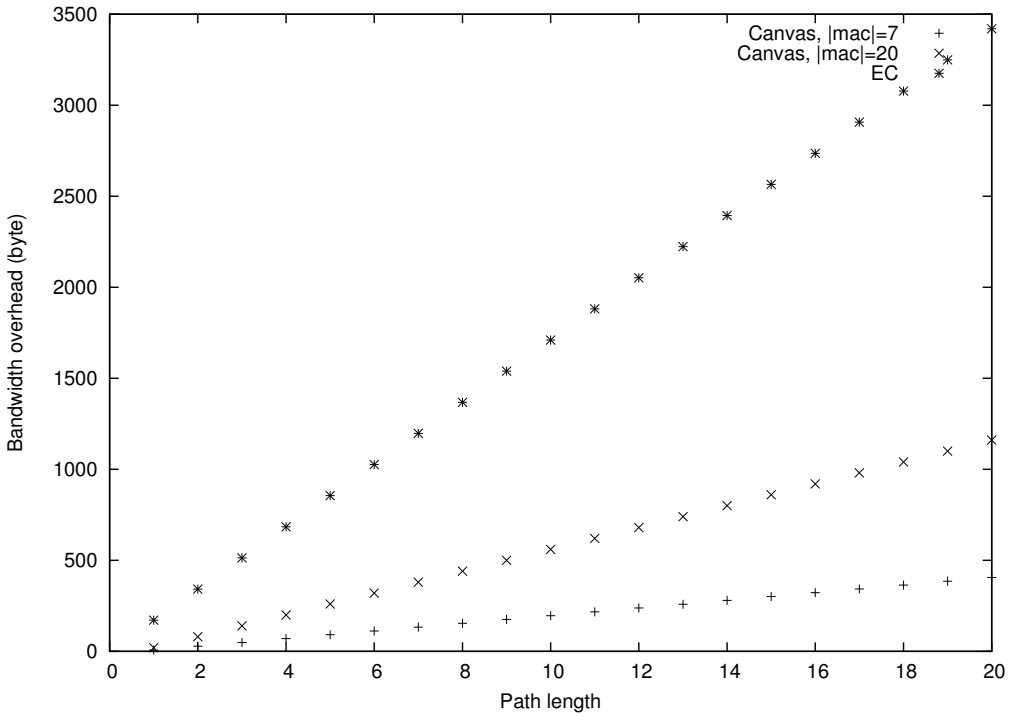


Figure 6.9: Bandwidth overhead

MACs are truncated, the advantage of *Canvas* is retained, however.

6.4 Security Evaluation

Before we actually present these results, we give a brief overview of the method that has been used to retrieve the numerical values for these comparisons.

6.4.1 Analytical Assessment of Resilience

We introduce here the function $B(x)$, which takes the number x of compromised nodes as input and returns an approximation of the number of compromised paths. It is obvious that this approximation can be very rough only. The number of compromised nodes says nothing about the topological position of the nodes within the network. Each position differs in how many paths it will help to compromise.

As a network model, we will use a random geometric graph. We loosely assume it has rectangular shape, and we ignore border effects.

The suffix E for $B()$ describes an approximation if end-to-end authentication is being used. In this case, only those paths are considered compromised that have at least one compromised end node. For each compromised node, there are on average $2(N-1)$ paths to other nodes that are considered compromised

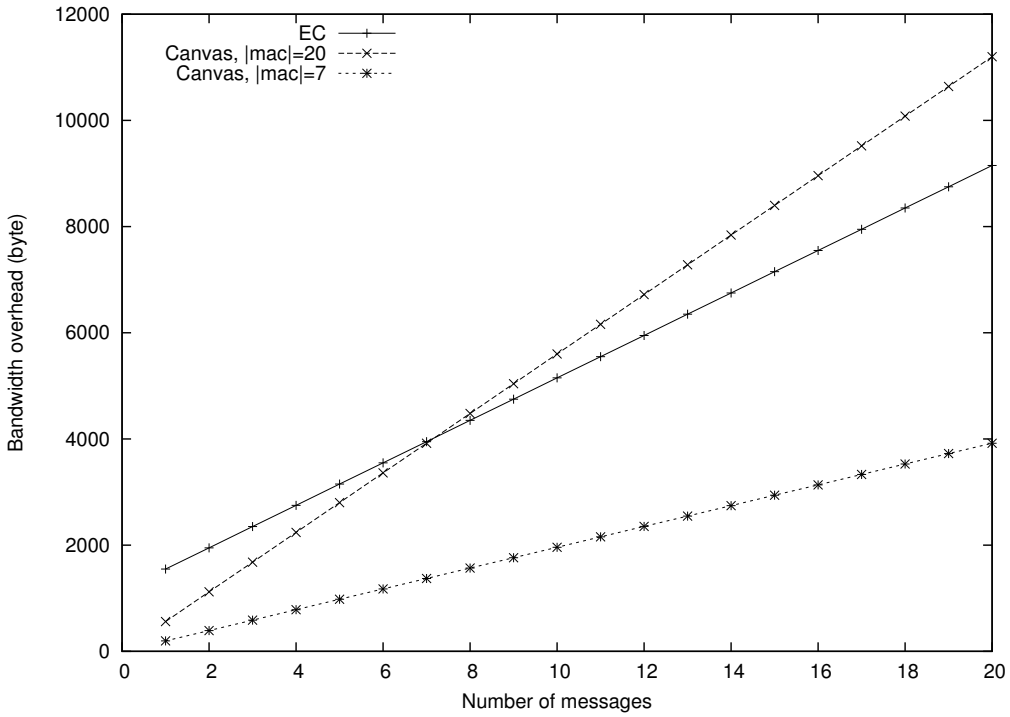


Figure 6.10: Signature bandwidth overhead amortization

(one incoming and one outgoing path to every other node). Counting these paths, we get

$$B_E(x) = \sum_{j=1}^{x} 2(N-j) = 2Nx - x(x+1) \tag{6.1}$$

as the total number of compromised paths, adding in each step the number of paths to the yet uncompromised nodes.

When using *Canvas* for message authentication, we use the suffic C. For this approximation, additional parameters are required.

First, we determine the average distance D to a neighbour node, which will be used later to determine the hopcount of an average path. Let r be the radio range. The probability of finding a node in distance $q \leq r$ is proportional to the circumference of a circle with radius q. In order to compute the average distance, we first sum up over all distances being weighed by the corresponding circumference. This sum is then averaged over the total area, which yields the average distance of a node to the center:

$$D = \frac{\int_0^r 2\pi qq\, dq}{\pi r^2} = \frac{\left[\frac{2}{3}\pi q^3\right]_0^r}{\pi r^2} = \frac{2}{3}r \tag{6.2}$$

Next, we determine the average hopcount of paths. According to [194], two randomly picked points on a unit square on average have distance $\Delta(2) \approx 0.52$. Let W be the side length of the square that constitutes the deployment area for

a sensor network. In terms of hop count, we get an average path length of

$$L = \Delta(2)W/D \approx .52W/D$$

hops.

Now we can estimate the number of paths going through a single node as follows. The total number of paths in the network is $N(N-1)$ and the average path length is denoted as L. To each position on each path, a node needs to be assigned, thus the total number of such assignments is $LN(N-1)$. These assignments are assumed to be equally distributed over all nodes, disregarding border effects, so there are

$$\frac{LN(N-1)}{N} = L(N-1) \tag{6.3}$$

assignments per node, which equals the number of paths that each node is part of. We observe that border nodes are part of fewer paths than central ones. Since we are interested in an average case analysis only, we will not deal with such effects.

Under the *Canvas* authentication regime, a path with non-compromised endpoints is compromised only if there are two adjacent compromised nodes somewhere on the path. Whenever a new node is being compromised, all of its paths that are shared with compromised neighbours will also be compromised. The number of newly compromised paths is thus dependent on the number of compromised neighbours. We can assume that a fraction of x/N of the neighbours are already compromised. Recall from Section2.5.3 that d denotes the density of the network. Thus, $L(N-1)/d \cdot xd/N = L(N-1)x/N$ new paths are compromised. But we must be careful not to count already compromised paths since a good deal of them might already be compromised by other pairs of nodes. Therefore we introduce another factor, c, which is the fraction of so far non-compromised paths:

$$c = 1 - \frac{B_C(x)}{N(N-1)} .$$

Through the multiplication by c we ensure that we count only a certain fraction of the newly compromised paths. Finally, this yields the following equation that describes the number of compromised paths when there are x compromised nodes:

$$B_C(x+1) = B_C(x) + c\left(2(N-1) + \frac{L(N-1)x}{N}\right) \tag{6.4}$$

Figures. 6.11 and 6.12 show the approximation given in equation 6.4 compared to the simulation of *Canvas* on the standard square. $N = 500$ and $N =$

1000 nodes with a communication range $r = 100$ have been used, the simulated attack was a random spread attack. The figures show the absolute and relative errors of the approximation compared to the simulation result.

For smaller numbers of compromised nodes, the approximation and the simulation results match quite well. Both the absolute and relative errors increase but are within acceptable limits – for $x < N/4$, the relative error remains below 0.1 in both cases. For higher x, the error increases slowly but remains below one until $x \approx \frac{3}{4}N$. Beyond that threshold, the relative error increases rapidly. The approximation overestimates the number of non-compromised paths, leading to inaccurate results for high values of x. However, the absolute error remains in the order of magnitude of the actual approximated values, so the approximation is qualitatively still useful.

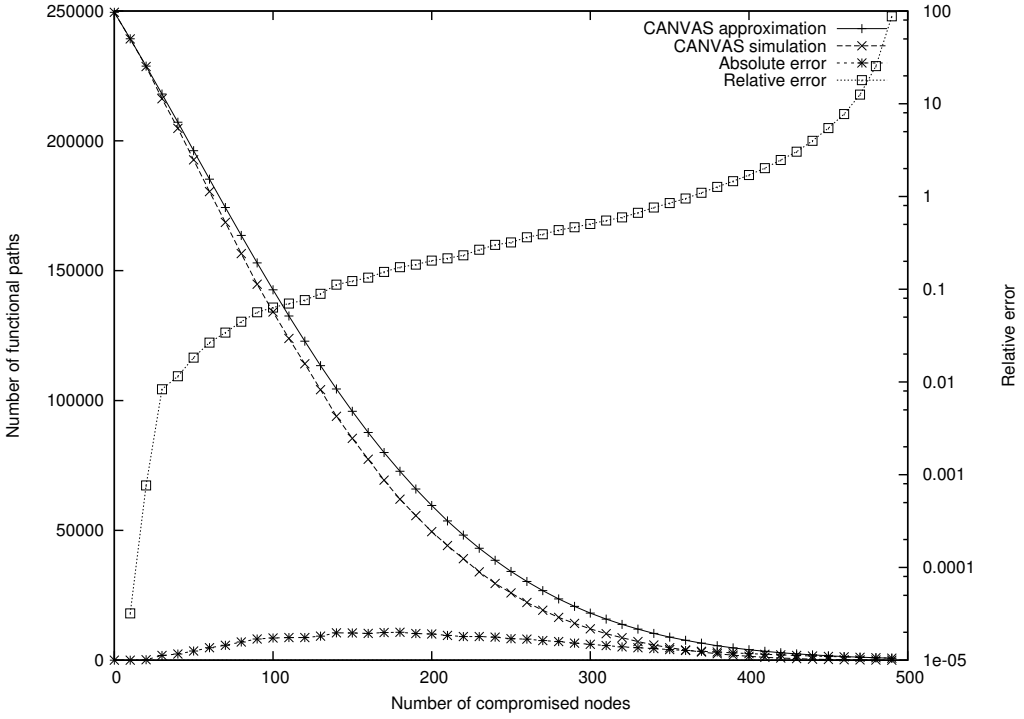


Figure 6.11: Precision of approximation compared to simulation, $N = 500$

To conclude, Equation 6.4 gives a good estimate of the number of functional paths (or, equivalently, of the compromised paths) for smaller x assuming a random spread attack. For other types of attacks we expect the approximation to be much less accurate. For structured attacks, the impact of a compromised node highly varies with its position relative to other compromised nodes. Especially the partitioning attack achieves a big impact with only a small set of nodes.

Assuming a k-connected graph, in the worst case (from the network operator point of view), the k nodes in the cut-off set and some of their neighbours are captured. This leads to a partitioning of the network. This could be achieved

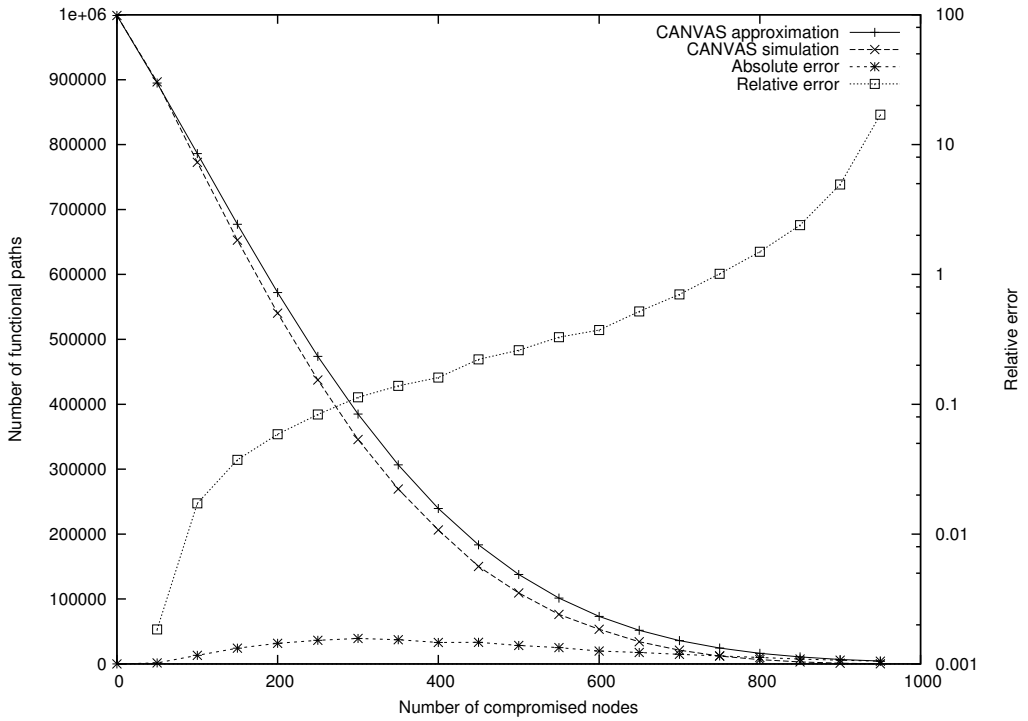


Figure 6.12: Precision of approximation compared to simulation, $N = 1000$

by capturing $k + kd/2$ nodes, where d is the average node degree. Assuming the partition has two approximately equally sized parts, there are about $N/2$ nodes in each of it. This leaves about $2 \times N^2/4 = N^2/2$ secure communication paths in total, i.e. half of the total number of communication paths. However, half of the secure paths are fully contained in one block of the partition, and the other half in the other block. All paths between both blocks are compromised. If important messages flow from one block to the other, they are prone to manipulation.

6.4.2 Numerical Approximation

A node capture attack proceeds by successive compromise of nodes. Thus, the status of a node can either be non-compromised or compromised. A *configuration* defines for each node its status. Thus, it is a mapping $C : N \rightarrow \{\texttt{legitimate}, \texttt{compromised}\}$. For brevity, we write $C(x) = \{s \in N : C(s) = x\}$, where $x \in \{\texttt{legitimate}, \texttt{compromised}\}$.

An *attack* is a sequence of configurations $C_0, C_1, \ldots, C_n$. The initial configuration contains no compromised nodes, i.e. $C_0(\texttt{compromised}) = \emptyset$. All subsequent configurations build monotonously on the previous one, such that $C_i(\texttt{compromised}) \subset C_{i+1}(\texttt{compromised})$. The nodes whose status is newly set to `compromised` in each step are chosen according to the different types of attacks, which are described in Section 3.3.6. For example, in a random spread

attack, the selection probability is equal for all nodes, while in a clustered attack, nodes that are spatially close to the center of attack are selected.

By using a simulation tool, which has been designed and built for this specific purpose, we are able to simulate such attacks. For each configuration that occurs during a simulated attack, using the tool we evaluate the statuses of the communication paths, which depend on the statuses of the nodes. Thereby, we can count the number of paths that are live or functional. This yields the measure described in Equation 3.1.

Using the information on the status of communication paths, we are then also able to compute the set of nodes that are, at least in principle, able to consent (cf. Section 3.5.3).

Simulation will yield only an approximation to the exact mean value for each of these measures. However, this seems to be acceptable since variations will occur in practice. Figure 6.13 shows Ψ_i for five simulation rounds. It is easy to see that there is a certain jitter in the level of security that the *Canvas* scheme provides, depending on the actual course of an attack. We are interested in the mean level, and from the same figure it seems likely that the mean is indeed within the interval defined by these sample simulations. Thus, we should be able to approximate the mean by taking the average of a sample.

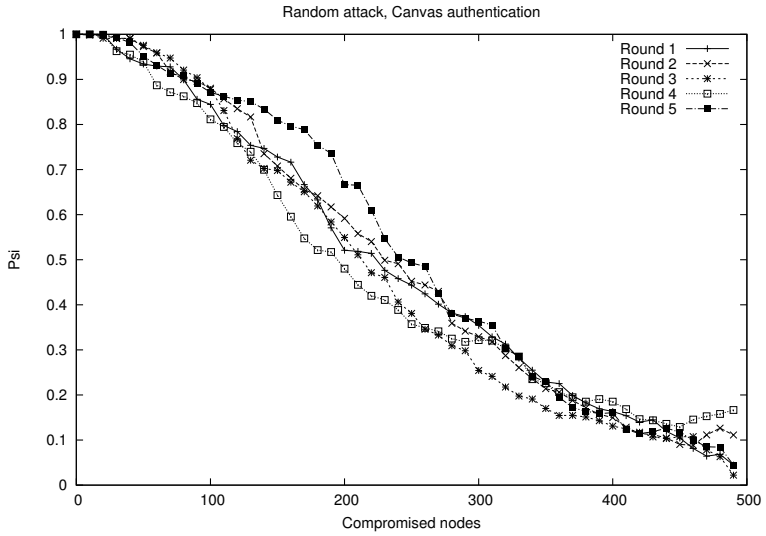


Figure 6.13: Sample simulation rounds, exhibiting the jitter of the security level

For the sake of saving simulation time, we prefer small samples. However, a minimum level of confidence is required. In order to calculate the required sample size, we consider the t-distribution [195], which approximates the Normal distribution if mean and variance are unknown, and only sample data is available.

We assume that a confidence level of at least

$$p \approx 0.80$$

is desirable. Further, we assume that an approximation that lies within an interval of size

$$\varepsilon = 2.7 \cdot sse \ ,$$

where *sse* is the standard error of the sample, is acceptable. These parameters are chosen such that they can be obtained by performing 10 to 20 simulation rounds for each sample. Taking into account that performing a simulation round takes considerable time, this seems reasonable. For example, observing 50 configurations from a random spread attack in a network with 500 nodes, takes about 4.5 minutes on a state-of-the-art laptop computer. A smaller interval and a higher confidence level at the same time would hardly be achievable using this method. Using the sample size, we would have to accept an interval size of $\varepsilon \approx 3.5 \cdot sse$ for a confidence level of $p = 0.90$. (Exact values for the interval size, given a sample size and a confidence level, can be obtained from a t-distribution table, which can be found in many textbooks on statistics.)

As a baseline, we use the respective measures for end-to-end cryptographic schemes. These are easy to obtain as they depend only on the number of compromised nodes:

- The number of functional paths equals the number of live paths for end-to-end schemes.
- Unless more than $1/3$ of all nodes are compromised, all uncompromised nodes are consensus-enabled.

6.4.3 Simulation Results

This section examines the effectiveness of the *Canvas* scheme under various attack types and in different network topologies. It is assessed based on the measure introduced in Section 3.5.3: the number of functional paths remaining in the network, and the number of nodes still able to engage in a Byzantine agreement protocol. The reference on the lower end is a simple link authentication scheme, also called hop-to-hop authentication. This scheme provides no protection against manipulations by single nodes, thus a single captured node degrades the integrity of all communications that pass through it. On the upper end, an end-to-end authentication scheme is assumed, such as one based on public-key signatures. Such schemes provide the highest security level achievable, where the integrity of a communication exchange is solely under control of the endpoints.

The standard topology being used for simulations is a 1000×1000 units square on which 500 sensor nodes are randomly and uniformly placed. The nodes have a communication range of 100 units. This choice of parameters guarantees almost certainly, i.e. with a probability greater than .99, full connectivity of the graph according to equation (2.1), whereas border effects are being ignored.

Simulation results have been obtained by taking the average of a set of simulation runs. As discussed in the previous section, each such set contains data from usually 20 or more simulation rounds.

The simplest attack model is a random spread attack, where each node being captured is randomly and independently chosen. Figure 6.14 shows the effect of such an attack on the absolute number of live and functional paths. The live paths correspond to the functional paths under an end-to-end authentication scheme, which is the best achievable result. The number of live paths deteriorates rapidly under an attack, since each captured node removes a live path for each remaining uncompromised node. With half the nodes being captured, only a quarter of the live paths remain.

The number of functional paths for the *Canvas* and the hop-to-hop authentication schemes are well below the number of live paths if there is a high number of compromised nodes in the network. This, of course, is expected, since hop-to-hop authentication provides no protection against malicious nodes at all. Thus, the difference to the end-to-end scheme is dramatic. *Canvas* is able to perform much better if there are few compromised nodes, but as this number grows, more of them will be placed adjacent to each other and thus become able to undermine the protection provided by *Canvas*.

The strength of *Canvas* is with a small number of compromised nodes, against which it is quite effective. *Canvas* is able to keep the number of functional paths close to the live paths, since isolated captured nodes can do no harm to paths of which they are not endpoints. Only when captured nodes are placed close to each other, they diminish the integrity of other paths. In the graph, this effect surfaces when around 50 nodes, i.e. 10% of all nodes, are captured.

While Figure 6.14 shows the absolute number of paths, the picture gets more detailed if the number of live paths is used as a baseline and Ψ is used as a measure, as defined in Equation 3.1. Using this measure, the graph in Figure 6.15 is obtained. Here, the advantage of *Canvas* compared to hop-to-hop authentication is visible even more strikingly. While *Canvas* manages to keep Ψ close to one for a small number of compromised nodes, it rapidly falls for the hop-to-hop scheme right from the start.

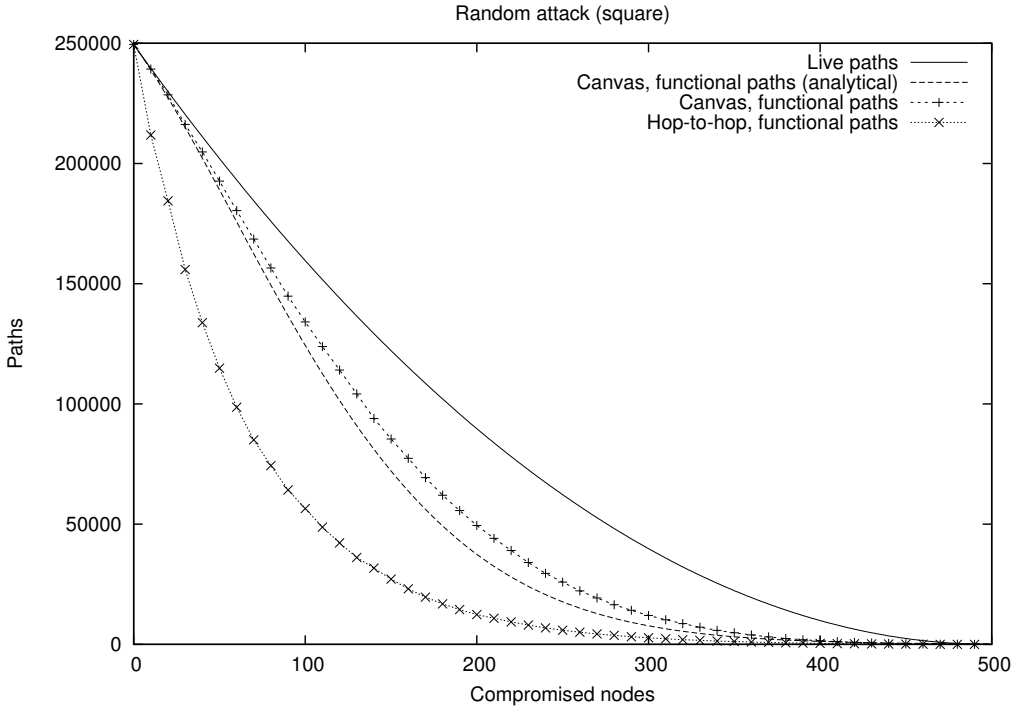


Figure 6.14: Deterioration of communication paths under a random attack; measured by absolute count of paths

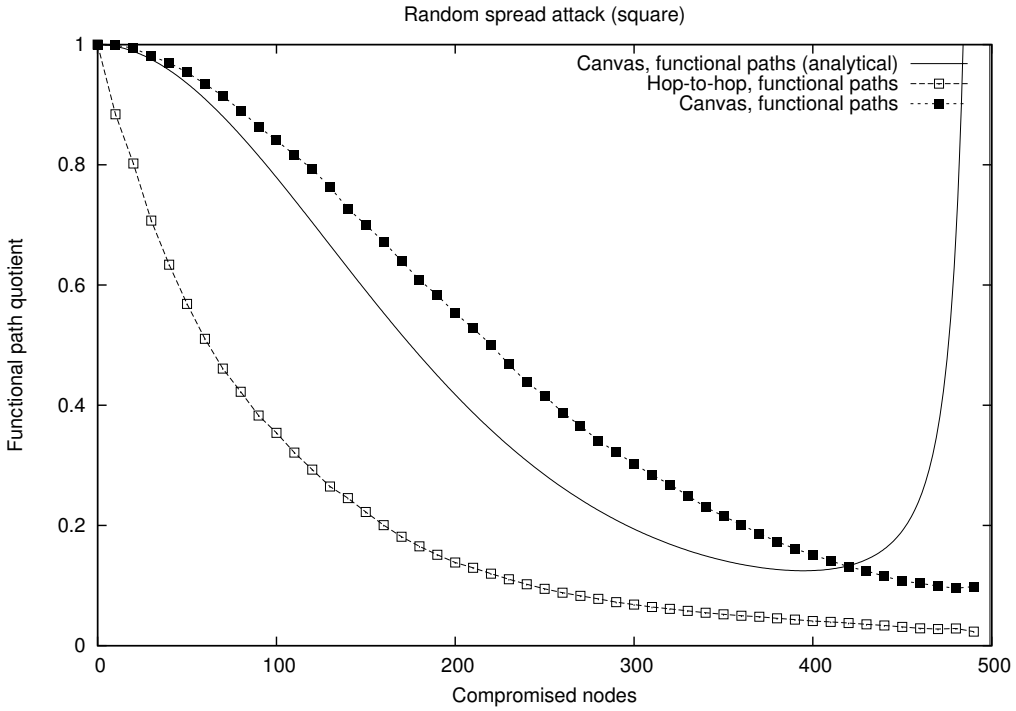


Figure 6.15: Deterioration of communication paths under a random attack; measured by Ψ

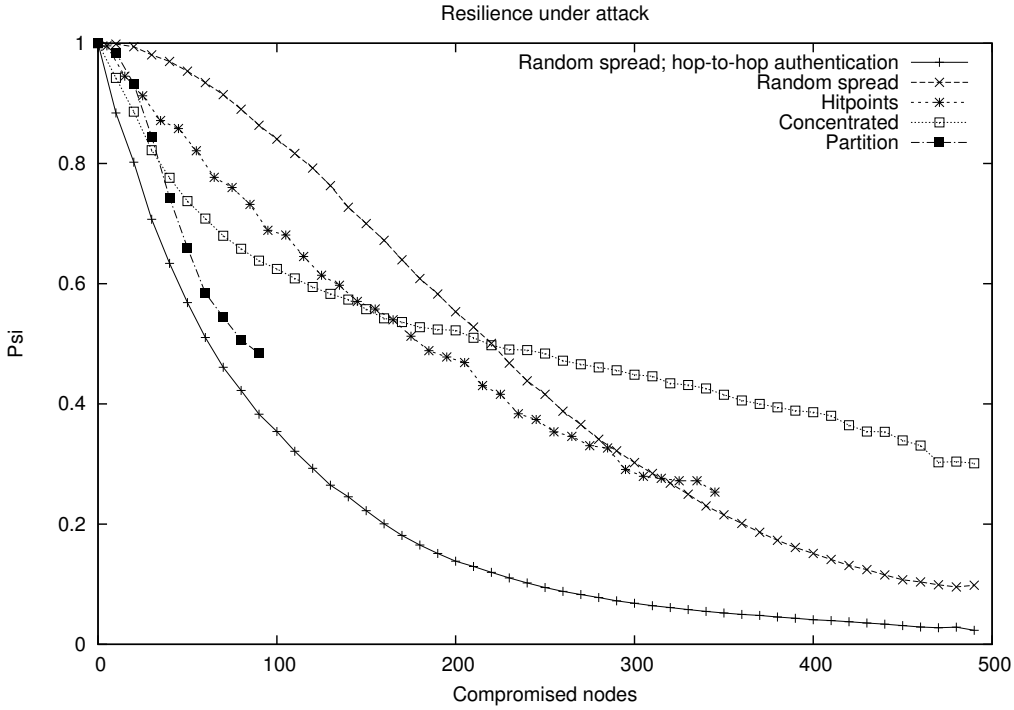


Figure 6.16: Resilience of *Canvas* under various attacks

While *Canvas* is able to maintain the integrity of many communication paths under a random spread attack, at least for small attack sizes, it is more difficult for attacks where captured nodes are arranged into some structure. If compromised nodes are clustered, they can manipulate many more paths.

Figure 6.16 shows simulation results for all four attack types. The partitioning attack, which is the most structured attack, has the highest impact on the communication security. It is very effective in breaking the protection provided by *Canvas*, as it is designed for this purpose. Under this attack, *Canvas* provides only a very small advantage compared to link-only authentication.

Under other attacks, *Canvas* performs much better, but it is clear that structured attacks are very effective even for a small number of compromised nodes. A "concentrated" and a "hitpoint" attack will also reduce the number of functional paths quickly. The concentrated attack is more efficient for a small number of compromised nodes than the hitpoint attack. This shows that a big cluster of nodes controlled by the attacker is more effective than distributed pockets of nodes. However, if more nodes can be compromised, it is better to distribute them in smaller clusters. In general, the effectiveness of these two attacks is reduced when more nodes are compromised – the benefit of an additional compromised node is reduced. This suggests that there are areas in the network that are hardly affected by the progressing attack.

Another way of assessing the security of a network under attack is the abil-

ity of the network to reach consensus among the uncompromised nodes. As long as more than 2/3 of the nodes are uncompromised, consensus is possible. However, only those nodes that are able to communicate securely to the uncompromised nodes are able to draw the same conclusion. The number of nodes that are able to do so serves as the second security measure. Using an end-to-end scheme, all uncompromised nodes would be able to reach agreement if no more than 2/3 of the nodes are compromised. As soon as this threshold is exceeded, no agreement is possible anymore.

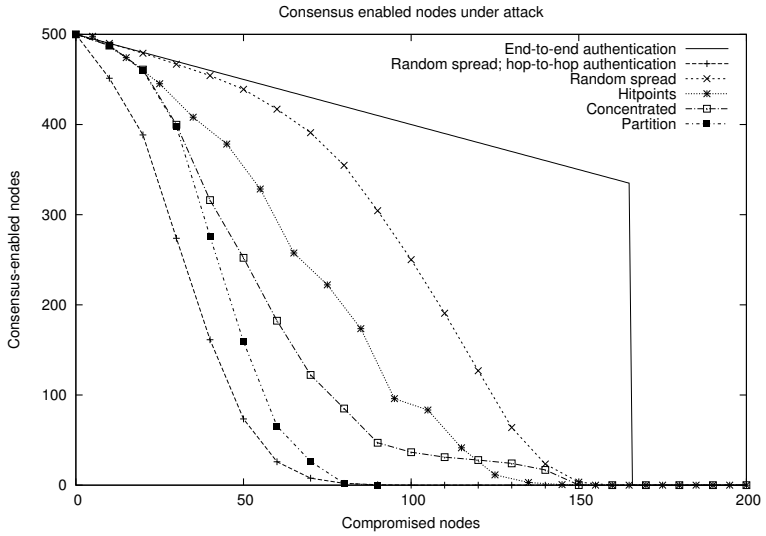


Figure 6.17: Consensus-enabled nodes under different attacks

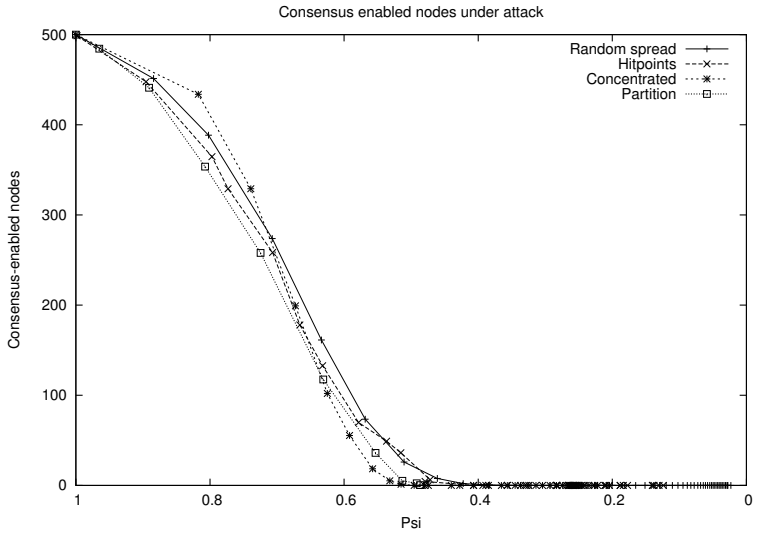


Figure 6.18: Correspondence between Ψ and consensus-enabled nodes

Figure 6.17 shows how *Canvas* behaves with regard to consensus-enabled nodes under different attacks. For comparison, the behaviour of hop-to-hop authentication under a random spread attack is also included.

This picture corresponds well with the functional path quotient Ψ shown in Figure 6.16. Indeed, as Figure 6.18 illustrates, there is a strong correspondence between the two security measures. This correspondence can be explained by the dependency of consensus on the availability of functional communication paths. In an end-to-end scheme, this dependency is not given – all paths with uncompromised endpoints are considered functional.

6.4.4 Addressing Message Injection

As laid out in this chapter, the *Canvas* protocol can only be applied if one is willing to accept a certain risk of message manipulation. Another risk is message injection. The authentication strength of *Canvas* could be considered "weak", in contrast to end-to-end authentication as two compromised nodes could, at any time, create a new message, both sign it, and pass it on to another node. This (legitimate) node would accept the message and pass it on to its destination. This, of course, could lead to an overwhelming amount of fake messages in the network.

The legitimate fraction of messages could sink below a threshold, where its impact on the operational outcome of the network would become neglibile. This can quickly render data collecting applications useless, since the most data they receive (and pass on) would be garbage.

The amount of fake messages, which can be produced by a set of compromised nodes, can be limited by a number of means. They all impose certain rules on the behaviour of nodes. Any violation of these rules would lead to attack detection. Since it may not be clear which node exactly is compromised, this could be considered a weak form of intrusion detection.

Limit Sending Rate

One of the immediate countermeasures against heavy message injections is to limit the rate by which a node can issue messages. A node receiving more than a threshold of messages from a single source would immediately stop accepting them and issue a warning. This limits the overall fraction of fake messages. It also limits the impact of fake messages that are received by a single node.

Clocked Transmissions

A simple model avoids, by definition, message injection. In this model, message transmission is only possible in certain time slots. Time slots are arranged in phases that are (loosely) synchronized throughout the network. The duration

of the phase allows that each node can send exactly one message to each of its neighbours. The model also demands that in each slot a message must be sent. By construction, this forbids the possibility of message injection.

Message Sequence Numbers

Any node issuing a message must attach a sequence number to it. A node receiving multiple messages from the same source would then be able to notice the existence of injected messages if the same sequence number is used twice.

A message injection attack might be successful for a certain period of time, until the original source starts emitting messages to the same recipients as the attacker. If one of them reaches a recipient, the attack is detected.

Source Address Verification

We describe two approaches to source address verification: one that relies on the cooperation of the source's neighbours; and a second one that challenges the source on a random basis. Both procedures could be combined and thus further reduce the risk of message injection.

The first approach is based on monitoring. Whenever a new message is created, it must be broadcast. This ensures that a number of nodes get a copy of the message. In many cases, broadcast channels are used by default, thus no additional changes are required.

When the second node passes the message on, it also has to broadcast the message. This adds more nodes to those that know the message. Now, a consensus protocol is executed between the knowing nodes. Only if an agreement can be achieved, the third node on the path is allowed to accept the message.

This approach has a number of drawbacks. First, potentially many nodes are participating in the consensus protocol, which implies a significant effort. Second, it only works if there are neighbours available. A third drawback is the requirement that messages must be sent in cleartext, i.e. without a link layer encryption step, at least for the duration of the agreement phase.

The second approach requires additional message exchange between the source and the receiver. The receiver of a message may send a challenge to the original sender. This challenge contains part of the received message such that the sender can verify if the sender is the source of the message. If this is the case, the sender returns an acknowledgement. Only if the acknowledgement is correctly received, the receiver will accept the message.

The challenge is useful in another way, too. If a node gets a challenge about a message that it has not issued, this indicates the presence of an attacker. This

information can be used in the quality assessment of the computed result.

6.4.5 MAC Security

Canvas requires that in general each node verifies k MACs before accepting a message. An attacker who manipulates a message has to make sure that all k MACs will be accepted by the receiver. A single compromised node would therefore have to manipulate the message and the MACs in either of the following ways. Assume that K is the key used for authentication. The adversary must either create a manipulated message m' such that $MAC(m',K) = MAC(m,K)$, i.e. perform a second preimage attack. Or, the adversary may exchange the authentication code a for a new one a' such that $MAC(m',K) = a'$, i.e. perform a preimage attack.

Both preimage and second preimage attacks are considered hard for good hash functions. For a hash output length of n bits, 2^n guesses are required to come up with a correct match. This makes it practically infeasible to find preimages if n is sufficiently large. As an alternative to finding a hash preimage, the adversary could try to recover the key K by an exhaustive search. This will also be infeasible if the key is of sufficient length.

The choice of the MAC length and the key length determines the security level. Depending on the resources available to an adversary, a value between $n = 56$ (which equals the security level of the DES cipher) and $n = 80$ is usually considered secure. However, with technology improvements, such attacks become cheaper and more feasible. For example, DES keys can be recovered within nine days at a cost lower than 10,000 USD [101].

HMAC is a very secure construction for a MAC as it can mitigate weaknesses of the underlying hash function. Even if the hash function is vulnerable to a second preimage attack, such an attack cannot be applied to HMAC since the key K is prepended to the actual message. Unfortunately, the prepending of the key makes using HMAC inefficient for *Canvas*. Since for each MAC being generated, a different key is prepended and thus in fact a new message is created, the complete message has to be hashed separately for each MAC. It would be more efficient if the message could be hashed only once and each key is applied to the result.

The *secret suffix* construction for MACs allows to first create a hash of the message independent of the key. The MAC creation in Algorithm 5 would be implemented as

$$\{A,P,m,c_{XV}\}_{KXV} = h(A\|P\|m\|c_{XV}\|K_{XV}) \ .$$

The secret suffix construction has the disadvantage that if the underlying hash function h is not resistant to finding second preimages, a MAC can be easily constructed for forged messages. This is particularly undesirable since the attack can be executed offline, i.e. no interaction with the receiving node is necessary. The original message m is known, so the attacker can use this knowledge to search for a message m' that yields the same hash output as m. The attacker can then transmit m' and the original MAC a. The verification done by the receiver will be successful.

It is not known whether modern cryptographic hash functions, such as the family of SHA functions, is indeed resistant to second preimage attacks. Weaknesses are continously showing up but it hasn't been demonstrated yet that arbitrary second preimages can be easily found. SHA-1 has been found to be weak against collision attacks [188], meaning that pairs of messages m_1 and m_2 can be found that yield the same hash value. However, this does not immediately lead to a vulnerability regarding second preimages, which requires to find a second message that yields the same hash value as a given message. Thus, we can safely use SHA-1 as an example hash function. However, any other hash function can be used in the *Canvas* scheme as well.

It is important to note that public key signature schemes are vulnerable to the same preimage attack as the secret suffix MAC if the signature is created on the hashed message. Only if the signature is created on the message directly, this attack does not apply. However, this is only possible if the message is short enough. Thus in general, hash-based signatures are applied. To conclude, the secret suffix method will yield a similar security level as a public key signature scheme.

6.4.6 Example: A Dynamic Application Scenario

Figure 6.19 shows the layout of a building with a number of rooms and a hall connecting these rooms. A possible application of a wireless sensor network in such a scenario is the reporting of sensor data to a guard walking through the hall. Sensor nodes are distributed throughout the area. The sensor data is reported to the node that is closest to the guard. It is assumed that the guard walks straight through the hall and collects data at regular intervals, twenty times in total. Each time, a randomly selected node from each room sends a message towards the location of the guard.

Figure 6.20 shows the number of non-tampered messages that are obtained by the guard while the network is under a random spread attack. With a small number of compromised nodes (up to 50, in this case), more than 80% of the

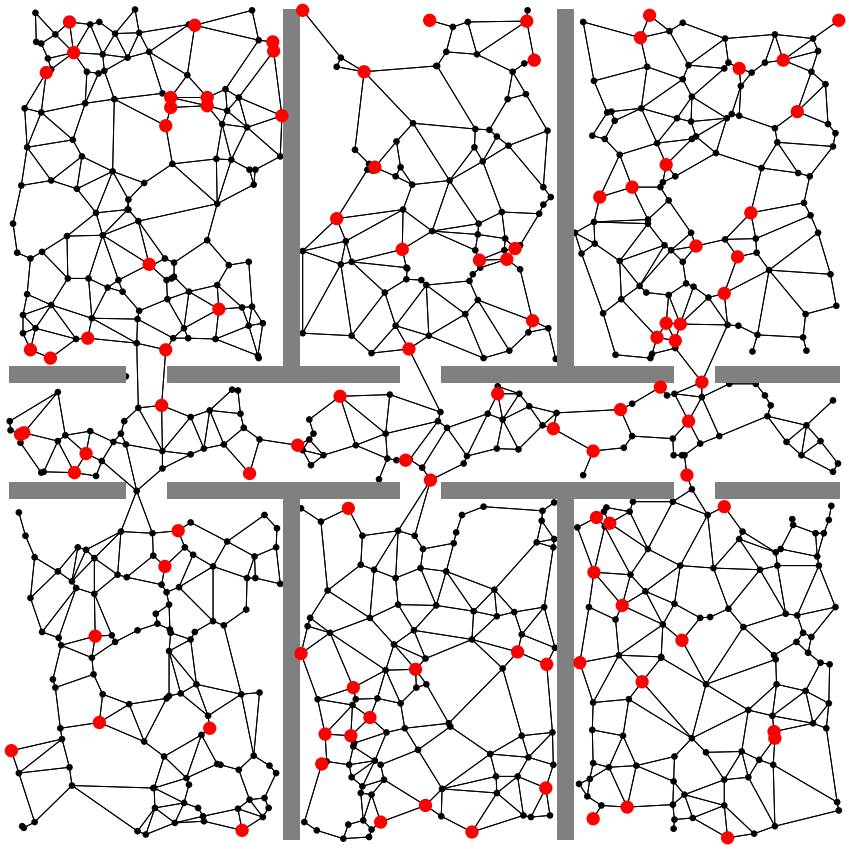


Figure 6.19: Rooms scenario for sensor data collection; 100 nodes captured

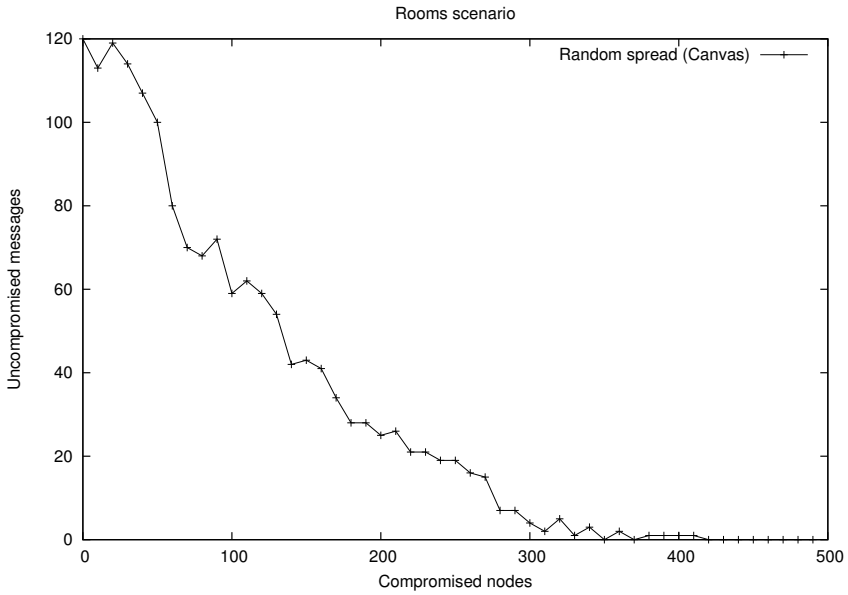


Figure 6.20: Number of non-tampered messages obtained during a walk

messages are correct. With more nodes being compromised, this rate quickly falls. This is due to the fact that the "doors" in this scenario are a bottleneck through which all messages have to pass. If the attacker happens to select nodes in such an area, all messages from the respective room will be subject to manipulation.

6.5 Extended Interleaved Authentication

The simple Canvas scheme provides basic protection against message manipulations that are carried out in unstructured attackes. But a highly structured attack, such as a partitioning attack, can render the Canvas scheme ineffective. We will now show that by introducting shortcuts in the authentication graph, structured attacks can be mitigated as well. They effectively force the attacker to become active (i.e. compromise nodes) in the close vicinity of the target location.

6.5.1 Protocol Description

We describe here a protocol that extends the basic *Canvas* scheme, i.e. all checks that are made when using *Canvas* only are made here as well. The extended protocol provides an additional layer of security that specifically addresses the weakness of *Canvas* that a cluster of compromised nodes can manipulate all messages that pass through nodes in this cluster.

The proposed scheme extends the basic *Canvas* scheme in the following way. Each node maintains a (small) number of security relationships with nodes that are not within its k-hop neighbourhood but distributed throughout the whole network. These long-range links are used for authenticating messages that are sent to remote locations. Having a direct security relationship between the source of the message and a node that is close to the receiver, large, potentially compromised, parts of the network can be bypassed. An attacker will only be able to tamper with messages if he is active within close range of the receiving node.

The long-range security relationships of nodes will be called "shortcuts", and the remote peer of a node will be called a "shortcut node". The security of this extended scheme is based on two principles:

1. If the distance between the sender of a message and its target location is less than δ, i.e. $d(A,P) \leq \delta$, it is sufficient to *Canvas*-authenticate the message. This is based on the consideration that if the attacker is active in the vicinity of the target location, there is a certain probability that

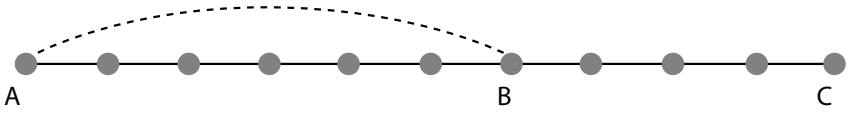


Figure 6.21: A shortcut path

the node eventually processing the message is compromised as well. An additional authentication path through a shortcut would add no security in such a case.

2. For longer distances, the sending node A has to select a shortcut node B that is close to the target location P. This means that if $d(A,P) > \delta$, a node B is selected with which A shares a shortcut key and for which $d(B,P) < d(A,P)$ and $d(B,P) \leq \delta$. A attaches a MAC using the shared key between A and B and the message is then first sent to B. After B has checked the validity of the MAC, the message is forwarded to P using *Canvas* authentication. This procedure ensures that clusters of compromised nodes on the path from A to B are not able to manipulate the message.

The parameter δ limits the distance over which messages may be sent with *Canvas* authentication only. This effectively limits the impact of clusters of compromised nodes. Messages passing through such a cluster are subject to manipulation. However, δ ensures that such manipulations manifest themselves only in the vicinity of the compromised cluster.

In the following, we assume that a message is addressed to a *target region* instead of a specific node. A target region is specified by a geographic location and a radius τ. Any node with a maximum distance of τ from the target location is a valid recipient of the message.

The resulting path of a message from source node A to a target node C may look as depicted in Fig. 6.21. Note that in this figure, the underlying *Canvas* authentication layer is omitted. It's obvious that the resulting path, which has to include shortcut node B, may be longer than the direct path from A to C. However, as we will see later, this must not be the case and the resulting path may even be shorter.

The rules for communication based on shortcut authentication are listed in Table 6.4. The rules describe the behaviour of a node X receiving a message M that originates at node A. There are two message formats being used, tagged with either SHORTCUT or DIRECT.

A SHORTCUT message is one that is being forwarded to node B because B is a shortcut for A. It carries an authentication code and a message counter for

B. Nodes different from B simply forward this message after the verification of the attached *Canvas* authentication information C (rule *shortcut-accept-and-forward* applies).

For the shortcut node B, there are two rules. Which one of them applies depends on the condition whether B itself is the target of the message. If condition $d(B,P) \leq \tau$ holds, B itself is qualified to process the message, since it is close enough to the target location. In that case, rule *shortcut-authenticate-and-process* applies. Otherwise, rule *shortcut-authenticate-and-forward* is applied: B converts the message into a DIRECT message and forwards it further to the target location. In either case, the verification conditions must be fulfilled, i.e. the purported counter must be greater than the stored value, and the authentication code must be valid.

A DIRECT message is targeted at its final destination. A node that encounters such a message accepts it only if its own location is close enough to the target of the message. Depending on whether the node itself qualifies as a receiver, the message is either immediately processed (rule *direct-accept-and-process*) or forwarded toward the target (*direct-accept-and-forward*). In both cases, the *Canvas* information must be verified.

As introduced above, the parameter δ is used for deciding whether a DIRECT message will be further forwarded. This has the effect that messages that are only authenticated using *Canvas* can be sent only within a range δ from a source. Messages with a target location further away than δ are simply discarded.

The number of shortcuts n_S that exist per node must be big enough such that coverage is maintained given parameter δ. On the other hand, n_S should be as small as possible since maintaining relationships to shortcut nodes consumes resources. This leads to the consideration that n_S must be at least, and should be close to

$$n_S \geq \frac{A}{\pi\delta^2} \, .$$

The actual value depends on additional considerations. First, the minimum number can only be achieved if the shortcut nodes are optimally distributed throughout the network. Second, a certain redundancy may be desirable for increased robustness against node failure. Third, the available memory on each node limits the space for shared keys and therefore limits the number of shortcut nodes a single node can support.

Note that the shortcut relation is not necessarily symmetric. A node B that serves as a shortcut to node A does not necessarily use A as one of its own shortcuts. Although such a symmetric relationship is favourable since fewer

shortcut-accept-and-forward:

$M = \langle \text{SHORTCUT}, A, B, c, P, m, a_{AB}, \mathcal{C} \rangle$
$X \neq B$
canvas-accept$(A, P, m, \mathcal{C})$

$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
send: $\langle \text{SHORTCUT}, A, B, c, P, m, a_{AB}, \mathcal{C}' \rangle$

shortcut-authenticate-and-forward:

$M = \langle \text{SHORTCUT}, A, B, c, P, m, a_{AB}, \mathcal{C} \rangle$
$X = B$
$c > c_{AB}$
$a_{AB} = \{A, B, c, P, m\}_{K_{AB}}$
$d(B, P) \leq \delta$
$d(B, P) > \tau$

$c_{AB} := c$
$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
send: $\langle \text{DIRECT}, A, P, m, \mathcal{C}' \rangle$

shortcut-authenticate-and-process:

$M = \langle \text{SHORTCUT}, A, B, c, P, m, a_{AB}, \mathcal{C} \rangle$
$X = B$
$c > c_{AB}$
$a_{AB} = \{A, B, c, P, m\}_{K_{AB}}$
$d(B, P) \leq \tau$

$c_{AB} := c$
process A, m

direct-accept-and-forward:

$M = \langle \text{DIRECT}, A, P, m, \mathcal{C} \rangle$
$d(X, P) > \tau$
$d(X, P) \leq \delta$
canvas-accept$(A, P, m, \mathcal{C})$

$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
send: $\langle \text{DIRECT}, A, P, m, \mathcal{C}' \rangle$

direct-accept-and-process:

$M = \langle \text{DIRECT}, A, P, m, \mathcal{C} \rangle$
$d(X, P) \leq \tau$
canvas-accept$(A, P, m, \mathcal{C})$

---

process A, m

Table 6.4: Rules for shortcut communication. The current node is denoted as X

data has to be stored at each node, it might not be feasible to set up these symmetric relationships efficiently. Setting up shortcut relationships is the topic of the next subsection.

6.5.2 Establishing Shortcuts

In principle, it is possible to establish shortcuts by physical links. However, this requires the use of additional hardware that exceeds the capabilities of standard wireless sensor nodes. Therefore, we confine ourselves to "virtual" shortcut links, which may span several physical communication links and are manifest only in the authentication graph, i.e. there is a shared key between two nodes involved in a shortcut.

The information, which nodes should act as shortcut nodes to other nodes can be either distributed in the pre-deployement phase of the network, or immediately after deployement. The first approach has the advantage that no further messages have to be sent in order to establish shortcuts. However, the disadvantage is that if nodes are randomly deployed, the locations of shortcut nodes are not known, which induces additional cost for actually sending a message to a shortcut. The second approach requires significant traffic overhead until all shortcuts are established. However, once shortcut nodes are known (including their locations), they can be efficiently used without any additional overhead.

Pre-determining shortcuts can be easily done during the key predistribution phase. The number of shortcuts per node, n_S, is determined according to the anticipated requirements of the concrete WSN deployment. For each node X, n_S other nodes are randomly selected. For each such node, a secret shared key is created and assigned to both nodes, and the identifier of the selected shortcut node is announced to X. If nodes are randomly deployed, it can be expected that the shortcut nodes of some node X are evenly distributed throughout the network. Thus, good coverage of the network should be given.

An alternative approach is a systematic assignment of shortcut nodes after deployment. Here, we describe two methods: one is based on a virtual token ring, the other is a directed selection of regions.

The idea of the token ring based approach is that for each node X, a token is circulated through the network that "collects" shortcut nodes for X. Assuming length l of the ring, after the token has travelled $l/(n_S+1)$ hops, the current node adds its own identifier to the token as another shortcut node for X and resets the hop counter of the token. When the token arrives at X again, enough shortcut nodes are accumulated.

A token ring allows to include all nodes in the network to be equally in-

cluded in the process of shortcut assignment. By adding some tolerance to the number of hops the token travels between assignments, the assignment can be balanced such that all nodes will eventually serve as shortcuts to the same number of other nodes.

There are two major problems with this approach. First, the number of messages that have to travel through the network is very big. A ring can be easily constructed based on a spanning tree, for example using an Echo algorithm, which however leads to a rather inefficient ring structure. For n nodes in the network, such a spanning tree contains $n-1$ edges and therefore the corresponding ring contains approximately $2n$ edges. In the optimal case, the ring has $n-1$ edges, but such a ring is hard to construct. Since each token has to travel along each edge once, in total between roughly n^2 and $2n^2$ messages have to be transmitted. A related problem is the fact that each node has to process at least n messages in total. The second problem is the distribution of shortcut nodes. With the described approach, an even distribution of shortcut nodes cannot be guaranteed.

A more efficient assignment of shorcut nodes, which also provides an even shortcut node distribution, is based on a controlled selection process. Each node X partitions the network in n_S geographical regions and sends a request to the center of each of these regions. The regions have to be constructed in a way such that with one shortcut node in each region, full coverage is provided. By construction, this approach provides an even distribution of the shortcut nodes. Its scalability is also improved compared to the token ring approach. Each node sends n_S request messages, receives the same amount of responses, and each message travels about L hops (average path length in the network). Thus, in total $2nn_SL$ messages are sent.

6.5.3 Long-Range Interleavings

In very large networks, it may be impossible to achieve full coverage with a limited number n_S of shortcuts and a given δ as a message may not be able to reach the δ-neighbourhood of its destination with a single shortcut. However, there might be another node, which is already part of the message path, that has a shortcut in the δ area of the target, or at least closer to it than the previous one. Thus, while a message travels on its path to its destination, additional shortcuts can be used to span the complete distance to the δ-region of the destination.

This idea is depicted in Figure 6.22. Here, two additional shortcuts are being used in order to get the message closer to the target G. The authentication links "monitor" each other: The link between B and E ensures that the message is

not manipulated either by C, D, or any other intermediate node. The last piece of the path is bridged with *Canvas*.

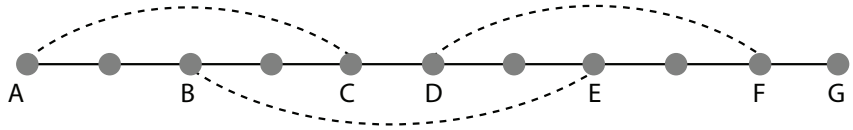


Figure 6.22: A path with long-range interleaved authentication

There is a simple rule according to which the nodes on a path act: Try to attach a long-range authentication code, i.e. a shortcut, that gets closer to the target than the previous one. At any given time, there are zero, one, or two shortcuts attached to a message. There may be a *primary* and a *secondary* shortcut. When the message starts off from the source, it has only one such shortcut, which is the primary shortcut. Following nodes on the path look for own shortcuts that are more closely located to the target than the primary shortcut. If a node finds one, it attaches a corresponding MAC to the message, which becomes the secondary shortcut. If there are two shortcuts already attached to a message and a path node finds a closer shortcut than the secondary one, the secondary shortcut is substituted by this new one. When the message gets closer to the destination, all shortcut MACs will be gradually removed and the message is confined to *Canvas* authentication.

The function to substitute or add a shortcut authentication code, *subst-shortcut,* is captured in Algorithm 6. Part of the input is a "list" of authentication code, γ, which has either zero or one elements. This list does not reflect the primary shortcut of the message, which is targeted at B, but may only contain the secondary shortcut if it exists. If γ is empty and a shortcut node can be found that is closer to the destination than B is, a secondary shortcut for the message is created. If a seondary shortcut already exists but a "better" one is found, i.e. one that is even closer to the destination P, the existing secondary shortcut is discarded and a new one is created. Note that the function *find-closest-shortcut* is not further detailed here. It simply returns the shortcut node of the current node that is closest to P.

Table 6.5 defines the rules for long-range interleaved authentication. The rule *shortcut-authenticate-and-forward* applies if the current node X is the primary shortcut of the message. It is checked whether the primary shortcut authentication code is correct and whether the *Canvas* protocol is adhered to. This rule demands that a secondary shortcut exists for this message (condition $len(\gamma) \geq 1$). This becomes the new primary shortcut for the message, and a new secondary shortcut is attached if one is found. To that end, the function *subst-shortcut* is invoked, which either creates a new shortcut authentication code

or returns the empty list. The message is passed on with appropriate *Canvas* authentication.

The rule *shortcut-accept-and-forward* is the standard forwarding rule for intermediate nodes that are not involved in the long-range authentication protocol. Such nodes simply check for the correct *Canvas* authentication. Additionally, they try to add a better secondary shortcut than the existing one. This is reflected by the invocation of the function *subst-shortcut*. It may happen, of course, that no appropriate shortcut is found, so the set of MACs remains the same.

The next rule, *shortcut-authenticate-and-forward-exit* is similar to the first one. Here, only a primary shortcut exists and the current node is the target of this shortcut. The necessary validity checks for the shortcut are performed, and the *Canvas* authenticity is checked as well. Additionally, this rule applies only if the current node does not qualify as a destination node of the message, i.e. $d(P,X) > \tau$. The message is forwarded towards its destination as a simple *Canvas*-authenticated message.

The rule *shortcut-authenticate-and-process* applies when the current node X is not only the target of the primary shortcut, but X itself also qualifies as a destination node of the message. In this case, the message is consumed by the current node.

The rules *direct-accept-and-forward* and *direct-accept-and-process* from the basic *Canvas* authentication (see Table 6.1) are used in this scheme as well. They are used for handling messages that are close to their destination and are only *Canvas*-authenticated.

The $\odot$ notation should be understood loosely as a list appending operation.

Algorithm 6 *subst-shortcut*

Input:
X: current node
P: destination location
B: next shortcut node on path
γ: list of shortcut authentication codes (length is either zero or one)

Output:
list of shortcut authentication codes, possibly equal to γ

```
1:  sflag := false
2:  W := find-closest-shortcut(X, P)
3:  if len(γ) = 0 ∧ d(W, P) < d(B, P) then
4:      sflag := true
5:  else if len(γ) = 1 then
6:      (Y, V, c', a') := γ
7:      if d(W, P) < d(V, P) then
8:          sflag := true
9:      end if
10: end if
11: if sflag then
12:     c_XW := c_XW + 1
13:     a := {A, X, W, c_XW, P, m}_K_XW
14:     return (X, W, c_XW, a)
15: else
16:     return γ
17: end if
```

shortcut-authenticate-and-forward:

$M = \langle \text{SHORTCUT}, A, P, m, (Z, B, c, a) \odot \gamma, \mathcal{C} \rangle$
$X = B$
$c > c_{ZB}$
$a = \{A, Z, B, c, P, m\}_{K_{ZB}}$
$len(\gamma) \geq 1$
canvas-accept$(A, P, m, \mathcal{C})$

$c_{ZB} := c$
$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
$\gamma' :=$ *subst-shortcut*(X, P, B', ε) where $(\_, B', \_, \_) = \gamma$
send: $\langle \text{SHORTCUT}, A, P, m, \gamma \odot \gamma', \mathcal{C}' \rangle$

shortcut-accept-and-forward:

$M = \langle \text{SHORTCUT}, A, P, m, (Z, B, c, a) \odot \gamma, \mathcal{C} \rangle$
$X \neq B$
canvas-accept$(A, P, m, \mathcal{C})$

$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
$\gamma' :=$ *subst-shortcut*(X, P, B, γ)
send: $\langle \text{SHORTCUT}, A, P, m, (Z, B, c, a) \odot \gamma', \mathcal{C}' \rangle$

shortcut-authenticate-and-forward-exit:

$M = \langle \text{SHORTCUT}, A, P, m, (Z, B, c, a) \odot \gamma, \mathcal{C} \rangle$
$X = B$
$c > c_{ZB}$
$a = \{A, Z, B, c, P, m\}_{K_{ZB}}$
$len(\gamma) = 0$
$d(P, B) > \tau$
canvas-accept$(A, P, m, \mathcal{C})$

$c_{ZB} := c$
$\mathcal{C}' :=$ *canvas-auth*$(A, P, m, \mathcal{C})$
send: $\langle \text{DIRECT}, A, P, m, \mathcal{C}' \rangle$

shortcut-authenticate-and-process:

$M = \langle \text{SHORTCUT}, A, P, m, (Z, B, c, a) \odot \gamma, \mathcal{C} \rangle$
$X = B$
$c > c_{ZB}$
$a = \{A, Z, B, c, P, m\}_{K_{ZB}}$
$d(P, B) \leq \tau$
canvas-accept$(A, P, m, \mathcal{C})$

$c_{ZB} := c$
process A, m

Table 6.5: Rules for long-range interleaved communication

6.5.4 Performance Evaluation

Routing a message not directly to its destination but to one or more intermediate nodes beforehand implies a certain overhead as the intermediate nodes are unlikely to be on the shortest path from the source to the destination. When selecting intermediate nodes, we may demand that the next intermediate node must be closer to the target node than the last intermediate node (or the source node, if the first intermediate is selected). How big the overhead is depends mainly on the number of intermediate nodes to choose from. Figure 6.23 shows the average length of interleaved paths relative to the shortest path length based on data obtained through simulation. If only 1% (5 out of 500) of all nodes are available as intermediates per node, the interleaved path may be almost twice as long as the shortest path. However, as the number of intermediates grows, this overhead shrinks quickly. At 2%, the overhead is about 50%, and at 6% (30 out of 500), it is as low as 20%. We therefore conclude that with a sufficiently large number of relays for each node, the increase in the path length is not a critical issue.

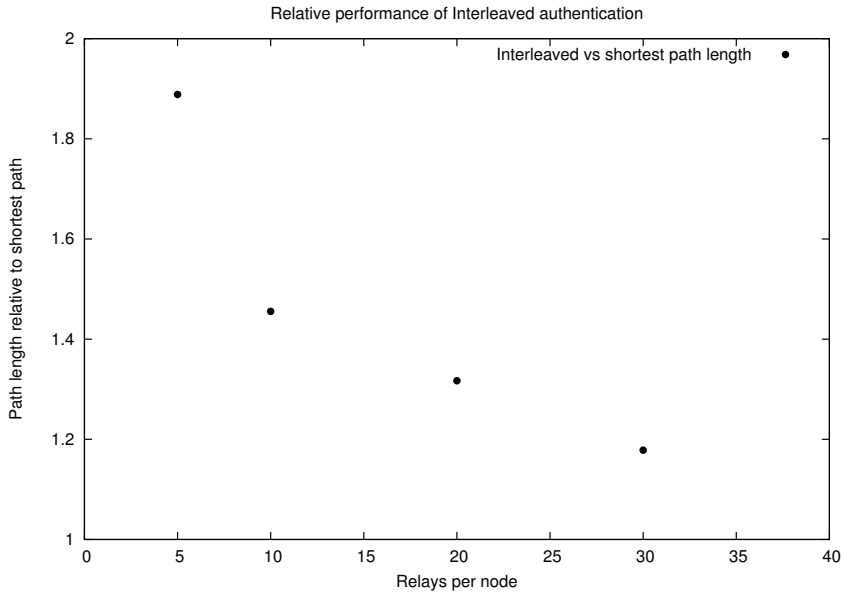


Figure 6.23: Relative length of interleaved paths

6.5.5 Security Evaluation

The security of interleaved authentication is based, like for the preceding schemes, on the creation of multiple authentication paths. Unless all of these paths are compromised, the integrity of a message will be ensured. The correctness condition for a communication path can be summarized as follows:

A node receives the correct message if at least one incoming authentication link carries the correct MAC.

This condition is encapsulated in the decision procedure *is-path-compromised* (Algorithm 7). The idea of the procedure is simple. Nodes are colored and start off being "white". Going along the path, a node is colored "red" if it is either compromised, or all incident authentication links emerge from nodes that have been already colored "red". The path itself is compromised it the target node is colored "red". Trivially, if one of the endpoints of the path is compromised, the path is also compromised. The path is not compromised if there is at least one authentication path from the source to the target that comprises only "white" nodes.

Long-range interleavings shorten the length of authentication paths by reducing the number of involved nodes. This makes a single path less vulnerable to an attack, since now a path depends on the integrity of fewer nodes. The reduction of the authentication path length by increasing the number of shortcuts corresponds to the observation that by adding a few long-range links to a graph that is dominated by local clusters (such as a wireless sensor network), a small-world graph can be constructed [191], where the mean path length is greatly reduced. By introducing long-range authentication links, the authentication graph becomes a small-world graph while the underlying physical communication graph remains a geometric graph.

Interleaved paths make it more difficult for an adversary to attack a certain path, and thus reduces the overall impact of an attack. In order to compromise a path, the adversary has to perform an attack that is focused on specific nodes. In particular, a path is compromised if one of the following conditions is fulfilled:

1. One of the endpoints (i.e. either the source or the target node) is compromised.

2. For a path segment that is not protected by a long-range authentication link, a consecutive group of k nodes on this segment is compromised.

3. If there is a long-range link starting at node A and there is a long-range link incident to node B and B is located closer to the target than A and there is no other node between A and B that is the endpoint of a long-range link, then both A and B have to be compromised, and either A or B have to be part of a compromised group of k consecutive nodes.

The results from simulations shown in Figure 6.24 demonstrate the security performance of long-range interleavings. For all examined attack types, the scheme performs on a high level until approximately half the nodes are

Algorithm 7 *is-path-compromised*(p)

Global values:
$\mathcal{A}$: the set of authentication links (pairs of nodes)
$\mathcal{B}$: the set of compromised nodes

Input:
p: a communication path

Output:
Return *true* if p is compromised, *false* otherwise

```
1:  c[s] := WHITE for all s ∈ p                          ▷ Initialize colors
2:  for i in {1,...,len(p)} do
3:      s := p[i]
4:      if s ∈ B then
5:          c[s] := RED                                  ▷ Compromised nodes are colored RED
6:      else
7:          if ∀j < i.(p[j],s) ∈ A ⇒ c[p[j]] = RED then
8:              c[s] := RED          ▷ RED, if all incident authentications from RED nodes
9:          end if
10:     end if
11: end for
12: return c[p[len(p)]] = RED                            ▷ Target node RED?
```

compromised. With an increasing number of nodes being compromised, security deteriorates at a different rate for each attack type. For a large number of compromised nodes, the protection is highest against a concentrated attack and lowest against a random spread attack. For low numbers, the scheme holds up well against all attack types, even the partitioning attack. Of course, this result confirms the assumption as this scheme was intended to provide good protection against the partitioning attack.

6.6 Comparing Interleaved and Multipath Authentication

6.6.1 Multiple Physical vs. Virtual Paths

There is an immediate similarity between interleaved authentication and multiple path communication. Interleaved authentication creates multiple authentication paths on top of a physical communication path. In multi-path communication, multiple communication paths are explicitly used to transfer a message and associated authentication codes. Thus in both cases, multiple (disjoint) paths are being used for transmitting authentication information. By introducing redundancy, both schemes are able to tolerate a number of compromised nodes.

While establishing disjoint paths in a multi-path scheme requires a trade-

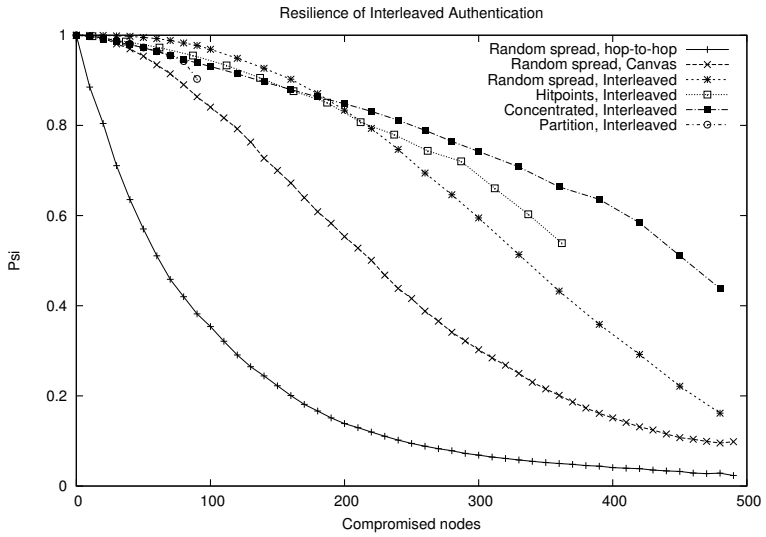


Figure 6.24: Resilience of interleaved authentication under various attacks

off between set-up complexity and path length, multiple "virtual" paths on the same physical path are established in the interleaved authentication scheme that are disjoint by construction. These virtual paths are guaranteed to exist, and they are easily constructed. In addition, only one physical path is involved and the virtual paths are not longer than the physical path.

6.6.2 Combining Authentication Techniques

While we have considered authentication based on multiple physical and virtual paths separately, both techniques can be combined. Each tree path itself may use another layer of authentication, for example *Canvas* authentication. In that way, the advantages of both, multi-path routing and *Canvas* authentication, are combined, cf. Fig. 6.25. Table 6.6 shows that this combination gives a slight advantage over each of the schemes alone.

6.7 Applications

A fundamental idea underlying interleaved authentication is security through collaboration: Two (or more) independent entities provide information about the authenticity of a message, thereby reinforcing the confidence of the receiver that the message is indeed correctly transmitted. We give some examples where this principle is applied.

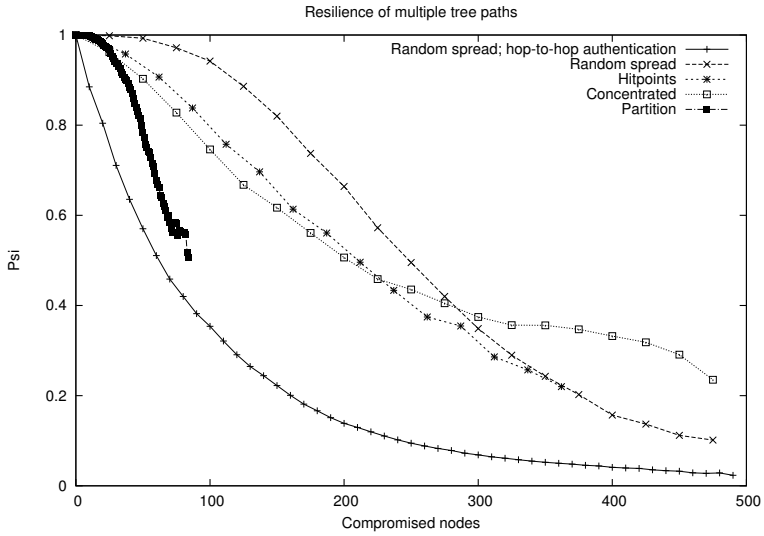


Figure 6.25: Resilience of multiple tree paths with *Canvas* combined

		Attacked nodes / *Change*											
Attack type	**Defense**	**10**		**20**		**50**		**100**		**250**		**400**	
Random spread	**Hop-to-hop**	0.885		0.804		0.570		0.353		0.094		0.040	
	Tree	-		0.853 (25)	*6.1%*	0.653	*14.6%*	0.366	*3.7%*	0.093	*-1.1%*	0.042	*5.0%*
	Canvas	0.998		0.994	*16.5%*	0.953	*45.9%*	0.840	*129.5%*	0.415	*346.2%*	0.151	*259.5%*
	Tree+Canvas	-		0.998 (25)	*0.4%*	0.992	*4.1%*	0.942	*12.1%*	0.495	*19.3%*	0.156	*3.3%*
	Interleaved	0.999		0.999	*0.1%*	0.994	*0.2%*	0.968	*2.8%*	0.746 (240)	*50.7%*	0.358 (390)	*129.5%*
Concentrated	**Hop-to-hop**	0.830		0.744		0.581		0.471		0.283		0.242	
	Tree	-		0.877 (25)	*17.9%*	0.732	*26.0%*	0.571	*21.2%*	0.389	*37.5%*	0.279	*15.3%*
	Canvas	0.942		0.886	*1.0%*	0.737	*0.7%*	0.624	*9.3%*	0.483	*24.2%*	0.386	*38.4%*
	Tree+Canvas	-		0.961 (25)	*8.5%*	0.902	*22.4%*	0.746	*19.6%*	0.435	*-9.9%*	0.332	*-14.0%*
	Interleaved	0.996		0.991	*3.1%*	0.972	*7.8%*	0.930	*24.7%*	0.811 (240)	*86.4%*	0.639 (390)	*92.5%*
Hitpoints	**Hop-to-hop**	0.934 (12)		0.758 (37)		0.642 (62)		0.455 (112)		0.198 (262)		0.131 (337)	
	Tree	0.984 (12)	*5.4%*	0.899 (37)	*18.6%*	0.763 (62)	*18.8%*	0.535 (112)	*17.6%*	0.213 (262)	*7.6%*	0.142 (337)	*8.4%*
	Canvas	0.968 (12)	*-1.6%*	0.880 (37)	*-2.1%*	0.794 (62)	*4.1%*	0.656 (112)	*22.6%*	0.347 (262)	*62.9%*	0.264 (337)	*85.9%*
	Tree+Canvas	0.993 (12)	*2.6%*	0.957 (37)	*8.8%*	0.906 (62)	*14.1%*	0.757 (112)	*15.4%*	0.374 (262)	*7.8%*	0.219 (362)	*-17.0%*
	Interleaved	0.997 (12)	*0.4%*	0.985 (37)	*2.9%*	0.972 (62)	*7.3%*	0.933 (112)	*23.2%*	0.743 (262)	*98.7%*	0.538 (362)	*145.7%*
Partition	**Hop-to-hop**	0.962		0.887		0.628		0.486 (90)		-		-	
	Tree	0.995	*3.4%*	0.977	*10.1%*	0.784	*24.8%*	0.482 (89)	*-0.8%*	-		-	
	Canvas	0.984	*-1.1%*	0.932	*-4.6%*	0.659	*-15.9%*	0.483 (90)	*0.2%*	-		-	
	Tree+Canvas	0.997	*1.3%*	0.980	*5.2%*	0.783	*18.8%*	0.507 (84)	*5.0%*				
	Interleaved	0.998	*0.1%*	0.993	*1.3%*	0.972	*24.1%*	0.903 (90)	*78.1%*	-		-	

Table 6.6: Ψ for various configurations

6.7.1 Coupling Heterogeneous Networks

Suppose two sensor networks are deployed in close proximity to each other by different operators, and both networks share a small overlapping area. Each operator uses his own network for monitoring purposes independently. At one point, the operators decide to coordinate their efforts. They want to allow both networks to exchange messages. However, they don't fully trust each other and want to keep the security relationships between nodes of both networks to a minimum.

The solution is to let the nodes in the overlapping area establish additional security relationships. A fresh set of keys is deployed to nodes in that area. The nodes then engage in a new key agreement phase where they establish shared keys in their k-hop neighbourhoods. The location address space is extended in both networks to include the area covered by the two networks. A node from one network is then able to send a *Canvas*-authenticated message to a node in the other network. The path necessarily passes through the common area, where the message is handed over from one network to the other. Initially, the source of the message may even have a shortcut in the overlapping area and authenticate the message directly to this node. The latter then hands the message over to a node from the other network, which passes it on further towards the destination location, possibly again using its own shortcut relationships. This situation is shown in figure 6.26.

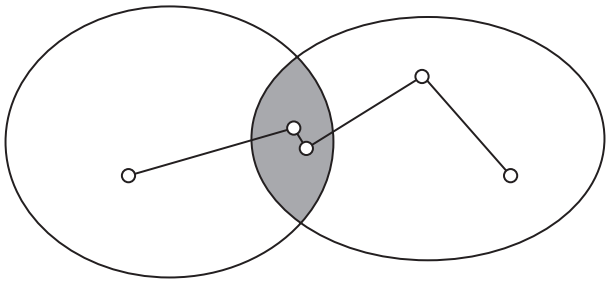


Figure 6.26: A path across two networks

Only nodes in the overlapping area need to be aware of the existence of the second network. To the other nodes, it simply looks like the address space has been extended, and they now are able to send messages to a greater area. In order to send a message, they choose the one shortcut that is closest to the destination, and this shortcut will still be a member of their own network. By geographic routing, the message will pass through the overlapping area. Here, the nodes are aware that a second network exists and that the message needs to be handed over in order to reach its destination. They will choose nodes from the second network as authentication targets. Unless k such authentication

codes are attached to the message, the message is only routed within its original network. As soon as k authentication codes targeted at a node from the second network are available, the message is handed over.

A node in the overlapping area first looks at the message to determine how many authentication codes for the second network are already available. Remember that the k next hops are already determined by previously attached authentication codes. If none of them is targeted at a node from the second network, the node selects an own k-hop neighbour that is a member of the second network and creates an authentication code. From then on, all following hops have to be members of the second network. The reason is that when the message leaves the overlapping area, there have to be k authentication codes from nodes in the second network.

6.7.2 Physical-World Examples

An example in the physical world is the requirement of witnesses for the acceptance, e.g. by some government authority, of certain types of documents. Often, one or more witnesses have to attest that an applicant has indeed signed a document himself. Thereby, it is much harder or even impossible for the signer to later repudiate the signature.

Another example is the delivery of credit cards and their associated PIN codes in two separate letters. Here, the goal isn't authentication but to avoid that an adversary gets access to both the card and the PIN code. This system helps against casual adversaries, even if they are located within the postal system, getting access to credit card letters. It is assumed that the path of both letters is sufficiently diverse such that the group of people seeing both letters is very small. Of course, this is insufficient to counter a dedicated, well-organized adversary. An adversary targeting a single victim, intercepting all letters addressed to the victim, could still obtain both letters.

6.7.3 Internet Applications

A well-known example from the virtual world, where security is achieved through the collaboration of multiple parties, is the PGP web of trust (described in, e.g., [23, 69]). The authenticity of a hitherto unknown public key is established through certificates that have been issued by others. If the issuers are unknown as well, certificates for them have to be obtained first. This continues until some known, trusted parties are encountered. Such a chain must not exceed a certain length (which the user can specify). The more of such (disjoint) chains exist, the higher is the confidence in the authenticity of a public key.

The use of independent paths to convey authentication information is common on the Internet. As an example, consider user accounts for web sites. Operators of such sites are usually interested in giving user accounts only to human beings but not to "robots". It has become a general practice to generate and display blurred pictures containing a code word on the account registration page. To complete the registration, it is required to enter that code word before the account is granted. Since this code word can only be understood by a human being, automatically operating programs cannot generate new user accounts, which inhibits denial-of-service attacks.

Another common example, in the same domain, is the acknowledgement of a newly generated user account through a second communication channel, usually e-mail. After a user has signed up for an account, he retrieves an e-mail message that he has to acknowledge before the account is activated. Thereby, the validity of the e-mail address is confirmed, which also constitutes a form of authentication.

6.7.4 E-Mail Origin Authentication

The fight against illicit e-mail, commonly called *spam*, has attracted some attention recently. One major solution attempt is the introduction of the ability to verify the purported identity of the sender of an e-mail mesage, i.e. message authentication. The Internet standard protocol for transmitting e-mail, Simple Mail Transfer Protocol (SMTP) [94], is not prepared to do so by default. Spam messages often originate at spoofed origins (for increasing their credibility and for distributing the load of creating and sending them). A large fraction of these messages could be filtered out reliably if it could be detected that the purported sender did in fact not send the message.

A number of proposals exist for different mechanisms to authenticate e-mail messages. The most prominent ones are DomainKeys [85, 73] and SPF/Sender ID [197]. The main idea of DomainKeys is to digitally sign a message (and some invariant header information) before it is being sent. The digital signature is created by the sending SMTP server (not the mail client program of the user). For verification, the public key is obtained from DNS. The outgoing SMTP server must ensure that only authorized users can send messages, otherwise a sender of spam messages could exploit such an open server.

SPF relies on authenticating the IP address of the sender's mail transfer agent (MTA, i.e. the outgoing SMTP server). The IP address is obtained through the TCP/IP connection between the sending MTA and the receiving MTA. DNS must be extended to contain a list of allowed IP addresses for every

domain. The receiver looks up the DNS record corresponding to the sender's e-mail address. If the IP address from which the message was received is not contained in that record, the message is rejected.

Van Oorschot [178] has proposed a mechanism for sender authentication that does require neither extension of the DNS nor public-key cryptography. The idea is to convey authentication information about e-mail messages through a second, access-restricted channel. For each message being sent, a hash value is computed and stored in some publicly readable location to which only the legitimate owner of the e-mail address has write access. The receiver of a message looks up the list of hash values and verifies whether the one matching the received message is present. Only if the hash value is found, the message is accepted.

This approach matches the first two steps in a *Canvas* protocol run. In the *Canvas* authentication scheme, the originator A sends a message through two channels provided by mutually shared secret keys. One message goes to the second-hop neighbour C, which corresponds to the receiver of an e-mail message. The other message goes to a one-hop neighbour B that is adjacent to both the originator and the receiver. Node B corresponds to the publicly readable site that conveys an authentication code for the message. This code is made available to C through the private channel between B and C.

A difference between *Canvas* and van Oorschot's proposal is that in the latter case, B is merely a passive storage location to which only A can write but everybody can read from. In *Canvas*, B is an active player and creates the authentication code for a message by itself. The fact that B includes the sender's identity of a message in that code corresponds to the exclusive write access in van Oorschot's proposal.

One problem of van Oorschot's proposal, which he also recognizes, is that the public site where message fingerprints are stored must be familiar to the receivers of messages. He assumes that this relationship may be established through out-of-band mechanisms, or through web sites that are uniquely associated with e-mail addresses. Out-of-band mechanisms raise problems for short-lived communication relationships where additional effort is usually undesirable. Establishing a site that provides fingerprints under URLs such as

```
http://www.e-mail-fingerprints.org/address=john@company.com
```

may be feasible, but raises questions regarding scalability, funding, and security. Such a central site would have effective control over what messages would be considered as spam, and which not. This would be obviously undesirable in an open communication network. The best solution might be to extend the

DNS and store an additional record that provides, for each address domain, the URL of a site that contains the message fingerprints. Thereby, domain owners are free to administer their own servers.

6.8 Related Work

To our knowledge, interleaved authentication schemes have been independently conceived independently from our work [180, 181, 182] by several authors [71, 206].

The first record is by Goodrich [71], which has been extended later [72]. In his scheme, for each node x, there is a key $k(x)$ that is shared among all nodes in the neighbourhood of x, excluding x itself. A node adjacent to x uses $k(x)$ to add an authentication code to a message before it sends it to x. When x passes the message (including the authentication code) on to another one of its neighbours, the receiver can verify that x has not modified the message. The difference to *Canvas* is the use of a single key shared by all neighbours to protect against the possible compromise of a node (x). In this approach, x can forward a message according to local requirements. It is not required that the sender of a message knows the path further down of x. *Canvas* requires this knowledge and thus implies additional communication. Goodrich applies the technique to securing the set-up of routing tables.

The second independent record is by Zhu et al. [206], who proposed interleaved authentication for integrity checking of messages that are passed along a path from sensor nodes towards a base station. Similar to our work, they extended the scheme for interleavings of more than two hops, which allows for protection against colluding nodes on the path. Their main application of the technique is filtering compromised messages before they actually reach a sink.

6.9 Summary

In this chapter we have presented a family of protocols for communication integrity protection. The protocols are especially suitable for the use in wireless sensor networks as they do not rely on extensive end-to-end security relationships. Instead, local security relationships between k-hop neighbours (with a small k) and interleavings of authentication paths provide a security level that approximates that of end-to-end security schemes. The security level can be further improved by introducing a small number of long-range security relationships. We have shown the security performance of these protocols through

simulations and by comparison with conventional hop-to-hop and end-to-end schemes.

Chapter 7

Conclusion

This work proposed a security infrastructure for protecting the communications in wireless sensor networks. We have identified the node capture attack as an important threat to wireless sensor networks. In order to protect a WSN against this threat, a range of mechanisms are required, covering all layers of the system. This includes the hardware design of the sensor platform, all protocol layers of the wireless communication, the platform's runtime environment, and the application software.

In this work, we have focused on providing the foundation for secure communications in a WSN. We started from the assumption that a node capture attack cannot be fully prevented and thus an attacker would be able to compromise a certain number of nodes. The goal has been set to provide a certain level of security for the communication in the network even under such adverse conditions.

The following contributions have been made:

- Based on existing approaches to key pre-distribution, we proposed a novel key agreement scheme that uses hash chains for strengthening the negotiated keys.

- For protecting the integrity of messages in a WSN, we proposed an interleaved authentication scheme that uses locally shared keys to create multiple, virtual authentication paths between communication endpoints and thus approximates the security guarantees of end-to-end authentication.

- Building on interleaved authentication, we proposed a protocol extension that leverages a small number of long-range security relationships to counteract sophisticated, structured attacks.

- Complementing the virtual authentication path approach, we proposed a novel way of constructing multiple physical communication paths that

provide physical communication redundancy, which can provide similar security guarantees.

7.1 Secure Communication in Wireless Sensor Networks

7.1.1 Key Establishment

Shared secret keys are a prerequisite for cryptographically secured communication between nodes in a network. In wireless sensor networks, key establishment schemes should be used that minimize the overhead for setting up shared keys. Therefore, key pre-distribution has been proposed as a mechanism as it achieves a two-fold goal. First, it requires only little computational effort during the actual key agreement phase, which takes place after node deployment when nodes have to rely on their individual, limited power supply. Second, it provides node authentication on the group level, i.e. only nodes that belong to the legitimate set of deployed nodes are able to successfully engage in a key agreement.

Random key pre-distribution is the most generally applicable pre-distribution scheme as no assumptions are made on the distribution of nodes after their deployment. If information about their distribution is available, more efficient schemes are possible. Key pre-distribution schemes rely on a large pool of keys, from which a subset of keys is assigned to each node. Pairwise key establishment is comprised of first determining the keys that both parties have in common, and second deriving a value from these keys, which then acts as the shared key.

We have considered another approach to key agreement that is based on sets of hash chains. Exploiting the one-way property of hash functions, keys can be derived from hash chain values that are resistant to low-strength attackers.

The real benefit of hash chains is realized when they are used to strengthen the keys that have been established using a random pre-distribution scheme. By combining both techniques, the resilience of keys is significantly enhanced.

7.1.2 Multiple Path Communication

Communication over a single path is generally vulnerable against failures and security threats. Usually, these vulnerabilities are mitigated by high-level protocols that, in case of link or router failures, verify the reliable transmission of messages and initiate retransmissions if necessary. Additional protocols can

ensure the secure transmission of messages by authenticating routers, or ensuring link security, for example.

An alternative communication scheme that can potentially counter these problems is multi-path communication. Here, the same message is sent over multiple paths that only share the same end-points, but use no common communication links or routers (or neither of them). Here, a threshold scheme can ensure that an attacker cannot reconstruct or modify without detection a message. Generally, constructing short disjoint paths is a complex task with a high computational overhead.

Protocols that work in conventional networks with high-powered nodes do not necessarily work well in highly resource-constrained environments like wireless sensor networks. Therefore, alternative approaches have been devised.

We have proposed a scheme for constructing multiple disjoint (node) disjoint paths that is suited for wireless sensor networks. The basic concept are routing trees that can be set up very easily by selecting a root for each tree and a single wave of broadcast messages for tree construction. A message is then routed along the links that are contained in a tree. This increases the path length by a manageable amount, but also provides for a high degree of disjointness for pairs of tree paths. This scheme therefore provides a practical trade-off between additional complexity and security.

7.1.3 Interleaved Authentication

One of the main goals of a secure communication protocol is to provide a means for remote nodes to exchange messages that are protected against manipulations that may happen while the messages are in transit. In a wireless sensor network, nodes act as routers, relaying messages on behalf of other nodes. Secure communication between two nodes thus depends on the collaboration of all intermediate nodes. Especially, intermediate nodes are expected to relay messages with their contents unaltered. An integrity-protecting communication scheme ensures that manipulations would be detectable.

In a conventional approach, an end-to-end message authentication protocol is being used, which is based on public key cryptography or pairwise shared keys. Public key cryptography introduces a relatively large overhead regarding computational effort for creating and verifying signatures, and requires the transmission of authentication data of significant size. On these grounds, public key cryptography is often dismissed for use in resource-constrained environments. An alternative are fully pairwise shared keys, which allow the use of more efficient symmetric key cryptography. However, the storage require-

ments are overwhelming and such a system would be very much constrained in its flexibility to accomodate extensions to the network.

Our approach to secure communication relies on the availability of shared keys only between nodes that are in close proximity to each other. This limits the overhead for key storage considerably. The *Canvas* scheme proposed in this work requires that each node shares a key with each of its one- to k-hop neighbours ($k \geq 2$). In a typical deployment setting with $k = 2$, this would require each node to store about 10 to 20 keys, which sensor nodes are well capable of. A message that is about to be transmitted to a remote node is authenticated by the source using at least two keys, which are shared with the following nodes on the communication path. This requires only minor adjustments on the routing layer, namely a look-ahead of k nodes on the routing path. The same authentication pattern is repeatedly applied by all nodes on the path.

The protection provided by such a scheme is able to render single compromised nodes ineffective. A message is authenticated by at least two authentication codes, but only one of them can be manipulated by the compromised node, thus any change in the message's content would be discovered by the next node on the path. At that point, the network would become aware of the attack – something an attacker wants to avoid as this degrades the trust in the network, which also degrades the value of the attack.

This interleaving of message authentication codes corresponds to creating multiple independent authentication paths, i.e. paths on which authentication information is passed. With $k = 2$, *Canvas* creates two such paths. Thus it is able to accomodate compromised nodes on both paths that are not adjacent to each other on the communication path. *Canvas* is similar to having two physically disjoint communication paths in the regard that a single compromised path cannot break the authentication. Additionally, *Canvas* gives the advantage that both paths are interlinked and breaking both of them requires a certain configuration of compromised nodes.

The *Canvas* scheme has a limited reach in that only isolated compromised nodes can be countered. As soon as the attacker manages to subvert clusters of nodes, the scheme becomes partially ineffective. Any message that passes through a pair of compromised nodes would be subject to manipulation. In order to counter certain attack patterns, it is therefore necessary to introduce long-distance authentication relationships.

7.1.4 Shortcut Authentication

The limited reach of *Canvas* is overcome by the introduction of authentication shortcuts. Such a shortcut is a security relationship, implemented by a shared secret key, between remote nodes. One node acts as a "trusted relay" for another node. A source sending a message to a receiver that is far away uses such a shortcut to strengthen the authentication of the message. The source adds an authentication code addressed to the relay and sends the message to its destination through a relay close to the destination. After authenticating the message, the relay forwards the message on to the target. Thereby, long distances can be spanned by a single authentication code, mitigating attacks that rely on clusters of compromised nodes.

A shortcut effectively shortens an authentication path. Thereby, shortcuts minimize the "attack surface" of the system. Simply stated, there are fewer nodes involved in authenticating a message, thus the attacker has fewer opportunities to attack a message. In general, this makes a WSN more robust against node capture attacks. In particular, shortcuts are the only effective means against partitioning attacks.

7.2 Future Work

Wireless sensor networks are an emerging technology that has a broad potential use in industrial, commercial, and personal applications and for public safety and security. The miniaturization of microprocessors and the further development of wireless communication and power generation for tiny devices give rise to the expectation that wireless sensor networks will become an integral part of structures of any kind, either natural or artificial. Thus, decisions will rely increasingly on the data provided by wireless sensor networks and it is important to ensure the secure and reliable operation of these networks. In this work, we have highlighted need for integrity protection and devised methods for doing so. It will be important to come up with a comprehensive framework for the security of wireless sensor networks that includes all aspects important to security, ranging from access control, privacy protection, integrity, timeliness, and accuracy. The challenge will be to find the right trade-off between the complexity that is necessary to provide the security guarantees, and the power-efficiency that is necessary for implementation on tiny, highly constrained devices.

Bibliography

[1] Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface . http://www.3gpp.org/, 1999.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[3] Ross Anderson. *Security Engineering*. John Wiley & Sons, 2001.

[4] Ross Anderson, Mike Bond, and Steven J. Murdoch. Chip and Spin. Technical report, Computer Laboratory, University of Cambridge, 2005. http://www.cl.cam.ac.uk/users/sjm217/papers/cl05chipandspin.pdf.

[5] Ross Anderson, Haowen Chan, and Adrian Perrig. Key Infection: Smart Trust for Smart Dust. In *12th IEEE International Conference on Network Protocols (ICNP'04)*, pages 206–215. IEEE, 2004.

[6] Ross Anderson and Markus Kuhn. Tamper Resistance – a Cautionary Note. In *The Second USENIX Workshop on Electronic Commerce*, pages 1–11. USENIX Association, 1996.

[7] Brad Arkin, Scott Stender, and Gary McGraw. Software Penetration Testing. *IEEE Security & Privacy Magazine*, 3(1):84–87, January 2005.

[8] Sasikanth Avancha, Jeffrey Undercover, Anupam Joshi, and John Pinkston. Secure sensor networks for perimeter protection. *Elsevier Computer Networks*, 43:421–435, 2003.

[9] Baruch Awerbuch. Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and related problems. In *Proceedings of the 19th Annual ACM Conference on Theory of Computing*, pages 230–240. ACM Press, 1987.

[10] Amitabha Bagchi, Amitabh Chaudhary, Michael T. Goodrich, and Shouhuai Xu. Constructing Disjoint Paths for Secure Communication. In

Distributed Computing, volume 2848, pages 181–195. Springer-Verlag, 2003.

[11] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Symposium on Network and Distributed Systems Security (NDSS '02)*. Internet Society, February 2002.

[12] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. IETF RFC 2201, September 1997. `http://www.ietf.org/rfc/rfc2201.txt`.

[13] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure Pebblenets. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 156–163. ACM Press, 2001.

[14] Matthias Bauer. Spass mit Hashes. `http://www1.informatik.uni-erlangen.de/tree/Persons/bauer/new/hash.ps`, 2002.

[15] Alexander Becher, Zinaida Benenson, and Maximilian Dornseif. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In *3rd Int. Conf. on Security in Pervasive Computing (SPC)*, volume 3934 of *LNCS*. Springer-Verlag, April 2006.

[16] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPTO '96: 16th Annual International Cryptology Conference*, volume 1109 of *LNCS*. Springer-Verlag, 1996.

[17] Zinaida Benenson, Felix C. Gärtner, and Dogan Kesdogan. An Algorithmic Framework for Robust Access Control in Wireless Sensor Networks. In *2nd European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[18] Christian Bettstetter. On the Minimum Node Degree and Connectivity of a Wireless Multihop Network. In *MOBIHOC'02*. ACM, 2002.

[19] Christian Bettstetter and Johannes Zangl. How to Achieve a Connected Ad Hoc Network with Homogeneous Range Assignment: An Analytical Study with Consideration of Border Effects. In *4th International Workshop on Mobile and Wireless Communication Networks*. IEEE, 2002.

[20] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund, and L. Thiele. Prototyping Wireless Sensor Network Applications with BTnodes. In

1st European Workshop on Wireless Sensor Networks, volume 2920 of *LNCS*. Springer-Verlag, 2004.

[21] Matt Bishop. *Computer Security – Art and Science*. Addison-Wesley, 2002.

[22] Erik-Oliver Blaß and Martina Zitterbart. Towards Acceptable Public-Key Encryption in Sensor Networks. In *2nd International Workshop on Ubiquitous Computing*, pages 88–93. INSTICC Press, 2005.

[23] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *IEEE Symposium on Security and Privacy*. IEEE, 1996.

[24] Rolf Blom. An Optimal Class of Symmetric Key Generation Systems. In *Advances in Cryptology: Proceedings of EUROCRYPT 84 - A Workshop on the Theory and Application of Cryptographic Techniques*, volume 209 of *LNCS*. Springer-Verlag, 1985.

[25] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. In *Advances in Cryptology - CRYPTO '92: 12th Annual International Cryptology Conference*, volume 740 of *LNCS*. Springer-Verlag, 1993.

[26] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Social, Economic, and Ethical Implications of Ambient Intelligence and Ubiquitous Computing. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*, pages 5–29. Springer-Verlag, 2005.

[27] Béla Bollobás. *Random Graphs*. Cambridge University Press, 2001.

[28] Mike Bond and Ross Anderson. API-Level Attacks on Embedded Systems. *Computer*, 34(10), 2001.

[29] Detlef Borchers. Nun doch ein Vorzeigeprojekt. *c't*, 2, 2005.

[30] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*. ACM Press, 2001.

[31] David Braginsky and Deborah Estrin. Rumor Routing Algorithm for Sensor Networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31. ACM Press, 2002.

[32] Sonja Buchegger and Jean-Yves Le Boudec. Cooperative Routing in Mobile Ad-hoc Networks: Current Efforts Against Malice and Selfishness. In *GI Jahrestagung*, volume 19 of *Lecture Notes in Informatics*, pages 513–517. Gesellschaft für Informatik, 2002.

[33] Mike Burmester and Tri Van Le. Secure Multipath Communication in Mobile Ad hoc Networks. In *International Conference on Information Technology: Coding and Computing (ITCC 2004)*, volume 2, pages 405–409, Las Vegas, April 2004. IEEE.

[34] Levente Buttyán and Jean-Pierre Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 8(5):579–592, October 2003.

[35] Antonio Carzaniga and Alexander L. Wolf. Content-Based Networking: A New Communication Infrastructure. In *Developing an Infrastructure for Mobile and Wireless Systems: NSFWorkshop IMWS 2001*, volume 2538 of *LNCS*. Springer-Verlag, 2001.

[36] Haowen Chan, Virgil D. Gligor, Adrian Perrig, and Gautam Muralidharan. On the Distribution and Revocation of Cryptographic Keys in Sensor Networks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):233–247, 2005.

[37] Haowen Chan and Adrian Perrig. Security and Privacy in Sensor Networks. *IEEE Computer*, October 2003.

[38] Haowen Chan and Adrian Perrig. PIKE: Peer Intermediaries for Key Estasblishment in Sensor Networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2005.

[39] Haowen Chan, Adrian Perrig, and Dawn Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–213. IEEE, May 2003.

[40] Xiao Chen and Xingde Jia. Package Routing Algorithms in Mobile Ad-Hoc Wireless Networks. In *International Conference on Parallel Processing Workshops*, pages 485–490. IEEE, 2001.

[41] Rohan Chitradurga and Ahmed Helmy. Analysis of Wired Short Cuts in Wireless Sensor Networks. In *IEEE/ACS International Conference on Pervasive Services (ICPS)*, pages 167–176. IEEE, 2004.

[42] Howard Chivers and John A. Clark. Smart dust, friend or foe? – Replacing identity with configuration trust. *Computer Networks*, 46(5), 2004.

[43] Martin Connolly and Fergus O'Reilly. Sensor Networks and the Food Industry. In *REALWSN*, 2005. `http://www.sics.se/realwsn05/proceedings.html`.

[44] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[45] Sadie Creese, Michael Goldsmith, Bill Roscoe, and Irfan Zakiuddin. The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model. In *Workshop on Formal Aspects in Security and Trust*, Pisa, Italy, 2003. `http://www.iit.cnr.it/FAST2003/`.

[46] Sadie Creese, Michael Goldsmith, Bill Roscoe, and Irfan Zakiuddin. Authentication for Pervasive Computing. In *Security in Pervasive Computing*, number 2802 in LNCS, pages 116–129. Springer-Verlag, 2004.

[47] David E. Culler and Hans Mulder. Smart Sensors to Network the World. *Scientific American*, June 2004.

[48] Swades De, Chunming Qiao, and Hongyi Wu. Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks. *Computer Networks*, 43(4):481–497, 2003.

[49] Jing Deng, Richard Han, and Shivakant Mishra. INSENS: Intrustion-Tolerant Routing in Wireless Sensor Networks. Technical Report CU-CS-939-02, University of Colorado, Department of Computer Science, 2002.

[50] Dorothy E. Denning. An Intrusion-Detection Model. In *IEEE Symposium on Security and Privacy*. IEEE, 1986.

[51] Christophe Devine. SHA-1 Source Code. `http://www.cr0.net:8040/code/crypto/sha1/`.

[52] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal*, 16(2–4):97–166, 2001.

[53] T. Dierks and C. Allen. The TLS Protocol. IETF RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt.

[54] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[55] Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in P2P Anonymity Systems. In *1st Workshop on Economics of P2P Systems*. University of California, Berkeley, 2003. http://www.sims.berkeley.edu/p2pecon/.

[56] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *Journal of the ACM*, 40(1), January 1993.

[57] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[58] John R. Douceur. The Sybil Attack. In *Peer-to-Peer Systems, 1st International Workshop (IPTPS)*, number 2429 in LNCS, pages 251–260. Springer-Verlag, 2002.

[59] Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, February 2004.

[60] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03)*, pages 42–51. ACM Press, 2003.

[61] 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 Transactions on Information and System Security*, 8(2), May 2005.

[62] Adam Dunkels, Thiemo Voigt, Niclas Bergman, and Mats Jönsson. The Design and Implementation of an IP-based Sensor Network for Intrusion Monitoring. In *Swedish National Computer Networking Workshop*, Karlstad, Sweden, November 2004.

[63] Joan G. Dyer, Mark Lindemann, Ronald Perez, Reiner Sailer, Leendert van Doorn, Sean W. Smith, and Steve Weingart. Building the IBM 4758 Secure Coprocessor. *Computer*, 34(10):57–66, October 2001.

[64] L. Eschenauer and V. D. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*. ACM Press, 2002.

[65] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270. ACM Press, 1999.

[66] L. Fernando Friedrich, John Stankovic, Marty Humphrey, Michael Marley, and John Haskins. A Survey of Configurable, Component-Based Operating Systems for Embedded Applications. *IEEE Micro*, 21(3):54–68, 2001.

[67] Bundesamt für Sicherheit in der Informationstechnik (BSI). Digitale Sicherheitsmerkmale im elektronischen Reisepass. `http://www.bsi.de/fachthem/epass/Sicherheitsmerkmale.pdf`, June 2005.

[68] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communications Review*, 1(2), 1997.

[69] Simson Garfinkel. *PGP Pretty Good Privacy*. O'Reilly, 1994.

[70] Brian Gladman. AES and Combined Encryption/Authentication Modes. `http://fp.gladman.plus.com/AES/`, 2005.

[71] Michael T. Goodrich. Efficient and secure network routing algorithms. `http://www.cs.jhu.edu/~goodrich/cgc/pubs/routing.pdf`, 2001.

[72] Michael T. Goodrich. Leap-Frog Packet Linking and Diverse Key Distributions for Improved Integrity in Network Broadcasts. In *2005 IEEE Symposium on Security and Privacy*, pages 196–207. IEEE, 2005.

[73] IETF Working Group. Domain Keys Identified Mail (DKIM). `http://www.ietf.org/html.charters/dkim-charter.html`, January 2006.

[74] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*, volume 3156 of *LNCS*. Springer-Verlag, 2004.

[75] Bert Gyselinckx, Chris Van Hoof, Julien Ryckaert, Refet Firat Yazicioglu, Paolo Fiorini, and Vladimir Leonov. Human++: Autonomous Wireless Sensors for Body Area Networks. In *Proceedings of the IEEE 2005 Custom Integrated Circuits Conference*. IEEE, 2005.

[76] Tom R. Halfhill. Embedded Microprocessors. *Computerworld*, August 2000. `http://www.computerworld.com/news/2000/story/0,11280,49033,00.html`.

[77] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *SecureComm*. IEEE, 2005.

[78] Helena Handschuh and Pascal Paillier. Smart Card Crypto-Coprocessors for Public-Key Cryptography. In *Smart Card Research and Applications (CARDIS '98)*, volume 1820 of *LNCS*. Springer-Verlag, 2000.

[79] Ahmed Helmy. Small Worlds in Wireless Networks. *IEEE Communication Letters*, October 2003.

[80] John Hering, James Burgess, Kevin Mahaffeya, Mike Outmesguine, and Martin Herfurt. Long Distance Snarf. `http://trifinite.org/trifinite_stuff_lds.html`, August 2004.

[81] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280, April 2002. `http://www.ietf.org/rfc/rfc3280.txt`.

[82] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *MobiCom '02*. ACM Press, 2002.

[83] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, San Francisco, CA, april 2003. IEEE. leash = Leine;.

[84] Andrew Huang. Keeping Secrets in Hardware: The Microsoft Xbox™ Case Study. In *Cryptographic Hardware and Embedded Systems – CHES 2002: 4th International Workshop*, volume 2523 of *LNCS*. Springer, 2003.

[85] Yahoo! Inc. DomainKeys: Proving and Protecting Email Sender Identity. http://antispam.yahoo.com/domainkeys.

[86] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of 6th Ann. Int. Conf. on Mobile Computing and Networking (MobiCom)*, pages 56–67. ACM, 2000.

[87] David B. Johnson, David A. Maltz, and Josh Broch. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353 of *The International Series in Engineering and Computer Science*, chapter 5. Springer, 1996.

[88] Don Johnson and Alfred Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical Report Technical Report CORR 99-34, University of Waterloo, 1999. http://www.cacr.math.uwaterloo.ca.

[89] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM Press, 2004.

[90] Chris Karlof and David Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Elsevier Ad Hoc Networks*, 1(2–3):295–315, September 2003.

[91] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 243–254. ACM Press, 2000.

[92] Nicky Kern, Bernt Schiele, and Albrecht Schmidt. Multi-Sensor Activity Context Detection for Wearable Computing. In *Ambient Intelligence*, number 2875 in LNCS, pages 220–232. Springer-Verlag, November 2003.

[93] Jon Kleinberg and Eva Tardos. Approximations for the Disjoint Paths Problem in High-Diameter Planar Networks. *Journal of Computer and System Sciences*, 56(1):61–73, August 1998.

[94] J. Klensin. Simple Mail Transfer Protocol. IETF, RFC 2821, http://www.ietf.org/rfc/rfc2821.txt, April 2001.

[95] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan, and Srivaths Ravi. Security as a New Dimension in Embedded Systems Design. In *41st Conference on Design Automation*, pages 753–760. IEEE, 2004.

[96] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology (Smartcard '99)*, pages 9–20. USENIX Association, 1999.

[97] Philip Koopman. Embedded System Security. *Computer*, July 2004.

[98] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. `http://arxiv.org/abs/cs.DC/0502003`, February 2005.

[99] D. Richard Kuhn, Thomas J. Walsh, and Steffen Fries. Security Considerations for Voice over IP Systems – Recommendations of the National Institute of Standards and Technology. `http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf`, January 2005. NIST Special Publication 800-58.

[100] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 63–72. ACM Press, 2003.

[101] Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking Ciphers with COPACOBANA – A Cost-Optimized Parallel Code Breaker. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *LNCS*, pages 101–118. Springer-Verlag, 2006.

[102] Kaspersky Lab. PDAs under attack. `http://www.kaspersky.com/news?id=151142122`, August 2004. Appearance of the *Backdoor.WinCE.Brador.a* Trojan horse program.

[103] RSA Laboratories. PKCS #5 v2.0: Passoword-Based Cryptography Standard. `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf`, March 1999.

[104] RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard. `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`, June 2002.

[105] Kristof Van Laerhoven, Benny P.L. Lo, Jason W.P. Ng, Surapa Thiemjarus, Rachel King, Simon Kwan, Hans-Werner Gellersen, Morris Sloman, Oliver Wells, Phil Needham, Nick Peters, Ara Darzi, Chris Toumazou, and Guang-Zhong Yang. Medical Healthcare Monitoring with Wearable and Implantable Sensors. In *UbiHealth 2004: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, 2004. `http://www.pervasivehealthcare.com/ubicomp2004/papers/`.

[106] Gregory Lamm, Gerlando Falauto, Jorge Estrada, Jad Gadiyaram, and Daniel Cockerham. Bluetooth Wireless Security Features. In *Workshop on Information Assurance and Security*. IEEE, June 2001.

[107] Leslie Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, 1981.

[108] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[109] David Larochelle and David Evans. Statically Detecting Likely Buffer Overflow Vulnerabilities. In *Proceedings of the 10th USENIX Security Symposium*, 2001.

[110] Jooyoung Lee and Douglas R. Stinson. Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In *SAC 2004*, volume 3357 of *LNCS*, pages 294–307. Springer-Verlag, 2004.

[111] Sung-Ju Lee and Mario Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In *IEEE International Conference on Communications, ICC*, volume 10, pages 3201 – 3205. IEEE, 2001.

[112] Tom Leighton and Silvio Micali. Secret-Key Agreement without Public-Key Crpytography (Extended Abstract). In *Advances in Cryptology – CRYPTO '93*, number 773, pages 456–479. Springer-Verlag, 1993.

[113] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. "Are You with Me?" – Using Accelerometers to Determine If Two Devices Are Carried by the Same Person. In *Pervasive Computing, 2nd International Conference*, number 3001 in LNCS, pages 33–50. Springer-Verlag, 2004.

[114] Philip Levis and David Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *ASPLOS-X: Proceedings of the 10th International Confer-*

ence on Architectural Support for Programming Languages and Operating Systems, pages 85–95. ACM Press, 2002.

[115] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, 2003.

[116] An Liu and Peng Ning. TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.1). `http://discovery.csc.ncsu.edu/software/TinyECC/`, September 2005.

[117] Donggnag Liu and Peng Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03)*, pages 52–61. ACM Press, 2003.

[118] Paul Lukowicz, Oliver Amft, David Bannach, and Gerhard Tröster. Heterogeneous, Distributed On Body Computing in the WearIt@Work Project. In *Proceedings of the International IEEE Conference in Mechatronics and Robotics*. IEEE, 2004.

[119] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[120] C. Mallanda, A. Suri, V. Kunchakarra, S. S. Iyengar, R. Kannan, A. Durresi, and S. Sastry. Simulating Wireless Sensor Networks with OMNeT++. Submitted for publication, 2005.

[121] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2005.

[122] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[123] Pietro Michiardi and Refik Molva. Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks. In *European Wireless Conference*, 2002. `http://www.ing.unipi.it/ew2002/`.

[124] Steven E. Minzer. Broadband ISDN and Asynchronous Transfer Mode (ATM). *IEEE Communications Magazine*, September 1989.

[125] Mirriam-Webster Online Dictionary. `http://www.m-w.com`, 2005.

[126] Anne Marie Mohan. RFID key to FDA's anti-counterfeiting strategy. *Packaging Digest*, June 2004. http://www.packagingdigest.com/articles/200406/54.php.

[127] Patrick Moor and Mario Strasser. Key Management for Sensor Networks. Student project report, ETH Zürich, Institute for Pervasive Computing, 2005.

[128] J. Moy. OSPF Version 2. IETF RFC 2328, April 1998. http://www.ietf.org/rfc/rfc2328.txt.

[129] Randall K. Nichols and Panos C. Lekkas. *Wireless Security. Models, Threats, and Solutions*. McGraw-Hill TELECOM, 2001.

[130] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[131] Peter Oehlert. Violating Assumptions with Fuzzing. *IEEE Security & Privacy Magazine*, 3(2):58–62, March 2005.

[132] National Institute of Standards and Technology. Specification for the Advanced Encryption Standards (AES). FIPS 197, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001.

[133] National Institute of Standards and Technology. The Keyed-Hash Message Authentication Code (HMAC). FIPS 198, http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf, March 2002.

[134] Andy Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly Media, Inc., 2001.

[135] Christof Paar. Embedded Security in Automobilanwendungen. *Elektronik Automotive*, January 2004.

[136] Jeongyeup Paek, Krishna Chintalapudi, Ramesh Govindan, John Caffrey, and Sami Masri. A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience. In *2nd IEEE Workshop on Embedded Networked Sensors*. IEEE, 2005.

[137] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *IEEE Symposium on Privacy and Security*, pages 49–63. IEEE, 2005.

[138] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *Journal of the Association for Computing Machinery*, 27(2):228–234, April 1980.

[139] Wuxu Peng, Xin Zhand, and Kia Makki. Locality Caching Mult-Root Multi-Generation Routing Algorithm in Mobile Ad Hoc Networks. In *12th International Conference on Computer Communication and Networks, ICCCN*, pages 229–234. IEEE, 2003.

[140] Mathew Penrose. *Random Geometric Graphs*, volume 5 of *Oxford Studies in Probability*. Oxford University Press, 2003.

[141] Elio Perez. 802.11i (How we got here and where are we headed). The SANS Technology Institute, `http://www.sans.org/rr/whitepapers/wireless/1467.php`, August 2004.

[142] Adrian Perrig, Rober Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(8):521–534, 2002.

[143] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Random Key-Assignment for Secure Wireless Sensor Networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*. ACM, 2003.

[144] K. Pister. Smart dust – autonomous sensing and communication in a cubic millimeter. `http://robotics.eecs.berkeley.edu/~pister/SmartDust/`, 2001.

[145] Niels Provos. A Virtual Honeypot Framework. In *Proceedings of the 13th USENIX Security Symposium*, pages 1–14, 2004.

[146] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure Information Aggregation in Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM Press, 2003.

[147] Michael O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[148] Mahalingam Ramkumar and Nasir Memon. An Efficient Key Predistribution Scheme for Ad Hoc Network Security. *IEEE Journal on Selected Areas in Communications*, 23(3), March 2005.

[149] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic Routing without Location Information. In *9th Ann. Int. Conf. on Mobile Computing and Networking*, pages 96–108. ACM Press, 2003.

[150] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 161–172. ACM Press, 2001.

[151] Srivaths Ravi, Anand Raghunathan, and Srimat Chakradhar. Tamper Resistance Mechanisms for Secure Embedded Systems. In *17th Int. Conf. on VLSI Design*, pages 605–611. IEEE, 2004.

[152] Michael K. Reiter and Stuart G. Stubblebine. Resilient Authentication Using Path Independence. *IEEE Transactions on Computers*, 47(12):1351–1362, 1998.

[153] Ruud Riem-Vis. Cold Chain Management using an Ultra Low Power Wireless Sensor Network. In *MobiSys Workshop on Applications of Mobile Embedded Systems (WAMES)*, Boston, USA, 2004.

[154] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 1978.

[155] Ronald L. Rivest. RFC 1321 - The MD5 Message-Digest Algorithm, April 1992.

[156] Martin Roesch. Snort – Lightweight Intrusion Detection for Networks. In *13th Systems Administration Conference (LISA)*. USENIX, 1999.

[157] Kay Römer. *Time Synchronization and Localization in Sensor Networks*. PhD thesis, ETH Zürich, 2005. Diss. ETH No. 16106.

[158] N. Sastry, U. Shankar, and D. Wagner. Secure Verification of Location Claims. In *WiSe'03*. ACM, 2003.

[159] Edward Sazonov, Kerop Janoyan, and Ratan Jha. Wireless Intelligent Sensor Network for Autonomous Structural Health Monitoring. In *Proceedings of SPIE: Smart Structures and Materials 2004: Smart Sensor Technology and Measurement Systems*, volume 5384. The International Society for Optical Engineering, 2004.

[160] Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to Context than Location. *Computer & Graphics*, 23(6):893–902, December 1999.

[161] Stefan Schmidt, Holger Krahn, Stefan Fischer, and Dietmar Wätjen. A Security Architecture for Mobile Wireless Sensor Networks. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS)*, volume 3313 of *LNCS*. Springer-Verlag, 2004.

[162] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4, 1991.

[163] Philips Semiconductors. SmartMX (smartcard processor) product brief. `http://www.semiconductors.philips.com/products/identification/smartmx/index.html`, 2005.

[164] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. SWATT: SoftWare-based ATTestation for Embedded Devices. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2004.

[165] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[166] Gaurav Sharma and Ravi Mazumdar. Hybrid Sensor Networks: A Small World. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 366–377. ACM Press, 2005.

[167] Frank Siegemund, Christian Flörkemeier, and Harald Vogt. The Value of Handhelds in Smart Environments. *Personal and Ubiquitous Computing*, October 2004.

[168] Sankalp Singh, James Lyons, and David M. Nicol. Fast Model-Based Penetration Testing. In *Proc. of the 2004 Winter Simulation Conference*. IEEE, 2004.

[169] Thorsten Staake, Frédéric Thiesse, and Elgar Fleisch. Extending the EPC Network – The Potential of RFID in Anti-Counterfeiting. In *Symposium on Applied Computing (SAC)*. ACM Press, 2005.

[170] Frank Stajano. *Security in Ubiquitous Computing*. Wiley, 2002.

[171] Frank Stajano and Ross J. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proc. of the 7th Int. Workshop on Security Protocols*, volume 1796 of *LNCS*, pages 172–194. Springer, 1999.

[172] Thanos Stathopoulos, Tyler McHenry, John Heidemann, and Deborah Estrin. A Remote Code Update Mechanism for Wireless Sensor Networks. Technical Report CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Sensing, 2003. http://www.isi.edu/~johnh/PAPERS/Stathopoulos03b.html.

[173] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160. ACM Press, 2001.

[174] Mario Strasser. Intrusion Detection and Failure Recovery in Sensor Networks. Master's thesis, ETH Zürich, 2005.

[175] Adam Stubblefield, Joah Ioannidis, and Aviel D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. In *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2002.

[176] TinyOS. www.tinyos.net.

[177] Ubimon: Ubiquitous monitoring environment for wearable and implantable sensors. http://www.ubimon.net/, 2003–2006.

[178] P. C. van Oorschot. Message Authentication by Integrity with Public Corroboration. In *NSPW '05: Proceedings of the 2005 Workshop on New Security Paradigms*, pages 57–63. ACM Press, 2005.

[179] John Viega and Gary McGraw. *Building Secure Software*. Addison-Wesley, 2002.

[180] Harald Vogt. Integrity Preservation for Communication in Sensor Networks. Technical Report 434, ETH Zürich, Institute for Pervasive Computing, February 2004.

[181] Harald Vogt. Exploring Message Authentication in Sensor Networks. In *Proceedings of ESAS 2004 (1st European Workshop on Security in Ad-hoc and Sensor Networks)*, volume 3313 of *LNCS*, Heidelberg, Germany, August 2005. Springer-Verlag.

[182] Harald Vogt. Increasing Attack Resiliency of Wireless Ad Hoc and Sensor Networks. In *25th IEEE International Conference on Distributed*

Computing Systems, Workshops, 2005, pages 179–184, Columbus, OH, USA, June 2005. IEEE Computer Society.

[183] Harald Vogt, Matthias Ringwald, and Mario Strasser. Intrusion Detection and Failure Recovery in Sensor Nodes. In *Tagungsband INFORMATIK 2005, Workshop Proceedings*, volume P-68 of *Lecture Notes in Informatics*, pages 161–163, Bonn, Germany, September 2005. Gesellschaft für Informatik.

[184] Harald Vogt, Michael Rohs, and Roger Kilian-Kehr. *Middleware for Communications*, chapter Middleware for Smart Cards. John Wiley & Sons, 2004.

[185] Thiemo Voigt, Adam Dunkels, and Torsten Braun. On-demand Construction of Non-interfering Multiple Paths in Wireless Sensor Networks. In *INFORMATIK 2005, Band 2*, volume P-68 of *Lecture Notes in Informatics*, pages 150–154. Gesellschaft für Informatik, 2005.

[186] David Wagner. Resilient Aggregation in Sensor Networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN)*. ACM Press, 2004.

[187] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM)*. ACM Press, 2005.

[188] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In *Advances in Cryptology – Crypto 2005: 25th Annual International Cryptology Conference*, volume 3621 of *LNCS*. Springer-Verlag, 2005.

[189] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer S. J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *Computer*, pages 44–51, January 2001.

[190] 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 (SASN)*. ACM Press, 2004.

[191] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 2003.

[192] Eric W. Weisstein. Birthday Problem. MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/BirthdayProblem.html.

[193] Eric W. Weisstein. Line Line Picking. MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/LineLinePicking.html.

[194] Eric W. Weisstein. Square Line Picking. MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/SquareLinePicking.html.

[195] Eric W. Weisstein. Student's t-Distribution. MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/Studentst-Distribution.html.

[196] Birthday paradox. Wikipedia, http://en.wikipedia.org/wiki/Birthday_paradox, accessed March 2007.

[197] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. IETF, RFC 4408, http://www.ietf.org/rfc/rfc4408.txt, April 2006.

[198] A. D. Wood and J. A. Stankovic. Denial of Service in Sensor Networks. *Computer*, 35(10):54–62, October 2002.

[199] Xiping Yang, Keat G. Ong, William R. Dreschel, Kefeng Zeng, Casey S. Mungle, and Craig A. Grimes. Design of a Wireless Sensor Network for Long-term, In-Situ Monitoring of an Aqueous Environment. *Sensors*, 2:455–472, November 2002.

[200] T. Ylonen. The Secure Shell (SSH) Protocol Architecture. IETF RFC 4251, January 2006. http://www.ietf.org/rfc/rfc4251.txt.

[201] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report USC-CSD-TR-01-745, UCLA Computer Science Department, May 2001.

[202] Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In *Proc. of the 6th annual international conference on Mobile computing and networking (MOBICOM)*, pages 275–283. ACM Press, 2000.

[203] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, 1999.

[204] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing Pair-wise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach. Technical Report ISE-TR-03-01, George Mason University, March 2003.

[205] 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 Communications Security (CCS)*, pages 62–72. ACM Press, 2003.

[206] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data Injection in Sensor Networks. In *IEEE Symposium on Security and Privacy*. IEEE, 2004.

Curriculum Vitae

Harald Vogt

Personal Data	
Date of birth	November 30, 1970
Place of birth	München, Germany
Citizenship of	Germany and Switzerland
Education	
1977 – 1979	Bergschule, Heidenheim a. d. Brenz, Germany
1979 – 1981	Silcherschule, Heidenheim a. d. Brenz, Germany
1981 – 1990	Hellenstein-Gymnasium, Heidenheim a. d. Brenz, Germany
May 1990	Abitur
1991 – 1997	University of Ulm, Germany, Study of Computer Science
July 1997	Graduation as Diplom-Informatiker (Master of Science)
2000 – 2005	ETH Zürich, Switzerland Ph.D. Student at the Department of Computer Science
Military Service	
1990 – 1991	4./PzGrenBat 302, Ellwangen/Jagst, Germany
Employment	
1997 – 1999	Deutsche Telekom AG, Darmstadt, Germany
2000 – 2005	ETH Zürich, Switzerland
since 2006	SAP AG, Karlsruhe, Germany